UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

Instituto de Ingeniería y Tecnología

Departamento de Ingeniería Eléctrica y Computación

PROTOTIPO WEB PARA LA ELABORACIÓN DE DIAGRAMAS DE FLUJO Y MÓDULO DE CODIFICACIÓN EN RUST

Reporte Técnico de Investigación presentado por:

Sandra Luz Yañez Rivera 121052

Requisito para la obtención del título de:

INGENIERO EN SISTEMAS COMPUTACIONALES

Asesor:

Dr. Everardo Santiago Ramírez

Co-asesor:

M. A. T. I. René Noriega Armendáriz

Ciudad Juárez, Chihuahua 3 de junio de 2022

Ciudad Juárez, Chihuahua, a 3 de Junio de 2022

Asunto: Liberación de Asesoría

Mtro. Ismael Canales Valdiviezo
Jefe del Departamento de Ingeniería
Eléctrica y Computación
Presente.-

Por medio de la presente me permito comunicarle que, después de haber realizado las asesorías correspondientes al reporte técnico PROTOTIPO WEB PARA LA ELABORACIÓN DE DIAGRAMAS DE FLUJO Y MÓDULO DE CODIFICACIÓN EN RUST, del alumno Sandra Luz Yañez Rivera de la Licenciatura en Ingeniería en Sistemas Computacionales, considero que lo ha concluido satisfactoriamente, por lo que puede continuar con los trámites de titulación intracurricular.

Sin más por el momento, reciba un cordial saludo.

Atentamente:

Dr. Everardo Santiago Ramírez
Profesor Investigador

Ccp: Mtro. David Absalón Uruchurtu Moreno
Coordinador del Programa de Sistemas Computacionales
Sandra Luz Yañez Rivera
Archivo

Ciudad Juárez, Chihuahua, a 3 de Junio de 2022

Asunto: Autorización de publicación

C. Sandra Luz Yañez Rivera

Presente.-

En virtud de que cumple satisfactoriamente los requisitos solicitados, informo a usted que se autoriza la publicación del documento de PROTOTIPO WEB PARA LA ELABORACIÓN DE DIAGRAMAS DE FLUJO Y MÓDULO DE CODIFICACIÓN EN RUST, para presentar los resultados del proyecto de titulación con el propósito de obtener el título de Licenciado en Ingeniería en Sistemas Computacionales.

Sin otro particular, reciba un cordial saludo.

Dr. Everardo Santiago Ramírez

Profesor Titular de Seminario de Titulación II

Declaración de Originalidad

Yo, Sandra Luz Yañez Rivera declaro que el material contenido en esta publicación fue elaborado con la revisión de los documentos que se mencionan en el capítulo de Bibliografía, y que la solución obtenida es original y no ha sido copiada de ninguna otra fuente, ni ha sido usada para obtener otro título o reconocimiento en otra institución de educación superior.

Sandra Luz Yañez.

Sandra Luz Yañez Rivera

Agradecimientos

Agradezco principalmente a Dios por permitirme llegar a este punto de mi desarrollo escolar y poder realizar una de mis metas de vida. A el Dr. Everardo Santiago Ramírez por su tiempo y dedicación al haberme guiado en este proceso y así poder llevar a cabo este proyecto. A M. A. T. I. René Noriega Armendáriz por su retroalimentación como Co-asesor de este proyecto. A mis profesores y compañeros por compartir su tiempo y conocimientos conmigo durante toda la carrera.

Dedicatoria

Dedico todo mi esfuerzo y dedicación en este proyecto a mi abuelita Lucy que sin su apoyo nada de esto sería posible, a mis padres, mi tía y hermanos que fueron un apoyo incondicional durante todo el proceso. Gracias por siempre confiar en mi.

Resumen

Este proyecto tuvo como objetivo general desarrollar el prototipo de un sistema web para la creación de diagramas de flujo usando la simbología del estándar ISO/ANSI 5807. Además, se implementó un módulo para la escritura de código en el lenguaje de programación Rust. De acuerdo con los resultados, el prototipo desarrollado se despliega adecuadamente en navegadores como Chrome, Brave, Edge, Firefox, Safari y Ópera. Por otra parte, el editor de código para Rust resalta la sintaxis del código.

Palabras claves: Diagramas de flujo, Lenguaje de programación Rust, CodeMirror, Estándar ISO/ANSI 5807.

Índice general

Índice de figuras

Índice de tablas

Índice de códigos fuente

Introducción

Actualmente, las tecnologías de información forman parte de la vida diaria y el ámbito de la educación no es la excepción. Algunas de las aplicaciones en el ámbito educativo para la elaboración de diagramas de flujo tienen algunas limitaciones. Por ejemplo, en [1], [2] y [3] se presentan aplicaciones web y de escritorio para la elaboración de diagramas de flujo, donde incluso una es utilizada por la Universidad Autónoma de Ciudad Juárez (UACJ) y posee un perfil de lenguaje personalizado para la materia de Fundamentos de Programación.

El desarrollo de un sistema web para la elaboración de diagramas de flujo con módulo de codificación Rust se cree sería importante para que los estudiantes empiecen a familiarizarse con este lenguaje de programación que pudiera ser el sustituto del lenguaje de programación C. Este documento presenta el desarrollo de un prototipo web enfocado en la elaboración de diagramas de flujo, utilizando la simbología del estándar ISO/ANSI 5807, con el fin de dar apoyo a los docentes y alumnos de las materias relacionadas a la programación.

El presente documento está dividido en cinco capítulos. En el primero, se plantea el problema y se presentan el objetivo general con el que se pretendió resolverlo. Además, se presenta la justificación, así como los alcances y limitaciones. En el segundo, se muestra el marco teórico haciendo mención a los temas de interés para este proyecto. El tercero presenta el desarrollo del proyecto, la descripción del producto propuesto, la metodología utilizada con sus respectivas fases. El cuarto, muestra los resultados y discusiones, Finalmente, se presentan las conclusiones de este proyecto y trabajos a futuro.

Capítulo 1
Planteamiento del Problema

En este capítulo, en la sección de antecedentes, se describen trabajos que guardan una relación con el proyecto de titulación reportado en este documento. Luego, se define el problema abordado y el objetivo general con el que se pretendió resolverlos. Después de esto, se justifica la realización de este proyecto y, finalmente, se describen los alcances y las limitaciones de la solución propuesta.

1.1. Antecedentes

Un diagrama de flujo es utilizado para describir un proceso, sistema o algoritmo informático. Estos se definen por emplear rectángulos, óvalos, diamantes, entre otras figuras que definen el tipo de paso, y las flechas conectoras determinan el flujo y secuencia [4]. En la simbología de los diagramas de flujo, el rectángulo es el símbolo de proceso y es donde se coloca la acción, función o proceso, el óvalo indica el inicio y el fin de cada diagrama, el diamante es el símbolo de decisión que por lo general, el resultado es un *si/no* o *verdadero/falso*, el conector se representa con un círculo y se utiliza en diagramas complejos haciendo la conexión de elementos separados en una página. La entrada y salida son representados por un trapecio también conocido como el símbolo de datos y muestra los datos disponibles como entrada o salida [5].

Existe una amplia diversidad de herramientas que pueden ser usadas para diseñar diagramas de flujo. En *PowerPoint* es posible generar diagramas de flujo utilizando la herramienta *SmartArt*, la cual despliega una pantalla que contiene diferentes plantillas de diagramas de flujo listos para usarse e introducir los datos necesarios para que el sistema en cuestión funcione. Así mismo la plataforma de *PowerPoint* ofrece otra opción para generar diagramas de flujo desde cero utilizando el menú de *Insertar* figuras. Con este método el usuario coloca las figuras de manera independiente en la diapositiva [1].

SmartDraw por su parte es una de las aplicaciones para aprendizaje visual más completa. Brinda apoyo en la elaboración de Mapas de ideas, Telarañas, Mapas conceptuales, *Diagramas de flujo*, Diagramas de Venn, Organigramas, Redes, Líneas del tiempo, Formatos, *Banners* y Planos. El entorno de trabajo se configura de acuerdo con el diagrama que se esté trabajando en ese momento, es claro e intuitivo, proporciona plantillas para realizar los diagramas o se pueden elaborar desde cero. Los diagramas hechos en esta plataforma pueden ser integrados en documentos de Word, PowerPoint, Excel, Google Docs, Goggle Sheets, por mencionar algunos [2].

El programa DFD es otra herramienta de software derivado de un proyecto colombiano llamado Editor e Intérprete de Algoritmos Representados en Diagramas de Flujo, mediante el arrastre de objetos y permite accionar el programa de manera gráfica. Maneja tipos de datos como: enteros, reales, cadenas de caracteres y lógicos, así como el uso de arreglos [6]. Esta herramienta es catalogada como editor, interprete y depurador de algoritmos representados en diagramas de flujo. Está hecho en el lenguaje C++. A pesar de ser amigable con el usuario y poseer un ambiente de trabajo fácil de utilizar tiene ciertos inconvenientes: es que carece de soporte para el análisis de los problemas y planteamiento de soluciones, solo presenta salidas y entradas en la pantalla de prueba sin mostrar el algoritmo completo a los usuarios, no posee una notación estándar para las estructuras cíclicas como *para y mientras*, no realiza ciclos infinitos y no posee una traducción del diagrama a un lenguaje de programación.

PSeInt, el cual es un intérprete de pseudocódigo y fue desarrollado utilizando el paradigma de programación orientada a objetos, admite tres tipos de datos enteros y reales, carácter y cadena de caracteres y lógico, en cuanto a estructuras de control utiliza *if-then*, *repeat-until*, *do-while*, *case*, *for*, todas expresadas en español. Para la estructura de los datos admite arreglos, de uno, dos o más dimensiones [3]. Actualmente, la UACJ utiliza esta aplicación, la cual posee con un perfil de lenguaje personalizado para la materia Fundamentos de Programación generado por el docente René Noriega Armendáriz. Los perfiles en esta aplicación funcionan como adaptación para cada docente, de acuerdo con las necesidades del curso impartido.

En el ámbito educativo PSeInt está pensado para dar apoyo a los estudiantes en la elaboración de algoritmos computacionales. Dividido por varios módulos los cuales se encargan de analizar, editar y ejecutar el pseudocódigo, generar el diagrama de flujo, traducir el pseudocódigo al lenguaje C++ y actualizaciones del software [7].

Draw.io al igual que *Lucidchart* son aplicaciones web para la creación de diagramas de flujo. Cada uno posee la posibilidad de importar documentos de una plataforma a otra, además de que aceptan otros formatos. *Draw.io* proporciona almacenamiento en la nube, tiene residencia de datos, brinda la posibilidad de bloquear el contenido de los diagramas para que solo la empresa o grupo de personas y la empresa creadora de *Draw.io* tengan acceso a los diagramas [8]. *Lucidchart* por su parte también está basado en el almacenamiento en la nube, posee una interfaz amigable para el usuario, sin importar el dispositivo, navegador o sistema operativo que el usuario esté utilizando. Permite la colaboración en tiempo real proporcionando chat dentro del mismo editor. Es utilizado por más de 6 000 000 de estudiantes y educadores, también puede ser utilizado en conjunto con *Google*, al utilizar *G Suite for Education* o *Google Classroom* se pueden crear y administrar los diagramas [9].

La herramienta *RAPTOR* por su parte surge en 2004 como proyecto de la Fuerza Aérea de Estados Unidos y es definido como un ambiente de programación basado en diagramas de flujo, para ayudar a estudiantes a visualizar algoritmos y evitar el bagaje sintáctico. Fue

desarrollado utilizando los lenguajes Ada y C++ y se ejecuta bajo la plataforma .NET. Su interfaz gráfica permite añadir comentarios a los símbolos del programa, lo que hace posible editar el enunciado del problema. También carece de soporte para el análisis del problema y planteamiento de la solución, no presenta la ejecución completa en pantalla, no utiliza notación estándar para las estructuras cíclicas, no evita ciclos infinitos, no cuenta con las estructuras *Para y Mientras*, su único idioma es el inglés y solo se ejecuta en Windows [10].

Para los modelos educativos, los diagramas de flujo son utilizados para la representación gráfica de los algoritmos, de esta manera el aprendizaje visual es un apoyo para el desarrollo de las habilidades del pensamiento, lo cual implica que el estudiante identifica los pasos a seguir para la resolución de un problema de forma clara y lógica, desarrollan una visión amplia, abren su mente a la posibilidad de encontrar todas las posibilidades que existe para la resolución de dicho problema, logran detectar la conexión que tiene los procesos entre sí para darles un orden lógico correcto, la codificación del algoritmo se vuelve más fácil, hacen que la comprensión de la secuencia lógica a la solución planteada sea más fácil para las demás personas [11].

1.2. Definición del problema

Para un alumno es más fácil realizar un diagrama de flujo de manera mecánica, es decir, usando solamente los recursos que un software le ofrece, dejando de lado sus habilidades de pensamiento rápido y lógico. Además, hay estudiantes que no les es fácil elaborar diagramas de flujo ya que tienen discapacidad visual parcial o completa o problemas para utilizar las manos al escribir causados por artritis o algunos no tienen sus extremidades. La ISO/ANSI 5807 han establecido un conjunto de símbolos estandarizados para la elaboración de diagramas de flujo. Sin embargo, dentro de lo explorado, no se encontró una herramienta que los implemente completamente y que contenga reconocimiento de voz.

1.3. Objetivo general

Desarrollar el prototipo de una plataforma web para la elaboración de diagramas de flujo utilizando la simbología del estándar de la ISO/ANSI 5807 y para la escritura de código, esto con el fin de promover el pensamiento rápido y lógico.

1.3.1. Objetivos específicos

Aquí se presentan los objetivos específicos que se llevaron a cabo para lograr el objetivo general:

- Identificar los requisitos de la plataforma web para la elaboración de diagramas de flujo.
- Diseñar la interfaz gráfica de usuario de una plataforma web para la elaboración de diagramas de flujo.
- Identificar los requerimientos de la base de datos para el almacenamiento de la información de diagramas de flujo.
- Diseñar el diagrama entidad-relación para la base de datos, considerando los objetivos específicos anteriores.
- Transformar el diagrama Entidad-relación a un modelo relacional.
- Construcción de la base de datos mediante la implementación del modelo relacional.
- Identificar características de los motores de reconocimiento de voz y usarlos en el diseño base del controlador.

1.4. Justificación

La forma en cómo los estudiantes crean actualmente diagramas de flujo, de alguna manera impide que las habilidades de pensamiento rápido y lógico no estén presentes en ellos. Esto debido a que tienen herramientas que hacen el proceso de manera automática. El desarrollo del pensamiento rápido y lógico es fundamental en los alumnos para entender el procedimiento que lleva a cabo un algoritmo desde su inicio hasta su término.

En este proyecto se propone desarrollar una base de datos cuyo objetivo es almacenar información de diagramas de flujo bajo el estándar de la ANSI/ISO 5807, para poder transformarlos en un futuro a código. Cuando el proyecto llegue a su término, el reconocimiento de voz será de mucha utilidad para personas con discapacidad visual o con problemas causados por artritis o falta de sus extremidades en la creación de diagramas de flujo.

Impacto Tecnológico: El prototipo desarrollado en este documento dejará las bases para que una vez finalizado los estudiantes y personas dedicadas a la programación tengan una herramienta que les permita elaborar diagramas de flujo mediante la voz. Se espera que esta herramienta sea capaz de potenciar sus habilidades de diseño y optimice su tiempo.

Impacto Académico: El proyecto impacta académicamente ya que los alumnos tendrán la necesidad de utilizar el razonamiento rápido y lógico en la creación de sus diagramas de flujo, por lo que se espera que el nivel académico de los alumnos de las materias relacionadas a programación pueda verse beneficiados con esta propuesta. Además de prevenir la deserción en las materias de programación y coadyuvar a que los alumnos de niveles avanzados tengan conocimientos más acertados de lo que son los diagramas de flujo y como crear estos mismos.

1.5. Alcances y limitaciones

El prototipo que se plantea en este documento fue desarrollado con base en la arquitectura Modelo-Vista-Controlador. Con esto en mente, los alcances que posee la propuesta son:

- Rutinas para almacenar información acerca de diagramas de flujo (modelo).
- Construcción de la interfaz gráfica de usuario para la plataforma web (vista).
- Construcción de un módulo de codificación en el lenguaje Rust.

Los aspectos que quedan fuera de la propuesta son los siguientes:

- Controlador no desarrollado en su totalidad.
- El prototipo a desarrollar no estará disponible inmediatamente para su uso.
- Al ser un proyecto a realizarse por distintas personas, la implementación completa del reconocimiento de voz queda fuera de la propuesta que se presenta en este documento.

Capítulo 2
Marco teórico

En este capítulo se exponen conceptos que fundamentan el desarrollo y comprensión del proyecto propuesto. En la sección 3.1 se aborda el tema de diagramas de flujo y el estándar de la ISO/ANSI 5807. En la sección 3.2 se presenta el tema de bases de datos. En la sección 3.3 se hace mención de las aplicaciones web y por último en la sección 3.4 se menciona el reconocimiento de voz.

2.1. Estándar ANSI e ISO para la elaboración de diagramas de flujo

Los diagramas de flujo son una herramienta utilizada para la representación visual de algoritmos para la solución de un problema. Estos diagramas utilizan símbolos especiales que representan los pasos o procedimientos a realizar para la solución de un problema dado. Los diagramas de flujo son un canal de entendimiento entre el programador y el usuario, permiten la detección anticipada de posibles errores, los cuales pueden presentarse al implementar un algoritmo. La estructura de los diagramas de flujo está compuesta por símbolos básicos, tales como el óvalo que indica el inicio y el fin del diagrama de flujo, un trapecio que es donde se introducen los datos de entrada que va a tener el diagrama de flujo, el rectángulo

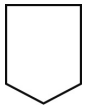
es donde se desarrollan los procesos u operaciones que el algoritmo contenga, el rombo es para la toma decisiones dentro del algoritmo, por ejemplo, tener una situación en donde el algoritmo necesite una aceptación o negación para realizar determinada operación, el símbolo de impresión como su nombre lo dice es aquí donde se muestran los resultados del algoritmo, y por último las flechas que indican el flujo de datos dentro del algoritmo. Estos símbolos se muestran en la Tabla 2.1.

Tabla 2.1: Símbolos principales en la creación de diagramas de flujo.

Símbolo	**Concepto**
	Inicio / Fin
	Entrada de datos
	Proceso
	Decisión
	Impresión de datos
	Flujo de datos

Hasta 1985 eran utilizados los símbolos de la ANSI e ISO por separado, hasta que se hizo la observación de que ambas organizaciones tenían símbolos que realizaban los mismos procesos y que se podía hacer una estandarización que tuviera símbolos de ambas. Fue así como se tomó la decisión de tomar los símbolos de ambas organizaciones y crear la estandarización ISO/ANSI 5807 eliminando los símbolos que realizaban procesos iguales [11]. Estos símbolos se muestran en la Tabla 2.2.

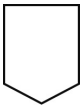
Tabla 2.2: Simbología implementada por la ISO/ANSI 5807 en 1985 para la creación de diagramas de flujo.

Símbolo	**Concepto**
	Inicio / Final
	Entrada / Salida de datos
	Entrada por teclado
	Llamada a subrutina
	Acción / Proceso
	Flujo
	Decisión
	Iteración
	Salida Impresa
	Salida en pantalla
	Conector
	Conector
	Cruce de líneas

A pesar de ser creada la ISO/ANSI 5807 en 1985, cada organización tiene sus símbolos y hay quienes prefieren utilizarlos a cada uno por separado. Los símbolos de la ANSI son utilizados para diagramas de flujo utilizando símbolos como los que se muestran en la Tabla 2.1. El estándar ANSI utiliza también el símbolo de documento que representa un documento que entra al proceso y se utilice ya sea que genere cambios o simplemente salga del proceso, el símbolo de triángulo invertido representa el almacenamiento de un archivo temporal y

permanentemente, por último, los conectores que representan la conexión a otra hoja diferente en el caso de conector de página y otro que representa la conexión a alguna parte del diagrama como se muestra en la Tabla 2.3.

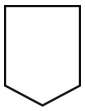
Tabla 2.3: Simbología de la ANSI para la creación de diagramas de flujo.

Símbolo	**Concepto**
	Inicio / Fin
	Operación / Actividad
	Documento
	Datos
	Almacenamiento / Archivo
	Decisión
	Líneas de flujo
	Conector
	Conector de página

Por último, en la Tabla 2.4, se muestra la simbología que utiliza el estándar de la ISO para los diagramas de flujo basados en la Norma 5807 que utiliza un círculo como operación que es donde se indican las principales métodos o procedimientos, un círculo dentro de un cuadro que representa la verificación o supervisión durante los procesos, un cuadro que representa la verificación de la los datos, una flecha que representa el movimiento de un documento, el triángulo invertido indica los datos que entran a proceso, el triángulo que representa la permanencia de un documento, utiliza también la demora la cual indica cuando un proceso es detenido por la espera de una acción o por el tiempo de respuesta, también tiene conector de página que representa la continuación del diagrama en una hoja nueva y el conector que

nos desplaza entre las distintas fases del diagrama de flujo.

Tabla 2.4: Simbología utilizada por la ISO para diagramas de flujo.

Símbolo	**Concepto**
	Operación
	Operación e Inspección
	Inspección y Medición
	Transporte
	Entrada de bienes
	Almacenamiento
	Decisión
	Líneas de flujo
	Demora
	Conector
	Conector de página

2.2. Patrón de arquitectura Modelo-Vista-Controlador

Creado en 1979 por Trygve M.H. Reenskaug, este modelo reduce el esfuerzo de programación necesario para la implementación de sistemas múltiple y sincronizados. Tiene como características el modelo, las vistas y los controladores que son tratados como entidades se-

paradas, lo que ocasiona que cualquier cambio que se produzca en el modelo también se vea reflejado en las vistas que se generen.

Las ventajas que presenta este modelo son:

- Los componentes del programa tienen una separación, por lo que pueden ser implementados por separado.
- Su interfaz está muy bien definida, cualquier persona que la utilice puede reemplazar el modelo, la vista o el controlador.
- La conexión entre el modelo y las vistas se da en tiempo de ejecución.

En la etapa del modelo es donde se trabaja con los datos, es aquí donde se obtienen mecanismos para el acceso a la información y así mismo poder actualizarla. Normalmente los datos están almacenados en una base de datos por lo que en la parte del modelo es donde se tienen las funciones que realizaran los *selects, updates, inserts*, los cuales se trabajan en las tablas.

La vista por su parte contiene el código de la aplicación que va a desarrollar la visualización de las interfaces del cliente, es decir, el código que ayuda a interpretar los estados de la aplicación. Las vistas obtendrán los datos del modelo para generar las salidas que la aplicación requiera.

El controlador es la capa de enlace entre el modelo y las vistas. Es en esta etapa de modelo-vista-controlador donde el código que atiende ciertas acciones de la aplicación que van desde cómo se visualiza cierto elemento, cómo se busca información dentro de la aplicación, entre otros. En la Figura 2.1 se muestra cómo trabaja el patrón de arquitectura modelo-vista-controlador.

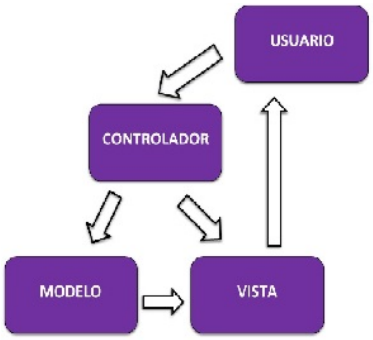


Figura 2.1: Arquitectura del Modelo-Vista-Controlador.

2.3. Bases de datos

Una base de datos consiste en el almacenamiento de datos los cuales están organizados mediante una estructura de datos. Cada base de datos es diseñada para cumplir con los requisitos de información de una determinada organización. Anteriormente, las bases de datos eran elaboradas con sistemas de ficheros, los cuales surgen para automatizar el manejo de archiveros manuales, con esto el acceso a los datos se hizo más eficiente [12]. Las bases de datos relacionales representan la información en forma de entidades y relaciones. Cada entidad y relación son creadas por tablas bidimensionales con sus respectivas filas y columnas. En la actualidad se tiene acceso a las bases de datos relacionales mediante herramientas como MySQL [13]. Las relaciones tienen ciertas características:

- Cada relación tiene nombre, y este debe ser distinto a las demás.
- Los dominios donde se describen los atributos son escalares, por tanto, los valores de los atributos son atómicos, donde cada tupla y cada atributo posee un solo valor y estas relaciones debe estar normalizadas.
- No existen atributos con el mismo nombre.

- Los atributos pueden estar ordenados o no.
- No hay tuplas duplicadas, por lo que cada una es distinta a las demás.
- Al igual que los atributos, las tuplas pueden o no estar ordenadas.

Para los sistemas gestores de bases de datos hay dos tipos de relaciones:

- Las relaciones base tienen nombre y son parte de la base de datos.
- Vistas, también llamadas relaciones virtuales, tiene nombre y derivadas. Por derivadas significa que son obtenidas de otras relaciones, no contienen datos almacenados propios, los datos que poseen están relacionados a los que están almacenados en las relaciones base [12].

El modelado de datos facilita la descripción de la base de datos incorporando los datos, las relaciones, restricciones [14]. En los modelos de datos también es posible hacer operaciones para la realización de consultas y actualizaciones de datos. Estos modelos pueden ser clasificados de acuerdo con los conceptos que ofrece la estructura de datos integrando una jerarquía de niveles. Los de alto nivel también llamados modelos conceptuales, los cuales disponen de conceptos cercanos al modo que la mayoría de los usuarios perciben los datos y los de nivel bajo o modelos físicos describen a detalle cómo es que se almacenan los datos en las computadoras, estos modelos están dirigidos al personal informático. Entre estos dos niveles se encuentran los modelos lógicos en donde los conceptos son entendidos por los usuarios finales, acercándose a cómo los datos están organizados físicamente, ocultando detalles de cómo se almacenan y pueden ser implementados directamente en un Sistema Gestor de Bases de Datos (SGBD).

El diagrama entidad-relación, también llamado modelo semántico, está dentro del nivel conceptual y es donde se especifica qué es lo que se va a representar en la base de datos. La principal característica de este modelo es que provee un modelo gráfico de la estructura

conceptual de la base de datos. En la Figura 2.2 se muestra un ejemplo de lo que es el diagrama entidad relación.

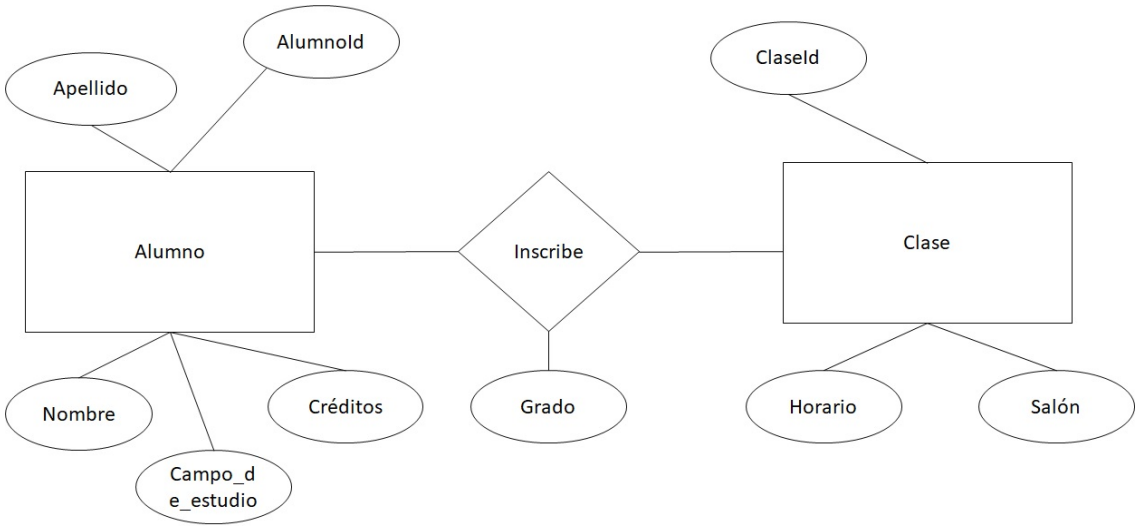


Figura 2.2: Ejemplo de un diagrama entidad-relación [14].

Para cada relación existen diferentes restricciones, es decir que una relación puede poseer ciertas características de otra relación. Estas restricciones son conocidas como *cardinalidades* las cuales se distinguen de la siguiente manera:

- Uno a uno: Una entidad A puede estar reaccionada con una entidad B y viceversa. Por ejemplo, si se tiene un entidad llamada Presidente y otra llamada País tenemos una cardinalidad de uno a uno ya que un país puede ser gobernado por un solo presidente.
- Uno a muchos: Una entidad A puede estar relacionada con cualquier entidad B y una entidad B puede estar relacionada con solo una entidad A. Por ejemplo, se tiene una entidad Cliente y otra llamada Cuenta, se puede apreciar que un Cliente puede tener muchas cuentas, pero una Cuenta puede llegar a pertenecer a un solo Cliente.
- Muchos a uno: Una entidad B puede estar relacionada con cualquier entidad A mientras que una entidad A puede estar relacionada con solo una entidad B. Si se tiene una

entidad llamada Madre y otra llamada Hijo, se puede notar la cardinalidad muchos a uno ya que un Hijo tiene una sola Madre, pero una Madre puede tener muchos Hijos.

- Muchos a muchos: Cualquier cantidad de entidades A está relacionada con cualquier cantidad de entidades B. Por ejemplo, con un entidad llamada Empleado y otra llamada artículo, en donde un Empleado puede vender muchos artículos y un artículo puede ser vendido por muchos empleado [14].

Luego de haber concluido con el modelo entidad relación se procede a realizar el modelo relacional, el cual se encuentra dentro de la etapa de diseño lógico. Es en este punto donde se lleva a cabo la implementación de la base de datos, donde se pretende obtener las tablas de nuestra base de datos partiendo de un diagrama E-R, para cada tabla se establecen las claves primarias, las claves alternativas, las foráneas, además se establecen restricciones que determinan la función de las tablas. La estructura de este modelo está basada en tablas con relación entre ellas. Con este modelo las entidades y relaciones son indicados usando tablas, donde las relaciones se manifiestan como tablas bidimensionales y las filas corresponden a los registros que se almacenaran en la base de datos y por último las columnas son los atributo. Tomando en cuenta la entidad Alumno del diagrama presentado en la Figura 2.1, en la Tabla 2.5, se muestra la transformación a el modelo relacional de esta entidad en específico.

Tabla 2.5: Transformación de la entidad Alumno a modelo relacional.

Alumno	**Apellido**	**Nombre**	**Campo_de_estudio**	**Créditos**
Al00	López	María	Historia	56
Al01	Martínez	Valentín	Matemáticas	90
Al02	Dominguez	Roberto	Artes	0
Al03	Renteria	Brenda	Ciencias Sociales	63
Al04	Flores	Irene	Ciencias de la Computación	15
Al05	Zaragoza	Ramón	Historia	36
Al06	Sánchez	Daniela	Matemáticas	42

Una vez terminado el modelo relacional, se prosigue a la parte de la *Normalización* en donde se busca que las tablas contengan un formato más exacto, por lo que serán menos vulnerables a anomalías de actualización que puedan presentarse. Para la implementación de estas formas de normalización es necesario que las tablas se encuentren en la primera forma normal, de lo contrario no será posible realizar la implementación. Es recomendable alcanzar como mínimo la tercera forma normal. Llegando a la tercera forma normal podrá decir que la base de datos es óptima para su uso.

En la *normalización* es fundamental la *dependencia funcional*, la cual muestra la relación que existe entre los atributos de cada tabla. Para decir que una tabla está normalizada es necesario ir comprobando que en cada tabla se realicen ciertas reglas, cada que se cumple una regla, el grado de normalización aumenta. Si por alguna razón una regla no se cumple, es necesario descomponer la tabla en varias tablas para que la regla se cumpla.

- Primera forma normal (1FN): Se dice que una tabla está en 1FN si y solo si todos sus atributos contienen valores atómicos, es decir, no existen grupos repetitivos. Para eliminar los grupos repetitivos es necesario tomar cada grupo que se repita y colocar cada uno en una tabla diferente, quedando la clave primaria de la tabla original como herencia.

- Segunda forma normal (2FN): Solo se puede llevar una tabla a 2FN siempre y cuando esta tabla se encuentra en 1FN, además que cada atributo que no pertenezca a la clave primaria sea dependiente de la clave primaria. Las claves primarias de las tablas en 2FN están compuestas por uno o varios atributos. Para lograr la 2FN en necesario eliminar las dependencias parciales que existen en la clave primaria, por lo que los atributos que son funcionalmente dependientes son colocados en una tabla nueva generando una copia de su determinante, cierto determinante está formado por los atributos dela clave primaria de los que depende.

- Tercera forma normal (3FN): En esta tercera forma normal se busca encontrar dependencias transitivas de los atributos con la clave primaria. Para pasar de 2FN a 3FN se deben eliminar las dependencias transitivas, es decir, se eliminan los atributos que contienen dependencia transitiva y se colocan en una relación nueva incluyendo su determinante.

- Cuarta forma normal o Forma normal de Boyce-Codd (BCFN): Al igual que en las formas normales anteriores, para llegar a BCFN es necesario que las tablas se encuentren en 3FN. En esta forma normal se busca eliminar dependencias parciales y dependencias transitivas de la clave primaria, es posible que estas dependencias existan en claves candidatas siempre y cuando estas existan.

Si se requiere comprobar si la normalización está bien hecha, se debe poner atención a las nuevas dependencias funcionales no deseadas, ya que solo deben permanecer las de la clave primaria completa. Otra de las tareas que se ejecutan en el diseño lógico, luego de haber obtenido la normalización correspondiente, es considerar las redundancias controladas y otros cambios dentro del esquema. Para llegar a tener un esquema estructural consistente y sin redundancias es necesario llegar hasta las 3FN, aunque puede darse el caso de que las bases de datos normalizadas hasta 3FN no facilitan una máxima eficiencia, por lo que es necesario aplicar la desnormalización.

En dado caso que se requiera utilizar la desnormalización debe tomarse en cuenta de que el sistema no consigue los servicios deseados. Además, se deben tener en cuenta los siguientes aspectos:

- La implementación de la desnormalización es mas compleja.

- Sacrifica la flexibilidad.

- Los accesos a los datos son mas rápidos, pero las actualizaciones son mas lentas.

Después de realizado el diseño lógico de la base de datos, se prosigue a el diseño físico, que es donde se elabora la descripción de la implementación de las bases de datos [12]. En esta etapa de la creación de bases de datos se selecciona un SGBD, donde se crearán las tablas, el tipo de datos que se almacenaran en las tablas, restricciones por ejemplo para especificar si existen datos NULL, UNIQUE, claves primarias, entre otros. En el SGBD también es posible generar las vistas que se tendrán en la base de datos, los índices si es que existen, se pueden generar consultas a las tablas, así como también eliminar, insertar, modificar y monitorizar los datos mediante disparadores [14]. Luego de realizar todas estas actividades con el SGBD, se puede proceder a la parte del diseño de la interfaz de la base de datos en el lenguaje, esto con la intención de crear los menús con los que el usuario pueda navegar en la aplicación final.

La interfaz es donde se podrán apreciar por ejemplo las vistas creadas en el SGBD, se les dan a los usuarios distintos roles o perfiles los cuales les proporcionan acceso a diferentes datos dentro de la base de datos, dependiendo de su rango en la empresa u organización.

2.4. Aplicaciones web

Las aplicaciones web están conformadas por la arquitectura cliente servidor, por lo que el cliente comprende a el navegador o visualizador y el servidor comprende a el servidor web utilizando el protocolo HTTP. El cliente es la parte de código que interactúa con el usuario el cual hace solicitudes a los servidores web para adquirir recursos por medio de HTTP. Normalmente, la parte del cliente está conformada por código HTML el cual brinda la visualización de la página web en conjunto con lenguaje *script* de los navegadores y *plugins* con los cuales se puede visualizar el contenido multimedia que pudiera tener la página web.

En la Figura 2.3 se muestra la arquitectura cliente-servidor utilizada en las aplicaciones web.

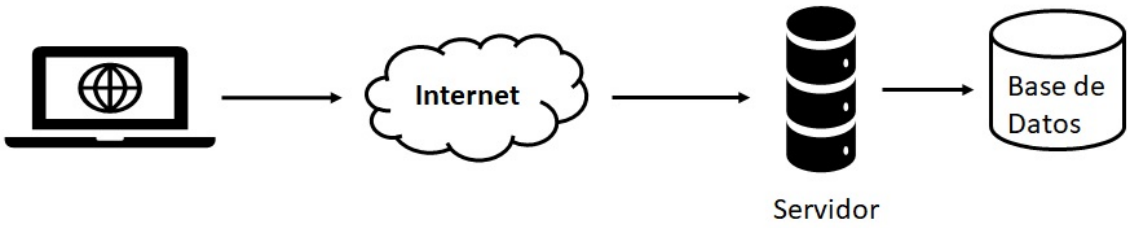


Figura 2.3: Arquitectura Cliente-Servidor [15].

Para el desarrollo de aplicaciones web en lo que comprende al cliente se suelen utilizar las siguientes herramientas:

- HTML
- CSS
- Lenguajes de *script* como *JavaScript*
- *Plugins*

El servidor web es un programa que está a la espera de solicitudes hechas por los clientes por medio de HTTP. Dicho servidor web está conformado por documentos HTML, los cuales muestran el contenido a los usuarios, también contienen multimedia o documentos adicionales los cuales pueden ser aplicados en las páginas web o pueden estar a la espera de ser descargados para después ser visualizados por el usuario. El servidor ejecuta los *scripts* cuando el usuario hace la petición a el navegador la mayoría de las veces la salida de estos *scripts* son HTML aunque algunas veces puede haber acceso a bases de datos [16].

Son cuatro las generaciones en las cuales se clasifican los sitios web:

- Primera generación: Surgen desde la creación de la Web en 1992 hasta la mitad de 1994. La creación de estos sitios era limitada por la tecnología existente en esos momentos, el

ancho de banda, los navegadores no estaban totalmente desarrollados y había monitores monocromos. A pesar de estas limitantes las páginas web tenían ciertas características: Cargaban rápidamente, la navegación no era saturada, las páginas eran muy largas que parecía que no tenían fin, el texto que presentaban parecía como si lo hubieran escrito en papel, los saltos de línea funcionaban como separadores, las líneas horizontales separaban secciones, la información era organizada por medio de listas, poco uso de enlaces, había enlaces intradocumentales, listas muy largas con enlaces a otros sitios, pueden apreciarse en casi todos los navegadores.

- Segunda generación: Son aquellos sitios generados a partir de 1995 hasta la actualidad, lo novedoso en estos sitios fueron los elementos gráficos, sus características principales son: El tiempo de carga es lento, el color de fondo ya no es gris o blanco, se implementan las tablas, su estructura es de arriba a abajo, su navegación es a partir de una página principal y de forma jerárquica, aparecen tecnologías multimedia.

- Tercera generación: Aparecen a mediados de 1996 y son más parecidas a las que conocemos hoy en día. Se implementa CSS (Hojas Estilo Cascada) y el tiempo de carga es más rápido, las páginas están limitadas a una sola pantalla sin necesidad de navegar de arriba a abajo, los sitios web tienen objetivo y van dirigidos a ciertos usuarios, el número de enlaces es menor, los principios de usabilidad y accesibilidad son incorporados.

- Cuarta generación: Se empieza a desarrollar desde 1999 hasta la actualidad, teniendo como características: excesos en recursos gráficos, se aprovecha hasta el último píxel, HTML evoluciona, se intensifica el uso de CSS y aparece DHTML (HTML Dinámico), hay nuevas tecnologías multimedia, aumento del ancho de banda [16].

Las aplicaciones web están basadas en la arquitectura cliente-servidor, donde el cliente es el navegador, explorador o visualizador y el servidor es el servidor web. Las diferentes arquitecturas son:

- Todo en servidor: Un solo ordenador hospeda el servicio HTTP, la lógica de negocio y los datos.
- Servidor de datos separado: Tomando la arquitectura anterior, aquí se separa la lógica de datos a un servidor de bases de datos.
- Todo en un servidor con aplicaciones: Con la primera arquitectura la lógica de negocio se separa de HTTP e incluye el servicio de aplicaciones que gestionan los procesos para la implementación lógica de negocio.
- Servidor de datos separados con servicio de aplicaciones: Partiendo de la arquitectura anterior, se aparta la lógica de datos y los datos de un servidor.
- Todo separado: Las tres funcionalidades básicas de los servidores web se separan entres diferentes servidores específicos [16].

Así como se tiene diferentes arquitecturas, existen diferentes entornos para desarrollar las páginas web:

- Editores de texto plano: Son editores simples de utilizar tales como un Bloc de notas en Windows o Gedit en GNU/Linux.
- Editor de texto con ventanas desplegadas: Son intérpretes de código HTML con resultados en tiempo real tales como Bluefish, BlueGriffon en Linux o Coda en Mac.
- Editores WYSIWYG: Lo que ves es lo que tienes. Se puede trabajar directamente con el resultado en la página como si se trabajara con un procesador de textos.
- Editores en línea: Permiten la edición del código directo en el navegador lo que facilita subirlo al servidor y la colaboración con otras personas [17].

2.5. Reconocimiento de voz

En la actualidad es cada vez más común interactuar con los dispositivos móviles mediante el reconocimiento de voz, por ejemplo, se tienen asistentes como Siri o Alexa con los cuales se puede configurar la electrónica de los hogares a través de dispositivos móviles y dando instrucciones con solo utilizar la voz.

En el caso de la creación de software se puede contar con aplicaciones como *Serenade* y *Talon* los cuales están a la cabeza en el mercado de software. *Serenade* por ejemplo fue creado por Matt Wiethoff después de tener que tomar la decisión de dejar su carrera como ingeniero de software por problemas de lesión por esfuerzo repetitivo donde se ven afectados músculos, ligamentos, tendones y nervios causadas por técnicas inadecuadas o uso excesivo de estas técnicas, provocadas por los malos hábitos a la hora de estar sentados frente a un ordenador. *Talon* creado por Ryan Hileman quien también se vio obligado a dejar su trabajo en 2017 por fuertes dolores en las manos, es una aplicación de codificación a manos libres donde se pretende sustituir por completo el ratón y el teclado, utilizando reconocimiento de voz, seguimiento ocular y reconocimiento de ruido. La arquitectura de *Serenade* cuenta con un convertidor de voz a texto el cual fue desarrollado para reconocer código.

La codificación por voz ayuda también a personas con discapacidad visual en la creación de software, además de tener la opción de transformar lo que una persona piensa en código [18].

Hace tres años Harold Pimentel y Naomi Saphra ambos con problemas en las manos a la hora de escribir desarrollaron un software llamado *VoiceCode* el cual daba la facilidad a los usuarios de hacer sus propias configuraciones y comandos personalizados para la creación de sus programas. Pimentel llegó a aprender 40 páginas de comandos mientras que Saphra después de practicar por dos meses aprendió el manejo de fórmulas en *LaTeX*. Ambos desarrolladores destacan que a pesar de que la codificación con voz tiene sus ventajas también

encuentran ciertas desventajas tales como dolores de garganta al tener que usar la voz por tiempos extensos lo que provoca estar constantemente tomando agua para hidratar la garganta, otra desventaja destacada es que después de un tiempo se extraña la forma tradicional de hacer código, y por último un aspecto destacado también fue que muchas personas al crear código escuchan música para mejor concentración y con el reconocimiento de voz no es posible hacerlo [19].

Capítulo 3
Desarrollo del Proyecto

En este capítulo se describe el desarrollo de un prototipo de plataforma web para el diseño de diagramas de flujo bajo el estándar de la ISO/ANSI 5807 y escritura de código en el lenguaje de programación Rust. Primero, en la sección 3.1 se resumen las principales características del prototipo desarrollado como proyecto de titulación. Luego, en las secciones 3.3 a 3.8 se describe el desarrollo del proyecto de acuerdo con cada fase del modelo de prototipos (descrita en la sección 3.2): *Análisis de requerimientos*, *Diseño rápido*, *Construcción del prototipo*, *Evaluación del usuario*, *Refinamiento del prototipo* e *Implementación y mantenimiento*.

3.1. Producto propuesto

El producto propuesto es un prototipo de una plataforma web educativa para la elaboración de diagramas de flujo y módulo de codificación. El módulo de creación de diagramas fue creado utilizando la herramienta Drawflow y este consta de una sección donde el usuario podrá arrastrar los símbolos de la ISO/ANSI 5807, un botón para limpiar la pantalla, un botón que genera un JSON del diagrama el cual se guarda en la computadora en formato js. El módulo de codificación fue creado con la herramienta CodeMirror y configurado para el lenguaje de programación Rust.

3.2. Descripción de la metodología

En el desarrollo del prototipo presentado en este documento se utilizó el modelo de prototipos [20] cuyas fases se muestran en la Figura 3.1. Los resultados de cada fase se ven reflejados en las siguientes secciones.

Figura 3.1: Metodología de prototipos usado para el desarrollo del prototipo propuesto.

3.3. Análisis de requerimientos

El sistema que se propone en este documento será desarrollado por varias personas y en diferentes tiempos, por lo que en este proyecto de titulación se abarcaron únicamente los siguientes requerimientos funcionales y no funcionales. Estos fueron definidos con base en las necesidades de docentes y áreas de oportunidad detectados en estudiantes de materias relacionados a programación.

El prototipo que se reporta en este documento cubre con los siguientes requerimientos funcionales:

- El módulo de creación de diagrama permite al usuario crear diagramas de flujo con la opción de arrastrar y soltar.
- La sección de tutoriales poseerá un conjunto de vídeos de ayuda para usar el sistema o

creación de diagramas de flujo.

- El módulo de codificación Rust permitirá al usuario crear código con este lenguaje.

Los requerimientos no funcionales para el sistema son los siguientes:

- La interfaz gráfica de usuario del prototipo desarrollada en HTML y CSS.
- El prototipo usa la simbología del estándar ISO/ANSI 5807 para los diagramas de flujo.
- El sistema permite al usuario crear diagramas de flujo que serán guardados en formato JSON (*JavaScript Object Notation*).
- El prototipo utiliza la biblioteca Drawflow para el manejo visual de la simbología ISO/ANSI 5807 dentro de la plataforma web.
- El prototipo puede ser ejecutado por el usuario como una aplicación web.
- El módulo de codificacion desarrollado con la herramienta *CodeMirror*.

Los requerimientos funcionales y no funcionales reportados en esta sección son usados en el diseño del prototipo para la creación de diagramas de flujo.

3.4. Diseño rápido

Se realizó una investigación de la simbología creada en 1985 donde se hace una fusión de los símbolos utilizados por la ISO/ANSI 5807 para la creación de diagramas de flujo. Para la realización de la aplicación web, se utilizó el editor de código fuente y texto llamado Atom, utilizando el lenguaje de marcado HTML y CSS para el diseño de las páginas.

La página principal de la plataforma web requirió de los siguientes aspectos:

- Logotipo.
- Nombre de la página.
- Un menú desplegable que contiene las secciones Apoyo, Acerca de, Diagramas y Plantillas.
- Información de la página y los diagramas de flujo.
- Una sección para consultar las nuevas adiciones de la página.
- Una sección *login* para el inicio de sesión de los estudiantes y maestros.

Una vez concluida la etapa anterior, con los requerimientos especificados se procedió con el diseño rápido de la aplicación web y la base de datos. Para la base de datos se requiere almacenar información acerca de los maestros tales como clave de empleado, nombre completo y disciplina, de cada estudiante se desea guardar el nombre completo, correo electrónico, nombre de usuario, contraseña y un ID. El diagrama entidad-relación resultante es mostrado en la Figura 3.2, muestra que cada maestro imparte una materia y cada materia tiene una clave, un nombre y descripción. Los maestros pueden impartir varias materias, pero una materia solo puede ser impartida por un maestro. Cada estudiante está inscrito en una o varias materias.

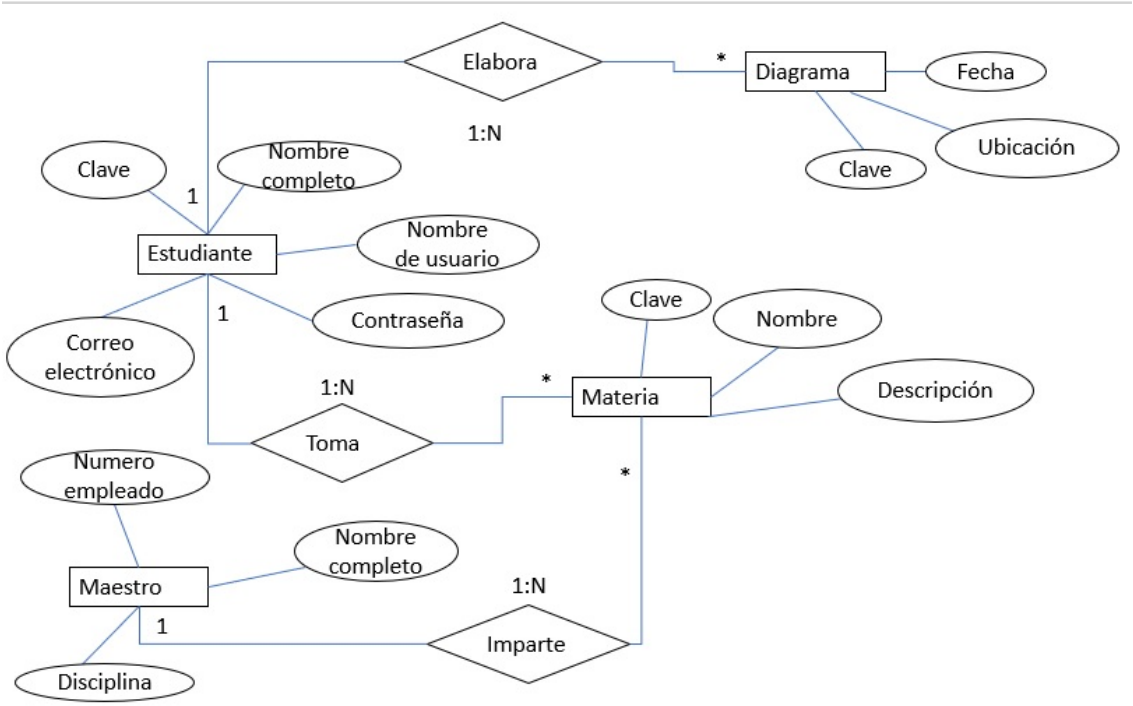


Figura 3.2: Diagrama Entidad-Relación de la base de datos creada para el prototipo.

En la Figura 3.3 se muestra el modelo relacional de la base de datos, es aquí donde cada entidad de la Figura 3.2 se convierte en tablas y los atributos en columnas. La tabla Maestro tiene como columnas la clave de empleado, nombre completo y disciplina. La tabla Materia tiene como columnas, la clave de materia, nombre y la descripción, la tabla Estudiante tienen como columnas ID, nombre completo, nombre de usuario, contraseña y correo electrónico.

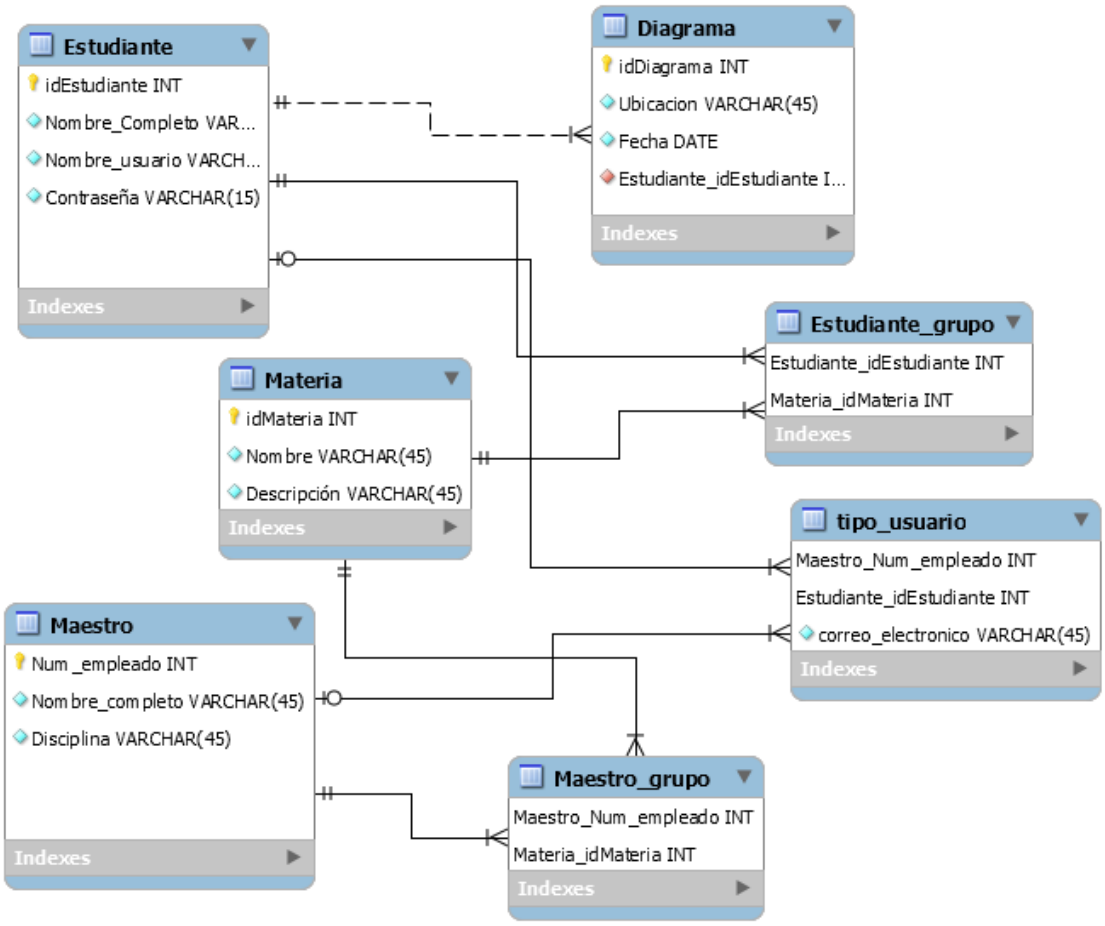


Figura 3.3: Modelo Relacional diseñado para la base de datos del prototipo propuesto.

La página principal de la aplicación web mostrada en la Figura 3.4 cuenta con un logotipo, un menú desplegable, en donde se podrá navegar hacia las otras páginas tales como las plantillas, el área de trabajo para la creación de diagramas y acerca de la página. Además de la sección de *login* con la cual los estudiantes y docentes podrán tener acceso y una sección donde aparecen las actualizaciones que se presenten una vez que la aplicación esté funcionado, así como información de la aplicación la cual es vista al entrar a la aplicación web.

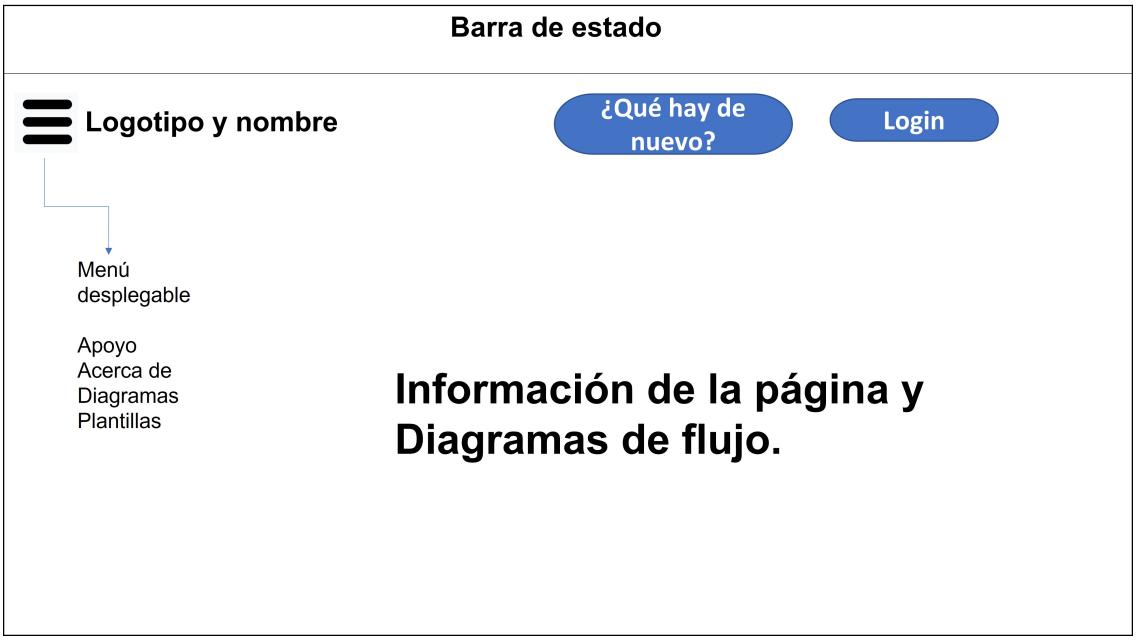


Figura 3.4: Bosquejo de la página principal

La página de trabajo o la página para la creación de diagramas de flujo mostrada en la Figura 3.5, cuenta con el logotipo de la aplicación y un panel de herramientas en donde el usuario tendrá acceso a los símbolos para la creación de diagramas de flujo creados por la norma ISO/ANSI 5807 en 1985, para posteriormente arrastrarlos hacia el área de trabajo. También cuenta con una sección para compartir los diagramas que se están creando, esto en caso de que se realicen trabajos en equipo y que cada uno de los integrantes tenga acceso a los diagramas o en caso de que el docente pueda hacer la revisión de estos diagramas. Por último, se cuenta con una sección en donde se muestra qué personas están trabajando en un mismo diagrama, siendo estas los alumnos y los docentes.

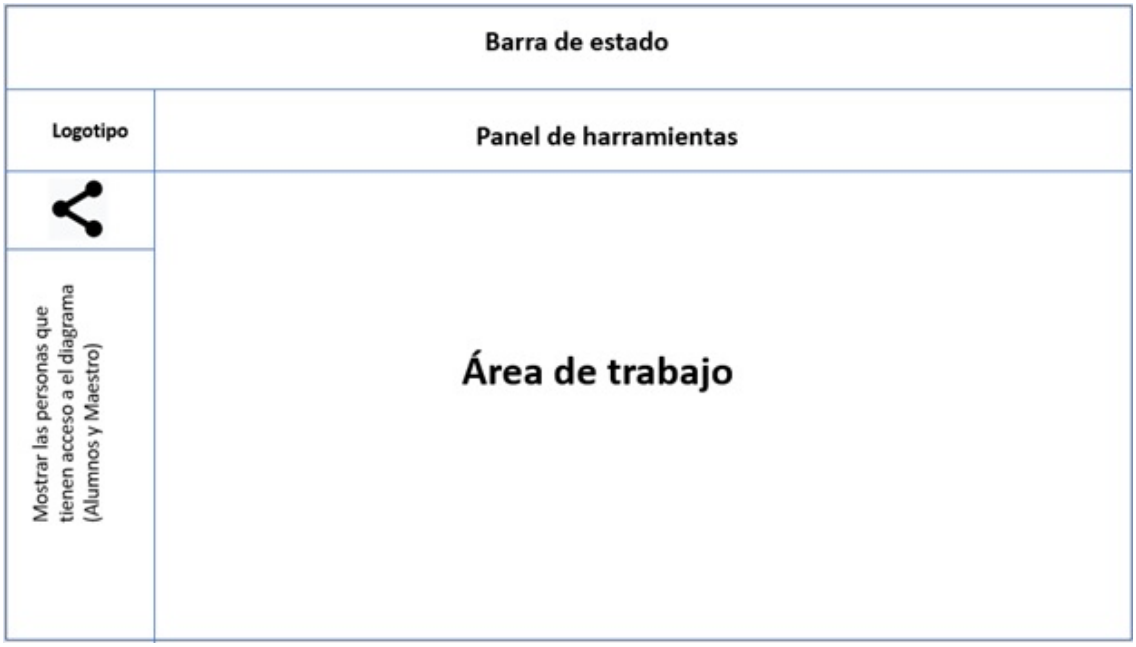


Figura 3.5: Bosquejo de la página de trabajo.

Al momento de acceder a la página de tutoriales consta de una sección de preguntas frecuentes, la cual se muestra en la Figura3.6, un fragmento con un tutorial acerca de la cuenta de usuario, un apartado dedicado al contacto con el soporte de la página, esto en dado caso que falle la aplicación el usuario tenga comunicación con el desarrollador y un bloque que contendrá la documentación del proyecto, así como también el logotipo. Para la parte principal de la página se le muestra a el usuario una serie de vídeos tutoriales, de los cuales van desde cómo crear un diagrama de flujo hasta cómo navegar por la aplicación web.

Figura 3.6: Bosquejo de la página de tutoriales.

En la Figura 3.7 se muestra la descripción donde se presenta en pocas palabras el objetivo del proyecto desarrollado. En la Figura3.8 lo que se pretende es proporcionar algunas plantillas con base en los usuarios para la creación de diagramas de flujo, estas plantillas proporcionan en algunas ocasiones ejemplos de diagramas completos y otros proporcionarán diagramas incompletos para que el usuario los termine.

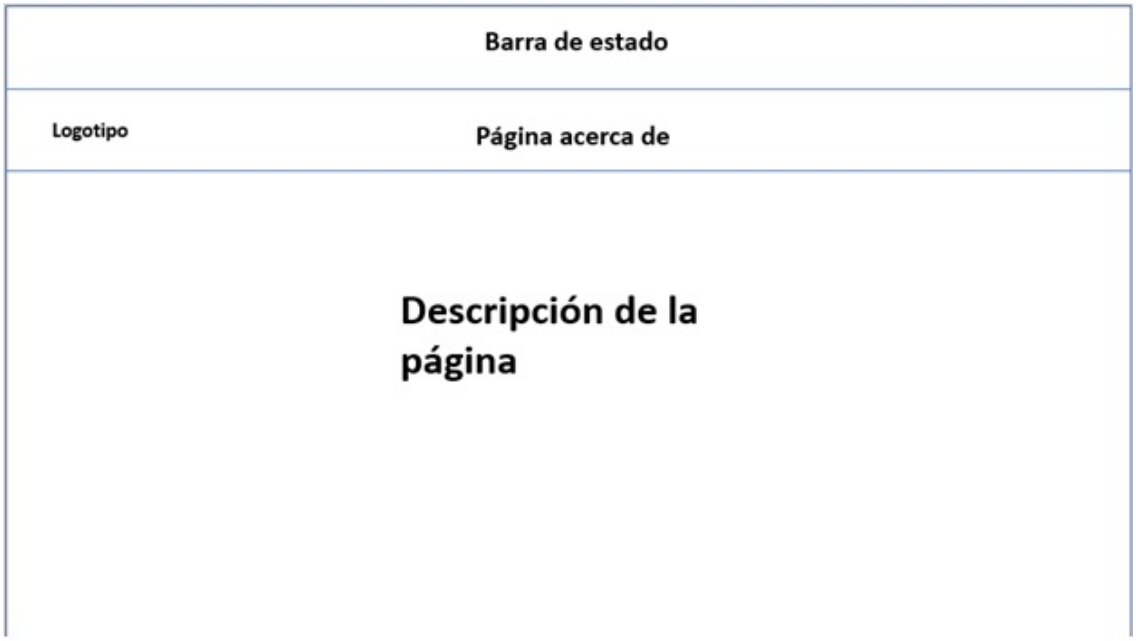


Figura 3.7: Bosquejo de la página de descripción.

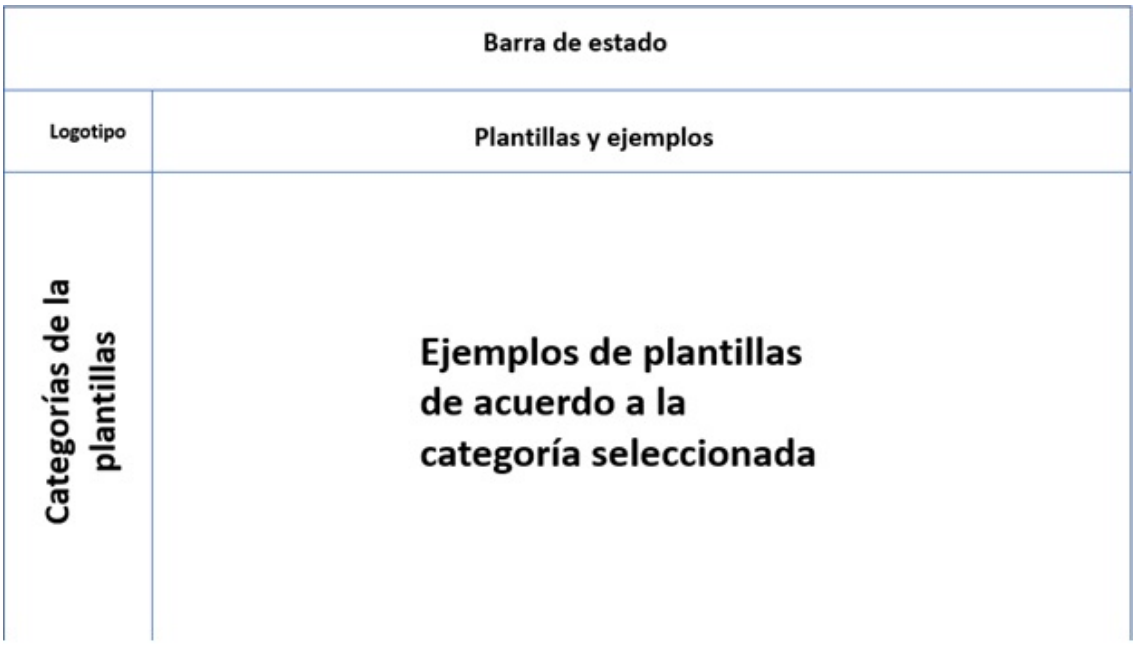


Figura 3.8: Bosquejo de la página de plantillas.

3.5. Construcción del prototipo

Una vez terminada la etapa anterior, se creó una base de datos utilizando MySQL y por último se llevó a cabo el diseño de la interfaz gráfica de usuario para la aplicación web utilizando *HTML Y CSS*.

3.5.1. Símbolos para diagramas de flujo para programas

Se hizo una investigación de los símbolos utilizados por la ISO/ANSI 5807 creada en 1985 para la creación de diagramas flujo. Este estándar es actualizado frecuentemente, siendo la revisión del 2019 los símbolos que se usaron en este proyecto. Una vez obtenida la simbología ISO/ANSI 5807 para diagramas de flujo se procedió a la creación de los símbolos utilizando la aplicación *Inkscape* y se guardó cada uno en formatos .png y .svg, para posteriormente utilizarlos dentro de la aplicación web propuesta.

En la Figura 3.9 se muestra el símbolo que representa la entrada y salida de datos sin importar el medio. Esto significa que el medio de entrada puede ser el teclado, sensor de temperatura, archivo, entre otros, mientras que la salida puede ser a una pantalla o impresora.

Figura 3.9: Entrada y salida de datos.

En la Figura 3.10 se muestra el símbolo que representa cualquier tipo de información que se ingrese manualmente en el momento del procesamiento, ya sea por teclado, configuraciones de switches, botones, lápiz óptico o código de barras.

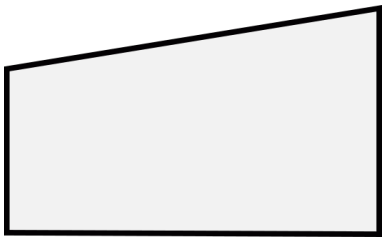


Figura 3.10: Entrada por teclado.

Cualquier tipo de función de procesamiento, ejecuciones de operaciones definidas o grupo de operaciones que resulten en un cambio de valor, puede ser representado por la Figura3.11, también puede representar la forma o la ubicación de la información y también determina la dirección de flujo de datos.

Figura 3.11: Acción o proceso.

El símbolo representado en la Figura 3.12 representa un proceso con nombre que consta de una o más operaciones o pasos a seguir que se especifican en otro lugar, por ejemplo, una subrutina o módulo.

Figura 3.12: Lamada a subrutina.

La Figura 3.13 representa la función del tipo de decisión o conmutación que tiene una

entrada de datos, donde debe haber varias salidas alternativas, de las cuales solo una puede activarse después de la evaluación de las condiciones definidas dentro del símbolo. Los resultados de la evaluación pueden escribirse junto a las líneas que representan las rutas.

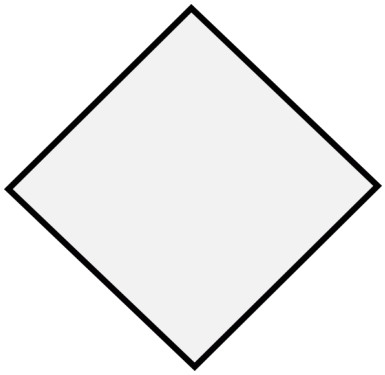


Figura 3.13: Decisión.

El símbolo de la Figura 3.14 representa la entrada o salida desde otra parte del mismo diagrama de flujo, se usa para romper una línea y continuar en otra parte. Los símbolos de conector correspondientes deben contener el mismo identificador único.

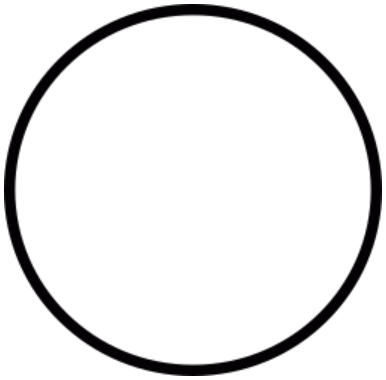


Figura 3.14: Conector.

En la Figura 3.15 símbolo representa el inicio y fin del diagrama de flujo, así como también el origen y destino de los datos.

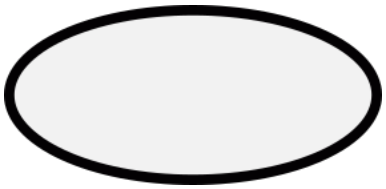


Figura 3.15: Inicio/Fin del diagrama.

El conector de página representa la continuación del diagrama de flujo dentro de la misma página, es utilizado cuando existe una referencia cruzada y en esa misma referencia existe un enlace de algún proceso diseñado en otra página y es representado en la Figura 3.16.

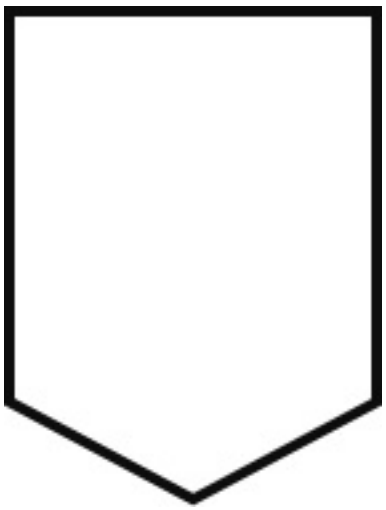


Figura 3.16: Conector de página.

La forma en como se deben cruzar las líneas al momento de crear un diagrama de flujo y la dirección en la que fluye la información se ven reflejadas en la Figura3.17 y en la Figura 3.18, respectivamente.

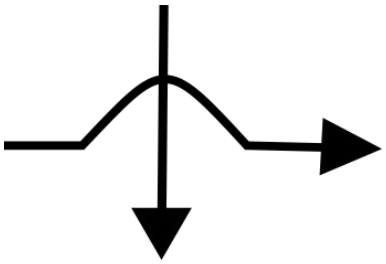


Figura 3.17: Cruce de líneas dentro del diagrama de flujo.

Figura 3.18: Flujo de la información dentro del diagrama de flujo.

3.5.2. Construcción de la base de datos

Se procedió a la creación de la base de datos implementando el modelo relacional y realizando la *Forward Engineer* para la creación de la base de datos en el sistema de gestión de bases de datos MySQL, tal y como se muestra en la Figura 3.19.

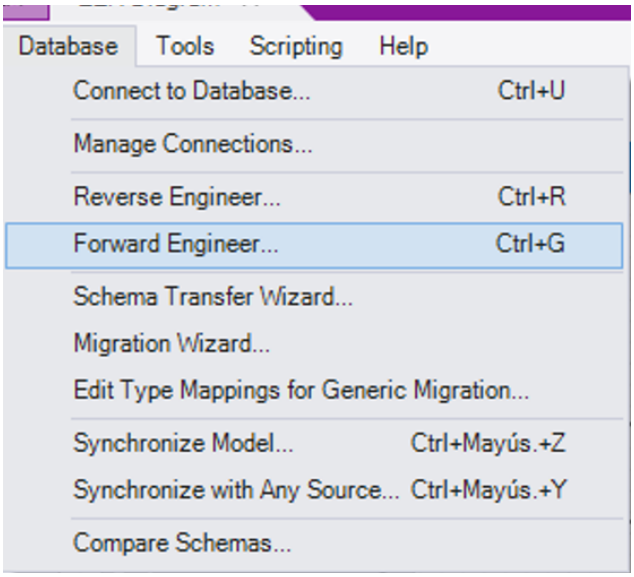


Figura 3.19: Ingenieria hacia adelante.

En la Figura 3.20 se presenta la primera ventana donde se muestran los parámetros de conexión con el DBMS. Es aquí donde se selecciona la instancia en la que se hará la conexión, el método de conexión que se utilizó en este caso es TCP/IP, además de otros parámetros como el *hostname*, nombre de usuario, contraseña el puerto a utilizar y en dado caso que exista un esquema predeterminado utilizarlo o de lo contrario se puede seleccionar un esquema más adelante.

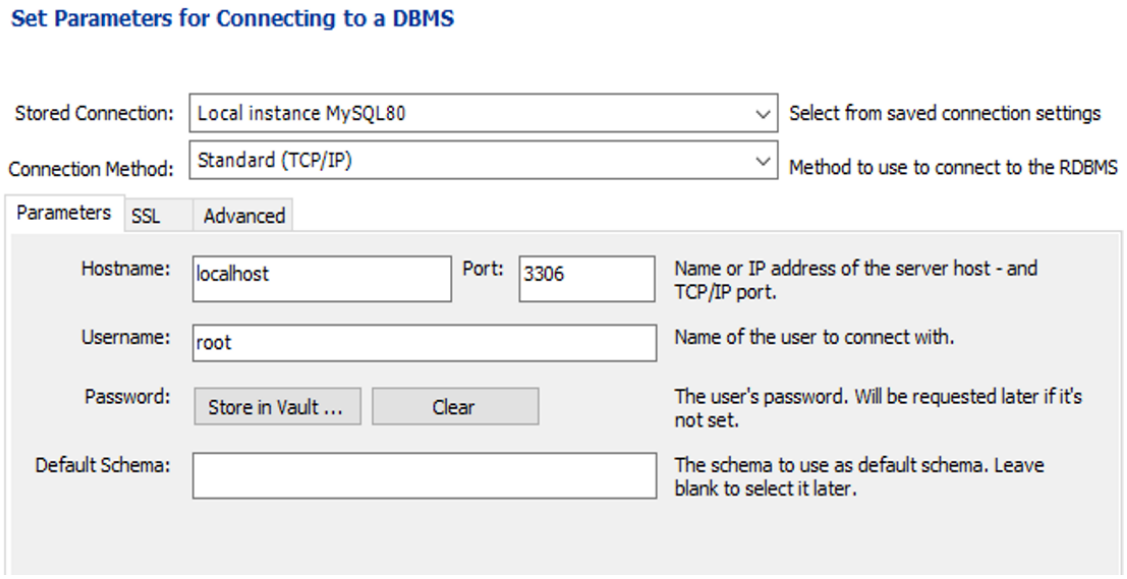


Figura 3.20: Parámetros de conexión.

Como paso siguiente, en la Figura 3.21 se muestran la opciones de la base de datos creada, ya sean tablas u objetos que se requieran agregar y la generación de código que se realizó.

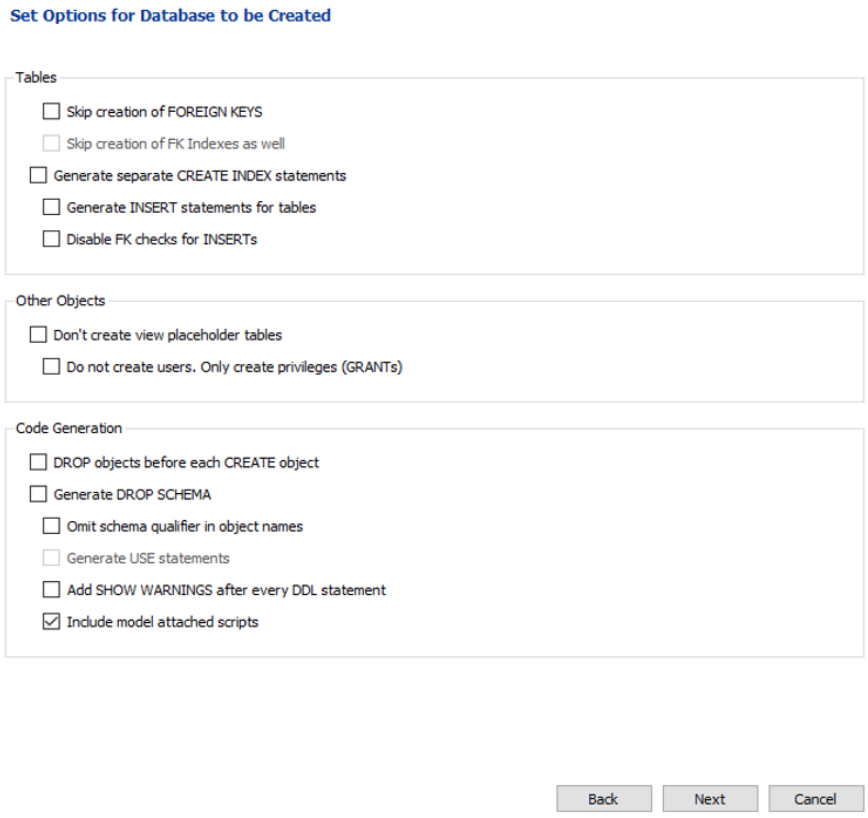


Figura 3.21: Opciones para la creación de la base de datos.

En la Figura 3.22 aparecen los objetos que se necesitan para la ingeniería hacia adelante que nos proporciona el DBMS, es decir son los objetos de la base de datos que serán exportados. Para este proyecto solo fue necesario seleccionar la primer opción, es decir solo se exportaron las tablas previamente creadas en el modelo relacional.

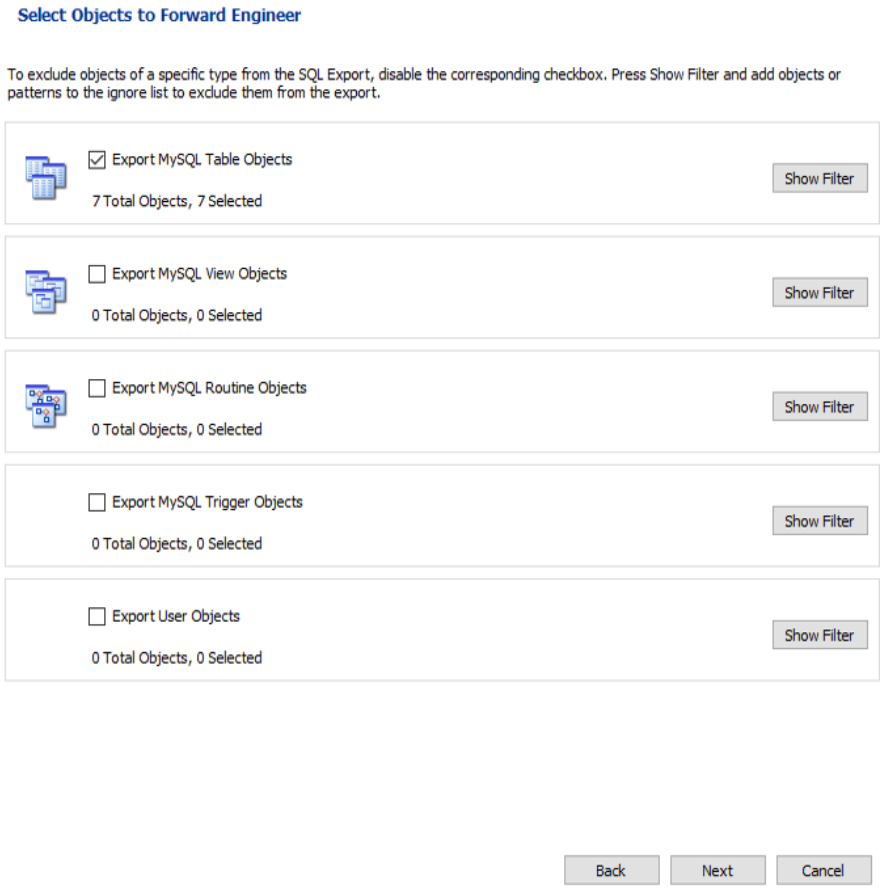


Figura 3.22: Selección de objetos a exportar.

Una vez seleccionados los objetos que se van a exportar, se selecciono el botón siguiente. En la siguiente página mostrada en la Figura 3.23 se puede ver el código SQL de la base de datos, una vez que se obtuvo el código se guardó con el nombre de *mydb*, tal como se muestra en la Figura 3.25.

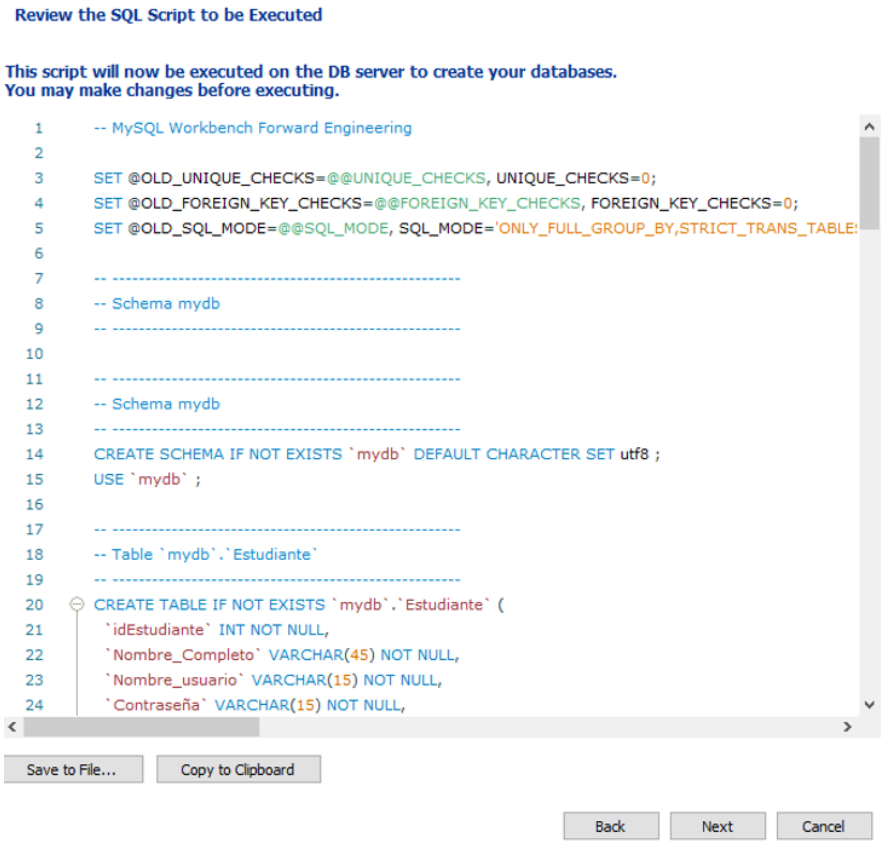


Figura 3.23: Generación del código SQL.

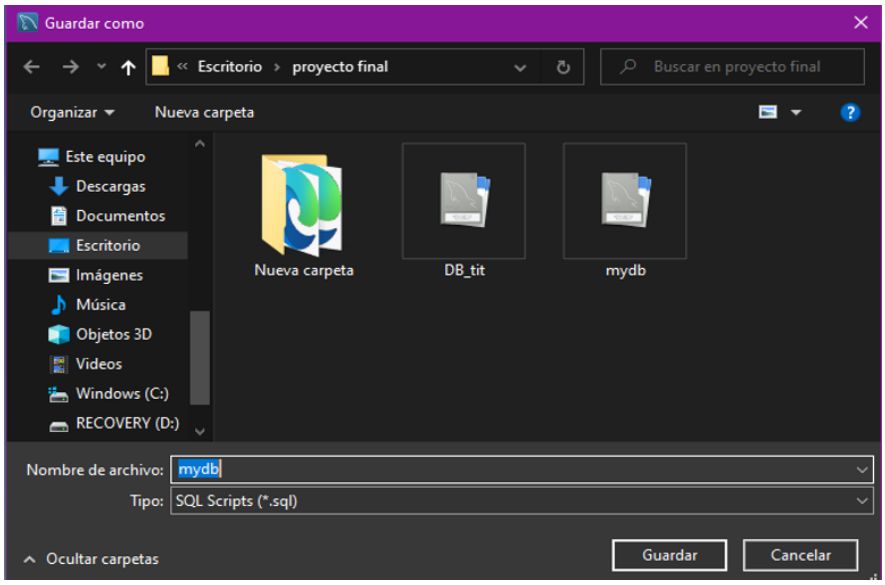


Figura 3.24: Almacenamiento del código SQL.

Una vez guardado el código se avanzó a la siguiente página, donde se ve el progreso de la ingeniería hacia adelante, es decir, que se establezca la conexión, que se ejecute el *Script*, que se lean los cambios hechos por el servidor, y por último que el estado de sincronización se guarde. Cuando estas opciones estén verificadas aparecerá el mensaje *Forward Engineer Finished Successfully*, el cual indica que la base de datos se ha creado.

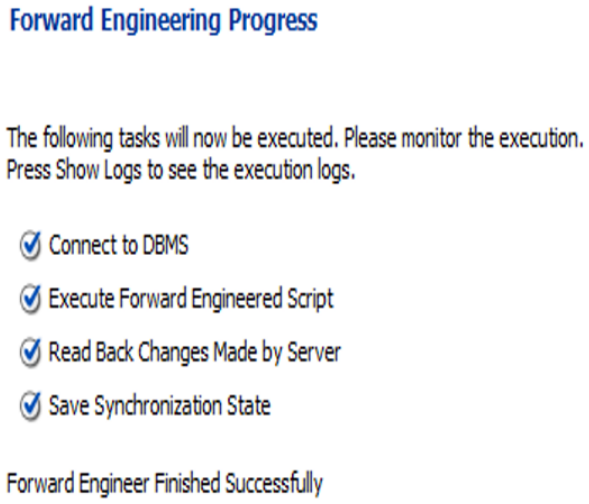


Figura 3.25: Ingeniería hacia adelante completada.

Una vez terminada la ingeniería hacia adelante, se accedió a el servidor de MySQL para comprobar que la base de datos se guardó correctamente, además de agregar algunos datos a las tablas de la base de datos. Con el comando *SHOW DATABASES;* se muestran las bases de datos dentro del servidor de MySQL. En la Figura 3.26 se muestra la base de datos utilizada.

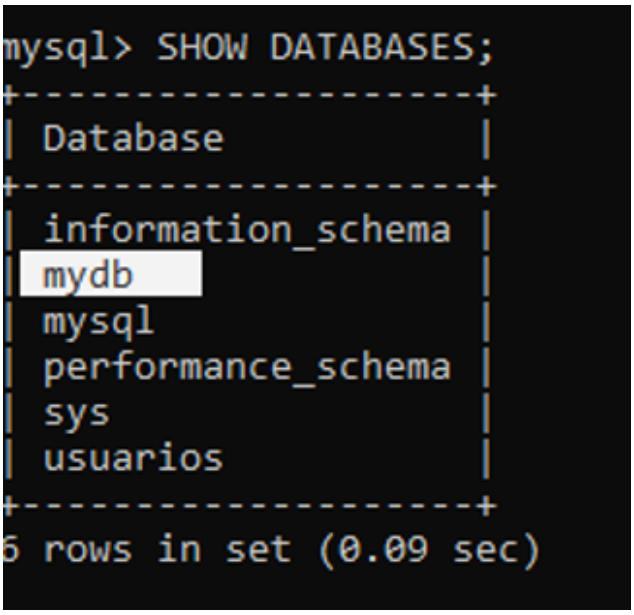


Figura 3.26: Base de datos utilizada en el proyecto.

Para hacer el cambio a la base de datos *mydb*, se utilizó el comando *USE mydb;*, luego para mostrar las tablas de la base de datos se utilizó el comando *SHOW TABLES;*, tal como se muestra en la Figura 3.27.

```
mysql> use mydb;
Database changed
mysql> show tables;
+------------------+
| Tables_in_mydb   |
+------------------+
| diagrama         |
| estudiante       |
| estudiante_grupo |
| maestro          |
| maestro_grupo    |
| materia          |
| tipo_usuario     |
+------------------+
7 rows in set (0.22 sec)
```

Figura 3.27: Tablas de la base de datos.

Utilizando el comando *INSERT INTO nombre de la tabla VALUES (columna 1, columna 2, columna 3);* se insertan los datos en las en cada una de las tablas. Por ejemplo, para la tablas estudiante se utilizó de la siguiente manera: *INSERT INTO estudiante VALUES(1, Sandra Luz Yañez Rivera, S03L04Y93, SLYR121052)* y *INSERT INTO estudiante VALUES(2, Javier López Barraza, J06L18B93, JLB123456)*. Para verificar que los datos fueron insertados correctamente se utilizó el comando *SELECT * FROM estudiante;* tal como se muestra en la Figura 3.28. El mismo procedimiento se sigue para todas las tablas.

```
mysql> select * from estudiante;
+-------------+-------------------------+----------------+-------------+
| idEstudiante | Nombre_Completo         | Nombre_usuario | Contraseña  |
+-------------+-------------------------+----------------+-------------+
|           1 | Sandra Luz Yañez Rivera | S03L04Y93      | SLYR121052  |
|           2 | Javier Lopez Barraza    | J06L18B93      | JLB123456   |
+-------------+-------------------------+----------------+-------------+
2 rows in set (0.19 sec)
```

Figura 3.28: Consulta de los datos insertados.

3.5.3. Construcción del prototipo web

Para la parte de la aplicación web, se utilizó *Atom* como entorno de desarrollo. La aplicación consta de una sección de Login, una página de inicio, un módulo para la creación de los diagramas de flujo,un módulo para crear código en Rust, una sección de tutoriales, así como también una página de ejemplos de diagramas ya hechos. Para evitar redundancia en la información proporcionada para el usuario se tomó la decisión de eliminar el apartado de Acerca de mencionada en el bosquejo presentado en la sección 3.4 de este mismo capítulo.

La página mostrada en la Figura 3.29, muestra la interfaz para hacer *Login*, la página se muestra sencilla ya que solo contiene las cajas para poner el nombre de usuario y contraseña, donde se utilizarán los datos guardados en la base de datos del prototipo y un botón que nos lleva a la página de crear diagramas.

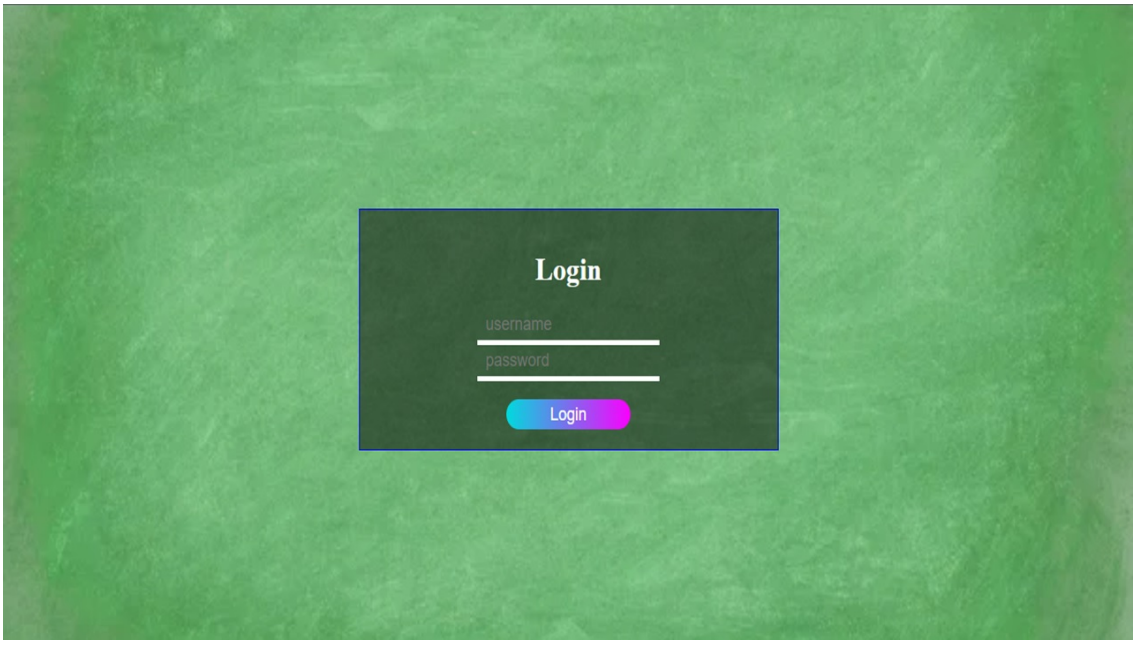


Figura 3.29: Página Login.

La página principal que se muestra en la Figura3.30 cuenta con una barra superior, don-

de se pueden apreciar las diferentes páginas con las que cuenta el prototipo y que fueron mencionadas en el párrafo anterior, así como también cuenta con la descripción de la aplicación. También cuenta con una sección de novedades, donde se pondrán a disposición las actualizaciones de la aplicación.

Figura 3.30: Página principal de la aplicación.

En la Figura 3.31 se muestra la página donde el usuario creará sus diagramas de flujo, una sección en la parte izquierda, donde el usuario podrá visualizar los símbolos disponibles para su diagrama proporcionados por el estándar ISO/ANSI 5807. Es en esta parte del proyecto donde se utilizó la biblioteca *Drawflow*, la cual permite que el usuario pueda arrastrar los símbolos a el área de trabajo, estos símbolos aparecen en la parte izquierda de la página cada uno con su respectivo nombre e icono. Además cuenta con un botón que limpia la pantalla el cual lleva el nombre de borrar y uno mas con el el nombre de exportar que contiene el JSON que se genera al crear el diagrama de flujo el cual será almacenado en la computadora como un archivo .js. En la parte inferior derecha aparecen los botones de zoom para alejar y

acercar la vista del diagrama.

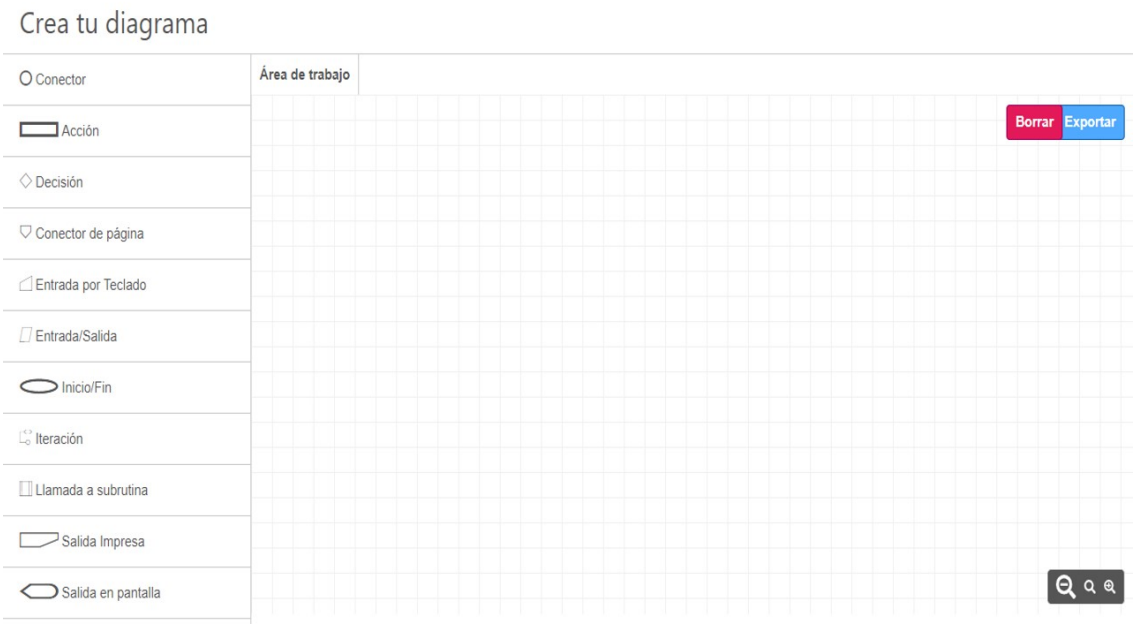


Figura 3.31: Página creación de diagramas.

El código presentado en el Listado 3.1 muestra las líneas que permiten cargar los iconos de la simbología para los diagramas. Estos se establecen en la sección izquierda de la página donde se pueden apreciar junto con el nombre de cada símbolo, para posteriormente ser arrastrados y soltados en el área de trabajo. Una vez que se sueltan en el área de trabajo, se carga la imagen correspondiente y con los puntos de control respectivos. Los puntos de control permiten conectarse con otros símbolos para así formar un diagrama.

```
1 <!DOCTYPE html>
2 [...]
3       <div class="drag-drawflow" draggable="true" ondragstart="drag(
    event)" data-node="Conector">
4         <i class="icon-Conector"></i>
5         <span>Conector</span>
6       </div>
```

```
7       <div class="drag-drawflow" draggable="true" ondragstart="drag(
    event)" data-node="AccionProceso">
8         <i class="icon-AccionProceso"></i>
9         <span>Acción</span>
10      </div>
11  [...]
12  </html>
```

Listado 3.1: Código de la página de crear diagramas.

A consideración del asesor y autor de este documento, se tomo la decisión de crear un módulo de codificación para el lenguaje Rust. Esto con el fin de que los alumnos lo conozcan ya que tiene el potencial de ser el sucesor del lenguaje de programación C. En la Figura 3.32 se muestra un ejemplo de este módulo en el que se imprime una lista de números almacenados en un vector. Para la creación de este módulo se utilizó la herramienta CodeMirror de la manera que se indica en el tutorial How to Make a Code Editor with Codemirror 6. El código mostrado por defecto fue obtenido de la siguiente página web Caso de prueba: Impresión de una lista.

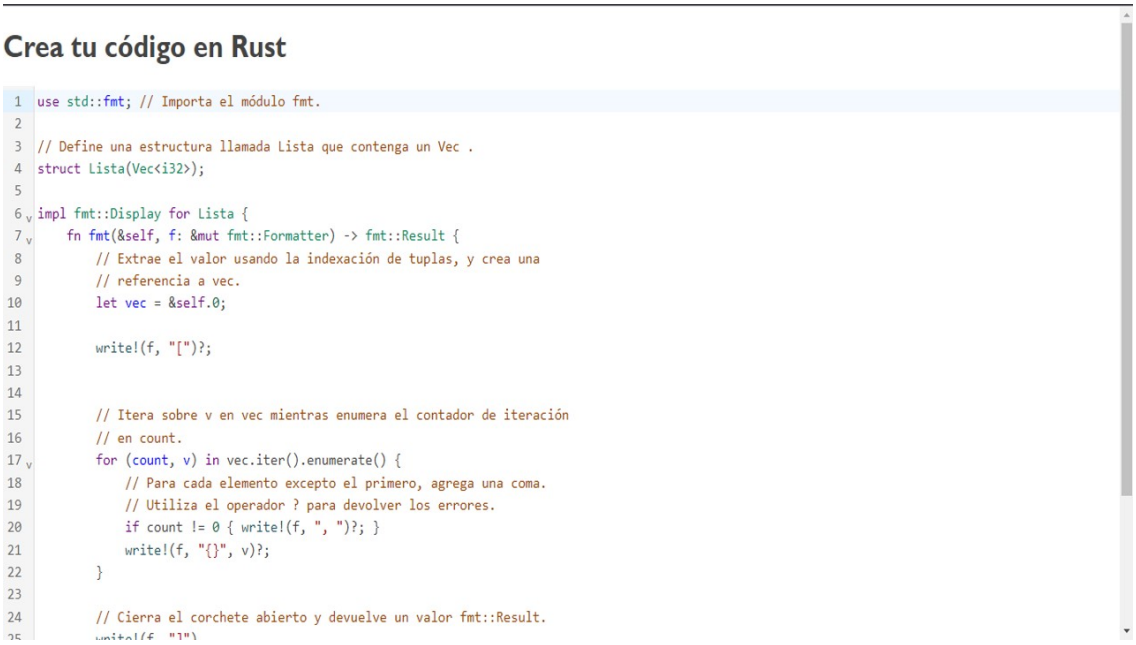


Figura 3.32: Página editor de código Rust.

En el Listado mostrado en 3.2 se muestra el contenido del archivo editor.js, el cual genera el editor de código principal, es aquí donde más tarde se agrega el codigo Rust mostrado en la Figura3.32.

```
import {EditorState, EditorView, basicSetup} from "@codemirror/
    basic-setup"
import {rust} from "@codemirror/lang-rust"

let editor = new EditorView({
  state: EditorState.create({
    extensions: [basicSetup, rust()]
  }),
  parent: document.body
})
```

Listado 3.2: Código que genera el editor principal.

La página de tutoriales contiene los espacios donde se pretende se muestren diferentes vídeos, los cuales servirán de apoyo a los usuarios para el manejo de la aplicación. Tal como se muestra en la Figura 3.33.

Figura 3.33: Página de tutoriales.

La Figura 3.34, muestra la página de plantillas y ejemplos, es aquí donde el usuario podrá tener acceso a algunos diagramas ya realizados que le servirán de guía al momento de realizar su propio diagrama, también podrán ser de ayuda para los docentes, al momento de dar ejemplos en clases. Los ejemplos mostrados en la página de plantillas fueron diseñados por el autor de este proyecto.

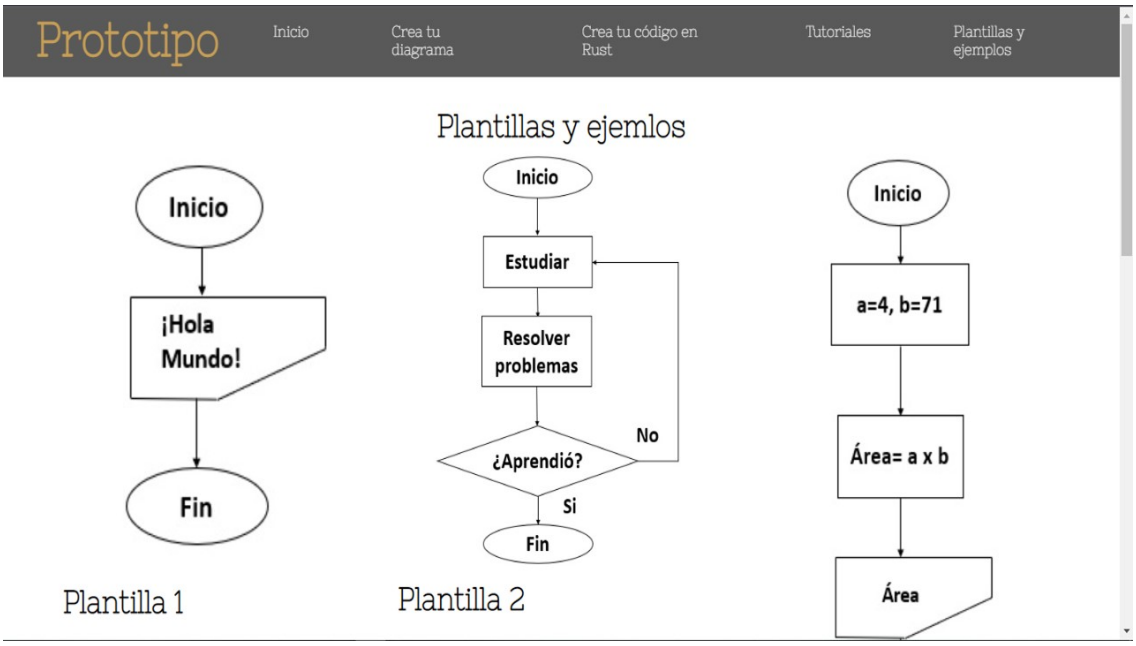


Figura 3.34: Página de Plantillas y ejemplos.

3.6. Evaluaciones del usuario

En esta etapa se puso a prueba el prototipo del sistema propuesto para determinar las fortalezas y debilidades en la elaboración de diagramas de flujo. Además, se recopilaron anotaciones y sugerencias realizadas por posibles usuarios cómo lo son estudiantes de las materias relacionadas a programación y algunos docentes.

El prototipo fue evaluado por cinco estudiantes del programa de Ingeniería en Sistemas Computacionales. Los aspectos a evaluar fueron relacionados a el diseño del prototipo tomando en cuenta aspectos como los colores, la tipografía, la estructura de las páginas, así cómo también se evaluó la herramienta para la creación de diagramas y el módulo de codificación en Rust.

Como resultado se obtuvo que dos de estos estudiantes quedaron satisfechos con el diseño

actual del prototipo y como está estructurado, otro estudiante sugirió hacer cambio en los colores y mencionó que para la sección de tutoriales le gustaría ver un tutorial acerca de cómo funciona el prototipo, otro estudiante comentó que se realizara un tutorial acerca de cómo es que funciona el módulo de creación de diagramas y uno más que esté dedicado al módulo de codificación en Rust y que también explique porqué se utilizó este lenguaje y no otro más común. El último estudiante sugirió darle el mismo diseño a el módulo de creación de diagramas y a el módulo de codificación que las otras páginas, ya que consideró que al entrar a estos módulos parece que salimos a otro entorno diferente.

3.7. Cambios y sugerencias

En esta fase se atendieron las anotaciones y sugerencias de la fase anterior. Esto con el fin de satisfacer el objetivo principal de la propuesta, llegando así al prototipo final. De acuerdo a la evaluación de los cinco estudiantes se tomaron en cuenta los siguientes cambios y sugerencias:

- Como sugerencia se hizo que para los módulos de codificación y creación de diagramas tuvieran los mismos colores que las otras páginas.
- Atendiendo lo mencionado en el punto anterior se modificó el módulo de diagramas y codificación para obtuvieran la misma apariencia que las otras páginas.
- Otra sugerencia fue el cambio de colores, pero debido a que la mayoría de los evaluadores concordaron con que los colores están adecuados no se realizó algún cambio.
- Se propusieron ideas de que les gustaría ver en la sección de vídeos tutoriales, los cuáles quedarán como trabajo a futuro por cuestión de tiempo. Por lo que se hizo e cambio mostrado en la Figura 3.35.

Figura 3.35: Cambios sugeridos por el usuario en la página de tutoriales.

3.8. Implementación

En esta última fase se hizo la implementación parcial ya que el prototipo no estará disponible inmediatamente para su uso. Al ser un proyecto a desarrollarse por varias personas en diferentes tiempos la implementación queda postergada como trabajo a futuro.

Capítulo 4
Resultados y Discusiones

En este capítulo se muestran los resultados obtenidos en la evaluación del prototipo. En la sección 4.1 se presentan las pruebas realizadas con los distintos navegadores. En la sección 4.2 se la interpretación y discusión de los datos obtenidos.

4.1. Resultados

En esta sección se muestran los resultados del prototipo web por medio de un cuadro comparativo. El prototipo fue evaluado en diferentes navegadores y dos dispositivos móviles, esto con el fin de evaluar el funcionamiento de la herramienta para crear diagramas, el módulo de codificación y el diseño en cada uno de ellos. Los resultados que se obtuvieron se muestran en la Tabla 4.1. Para la evaluación del prototipo se utilizó la versión mas reciente de cada navegador y en una computadora con las siguientes características: memoria RAM de 8.00 GB, sistema operativo Windows de 64 bits y con procesador x64, un teléfono celular con Android en su versión más reciente y una tableta con IOS también en su versión más reciente.

Tabla 4.1: Evaluación del prototipo en navegadores.

Navegador	Módulo de diagramas	Módulo de codificación	Diseño
	✓	✓	✓
	✓	✓	✓
	×	×	×
	✓	✓	✓
	✓	✓	✓
	✓	✓	✓

En la Figura 4.1se muestra la evaluación en el navegador Chrome y en la Figura 4.2 la del navegador Brave las páginas mantienen su diseño y estructura, la pruebas realizadas a el módulo de crear diagramas arrojan que esta parte del producto funciona satisfactoriamente. El módulo de codificación por el momento también cumple con su funcionalidad sin ningún problema.

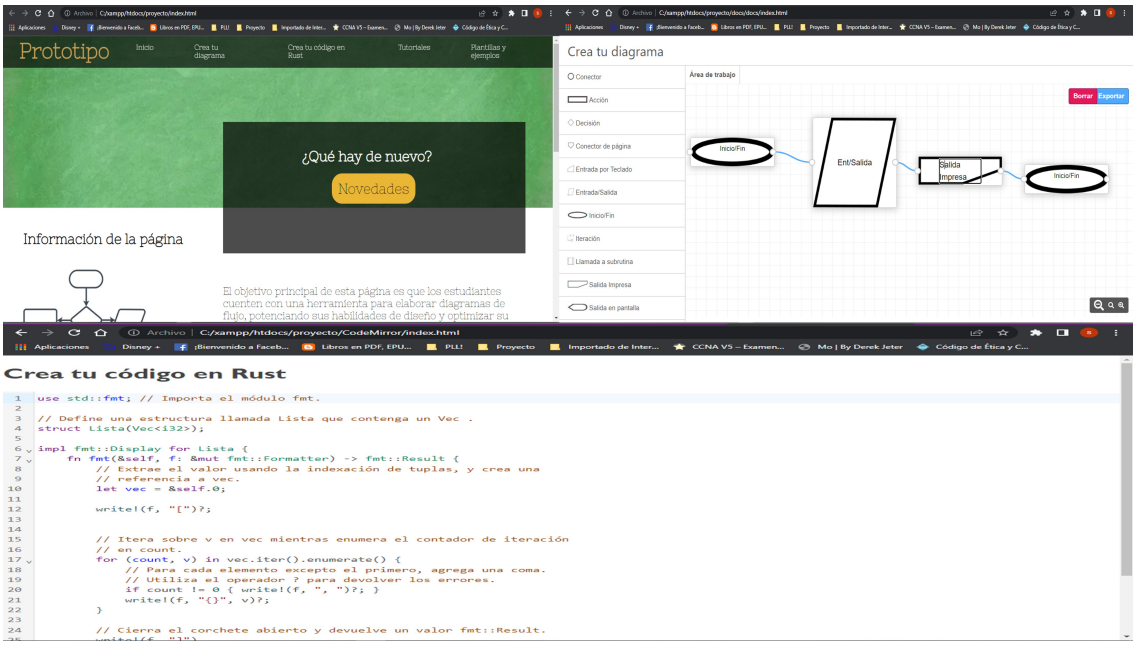


Figura 4.1: Vistas del prototipo en navegador Chrome.

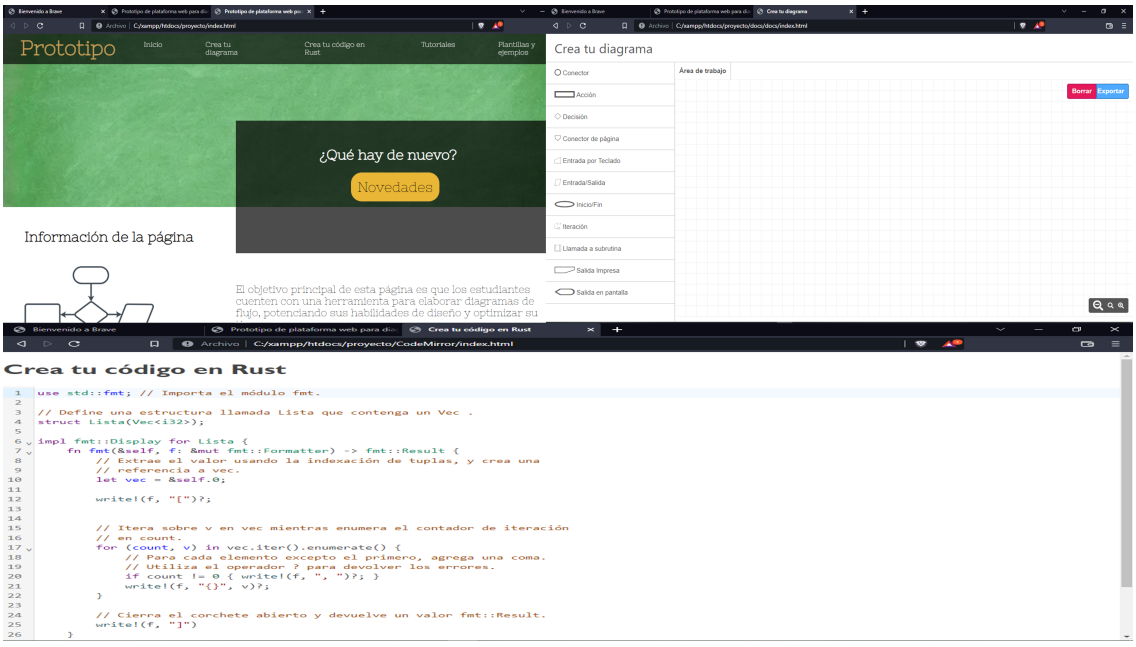


Figura 4.2: Vistas del prototipo en navegador Brave.

Para el navegador Firefox mostrado en la Figura 4.3 se pudo notar un ligero cambio en la página de tutoriales en el diseño de las cajas de los vídeos, mientras que en el navegador Edge mostrado en la Figura 4.4, la estructura y diseño de las páginas no sufre ningún cambio. El módulo de diagramas cumple con sus funciones al igual que el módulo de codificación en ambos navegadores.

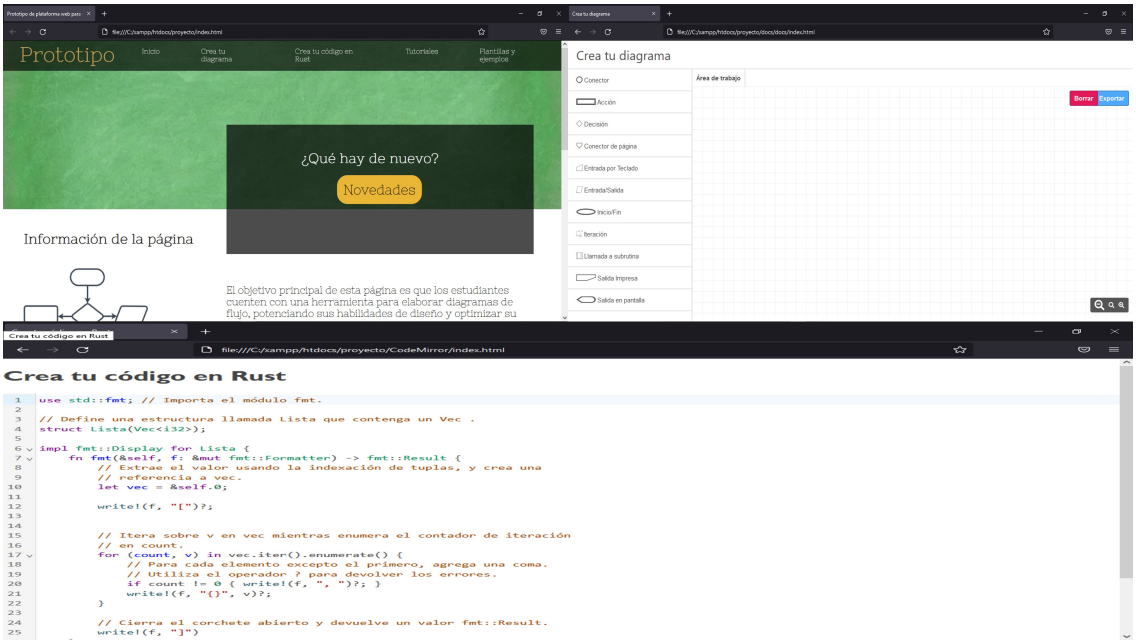


Figura 4.3: Vistas del prototipo en navegador Firefox.

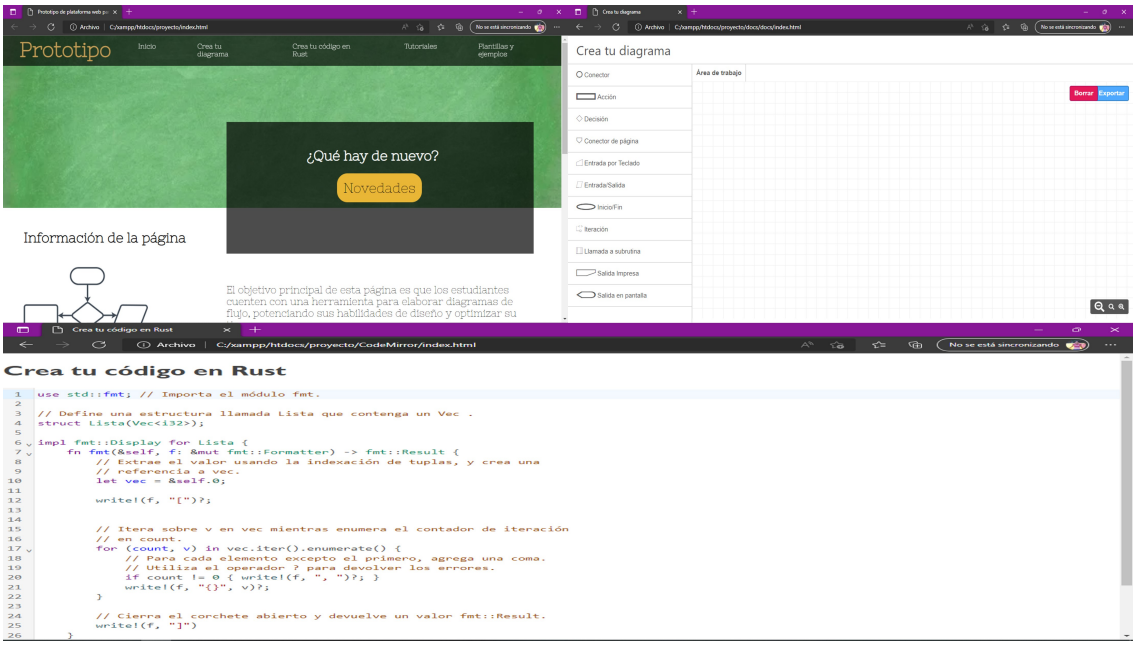


Figura 4.4: Vistas del prototipo en navegador Edge.

La evaluación en el navegador Ópera también fue exitosa de acuerdo a los criterios mostrados en el cuadro comparativo al principio de esta sección. En la Figura 4.5 se muestran las vistas de la página principal, el módulo de creación de diagramas y módulo de codificación de este navegador.

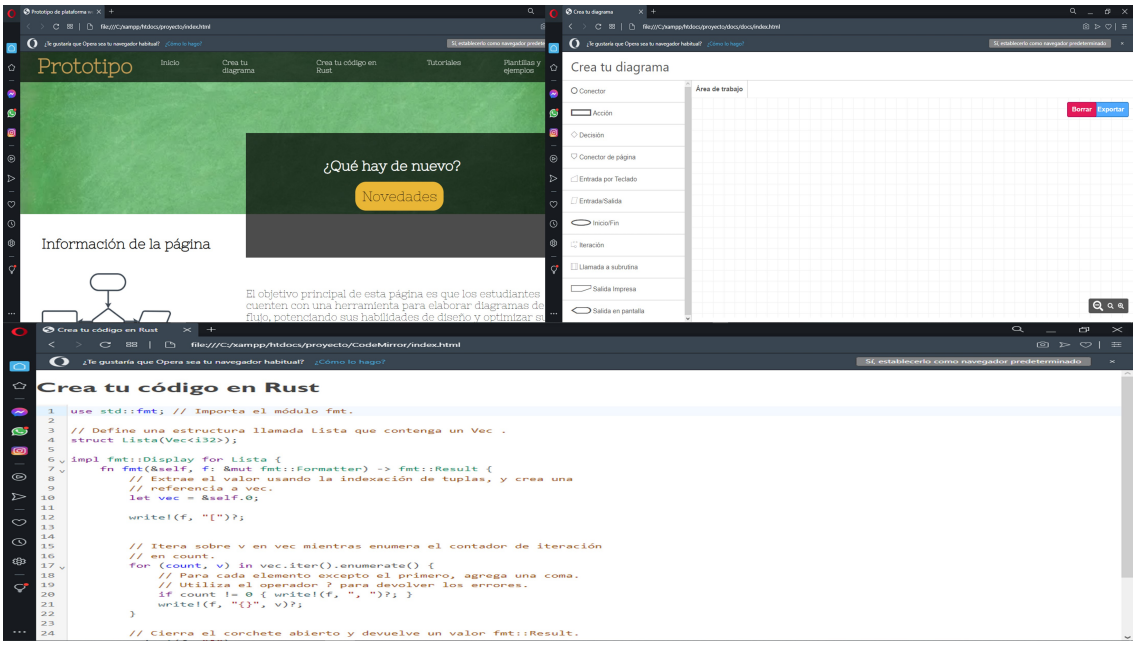


Figura 4.5: Vistas del prototipo en navegador Ópera.

El prototipo en navegador Safari mostrado en la Figura 4.6 versión Windows sufre afectaciones en diseño, no permite navegar entre páginas ya que las secciones no aparecen por lo tanto en esta versión de Safari el prototipo no es aceptable, tal como se muestra en la Figura 4.6. También se trató de probar el prototipo en una tableta con IOS y en un teléfono celular con Android, sin embargo, no se obtuvieron buenos resultados ya que no fue posible visualizar la aplicación en una navegador. Los resultados se muestran en la Figura 4.7.

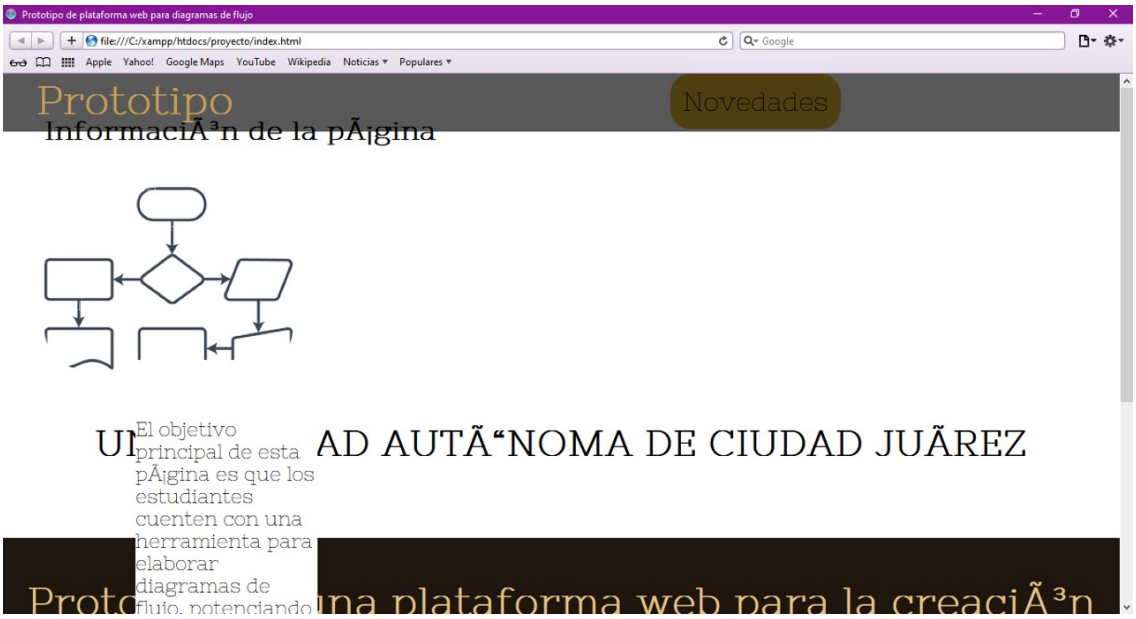


Figura 4.6: Vistas del prototipo en navegador Safari.

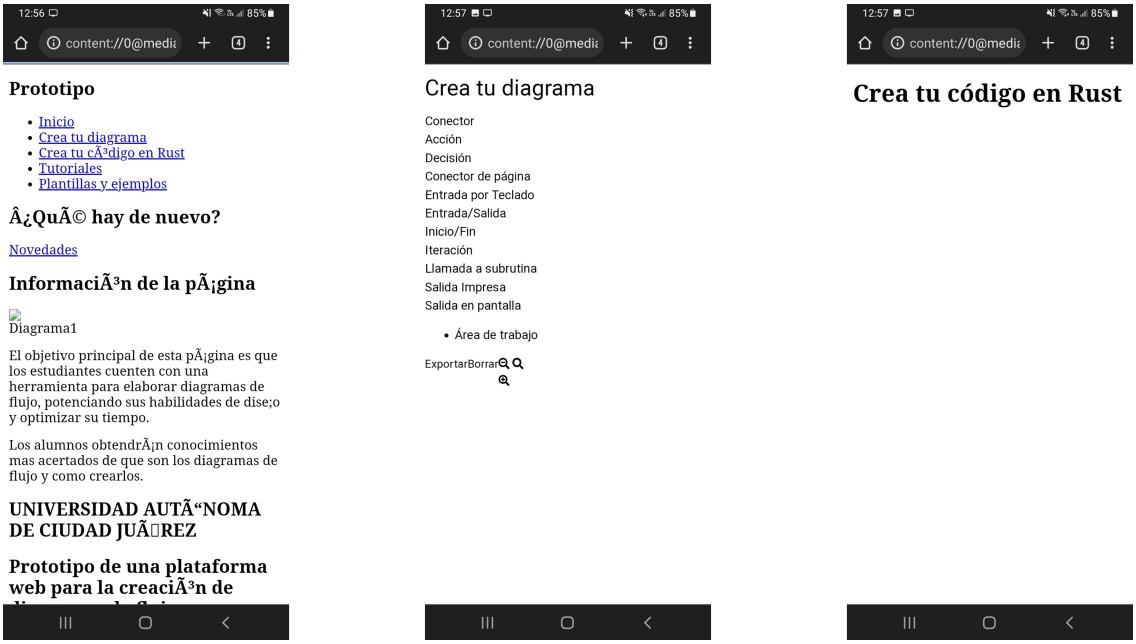


Figura 4.7: Vistas del prototipo en dispositivos móviles.

4.2. Discusión

Dentro de las aplicaciones exploradas en la sección 1.1 no se encontraron herramientas que implementen completamente los símbolos establecidos por el estándar ISO/ANSI 5807, contenga reconocimiento de voz y además que promueva en los estudiantes las habilidades de pensamiento rápido y lógico. El prototipo presentado cuenta con un módulo de creación de diagramas de flujo que promueve el uso de estas habilidades, así como un módulo de codificación en el lenguaje de programación Rust. Esto último como un extra agregado al final del proyecto teniendo en cuenta que este lenguaje de programación tiene el potencial de sustituir al lenguaje C y que a consideración del autor y asesor de este documento se llegó a la resolución de que los estudiantes deberían de empezar a familiarizarse con dicho lenguaje.

Se obtuvo como resultado de que el prototipo es funcional en cinco de los seis navegadores evaluados, una tableta con IOS y un teléfono con Android. Obteniendo que en el navegador *Safari* el prototipo no se pudo visualizar de la manera correcta y no permite la navegación entre las distintas páginas, cabe mencionar que para este navegador se utilizó la versión para *Windows*, al igual que en el navegador *Safari*, en los dispositivos móviles no fue posible visualizar el prototipo. En cuanto a los cinco restantes, el prototipo se muestra tal y como fue diseñado, no se muestran afectaciones al interactuar con las distintas secciones. En cuanto al módulo para la creación de diagramas se evaluó que las funciones de arrastrar y soltar los símbolos al área de trabajo funcionarán adecuadamente y que el módulo de codificación muestre el código de ejemplo propuesto por el autor del prototipo para que en un futuro el estudiante pueda crear código desde cero en el lenguaje de programación Rust.

Con los resultados obtenidos se obtuvo que el prototipo utiliza el estándar ISO/ANSI 5807 para la creación de diagramas de flujo, con lo cual se resuelve el problema definido en el capítulo 1, en el cual se menciona que ninguna herramienta explorada lo implementa por completo.

Capítulo 5
Conclusiones

En este capítulo se describe cómo es que se logró el cumplimiento del objetivo general y los objetivos específicos establecidos para la realización del prototipo, así como también se presentan futuras recomendaciones con las cuales se podrá continuar con el desarrollo y evolución del prototipo.

5.1. Con respecto al objetivo de la investigación

El objetivo general de este proyecto fue Desarrollar el prototipo de una plataforma web para la elaboración de diagramas de flujo utilizando la simbología del estándar de la ISO/ANSI 5807 y para la escritura de código, esto con el fin de promover el pensamiento rápido y lógico.. De los resultados descritos en el capítulo anterior y este objetivo general, se deducen tres conclusiones. Primero, que el prototipo permite el diseño de diagramas de flujo usando símbolos estandarizados desde cualquier navegador web. Segundo, al usar un estándar internacional para los diagramas, éstos serán comprendidos fácilmente por personas de diferentes lugares. Tercero, CodeMirror facilita la escritura de código en el lenguaje de programación Rust ya que contiene resaltador de sintaxis y autocompletado. Estas tres conclusiones sientan las bases para que, en un futuro, se diseñen estrategias de enseñanza en donde el estudiante pueda traducir los diagramas de flujo a un programa computacional.

Al ser un prototipo, se consideró que no era apropiado realizar pruebas para verificar si se lograba promover el pensamiento rápido y lógico. Sin embargo, esto se deja para cuando se tenga el sistema completamente funcional. Por otra parte, lo relacionado a la voz que pudiera ser de mucha utilidad a personas con alguna dificultad en sus manos o vista, se deja en este documento una lista de herramientas que se pudieran usar para lograr tal propósito.

5.2. Recomendaciones para futuras investigaciones

Con el fin de seguir desarrollando el prototipo se proponen las siguientes actividades:

- Creación de diagramas de flujo por medio del reconocimiento de voz.
- El desarrollo del los vídeos tutoriales.
- El diseño de las plantillas y ejemplos.
- La conexión de la base de datos a la página de *Login*.
- La creación de diagramas compartidos.
- La implementación y mantenimiento del prototipo.

Bibliografía

[1] (2021), "Como crear un diagrama de flujo en PowerPoint." [Internet]. Disponible en: https://bit.ly/3r6XFUk.

[2] (2021), "How to create your flowchart online quickly and easily." [Internet]. Disponible en: https://bit.ly/3cPAqcx.

[3] N. P. Loyarte Horacio, "Desarrollo e implementación de un intérprete de pseudocódigo para la enseñanza de algorítmica computacional," Agosto 2006.

[4] (2021, Ag. 20), "£qué es un diagrama de flujo?." [Internet]. Disponible en https://bit.ly/3cL0G7G.

[5] (2021), "Simbologia de un diagrama de flujo." [Internet]. Disponible en: https://bit.ly/3r4u9yH.

[6] A. del Prado and N. Lamas, "Revista Electronica Iberoamericana de Educacion en Ciencias y Tecnologia," vol. 5, pp. 102–113, 2014.

[7] F. H. Evangelista and J. P. Novara, "Interprete para probar un programa escrito en pseudocodigo," vol. 17, no. 1, pp. 101–109, 2014.

[8] "Draw.io." [Internet]. Disponible en: https://drawio-app.com/.

[9] B. Dilts and K. Sun (2021), "Lucidchart." [Internet]. Disponible en: https://bit.ly/38YYf0c.

[10] J. A. Pimentel, S. N. García, R. S. González, and G. A. López, "Software para la enseñanza-aprendizaje de algoritmos estructurados," *Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología*, pp. p. 23–33, dic. 2012.

[11] (2001-2019), "Procesos educativos." [Internet]. Disponible en: `https://bit.ly/2NByKux`.

[12] M. Marqués, *Bases de datos*. Castellon de la Plana: Universitat Jaume I. Servei de Comunicacio i Publicacions, Primera edición ed., 2011.

[13] A. M. Rioja, *Bases de datos. Diseño y gestión*. Madrid, España: SINTESIS,S.A.

[14] C. M. Ricardo, *Bases de datos*. México: Jones and Bartlet Publichers Inc., Primera Edicion ed., 2009.

[15] J. S. Castejón Garrido, "Arquitectura y diseño de sistemas web modernos," *Revista de ingeniería informática del CIIRM*, Jan. 2004. [Internet]. Disponible en: `https://bit.ly/3e7qgol`.

[16] S. L. Mora, *Programación de aplicaciones web: historia, principios básicos y clientes web*. Editorial Club Universitario, 2002-10-31.

[17] R. V. Lerma-Blasco, J. A. M. Andrés, and E. M. Talón, *Aplicaciones Web Ciclo Formativo Grado Medio*. McGraw-Hill/Interamericana de España, S.L., 2013.

[18] R. D. Caballar, "La programación por voz puede ser la proxima frontera en el desarrollo de software," *IEEE SPECTRUM*, Mar. 2021. [Internet]. Disponible en: `https://bit.ly/3ecQ1mZ`.

[19] M. Merino, "Escribir código usando reconocimiento de voz, una nueva barrera superada en el desarrollo de software (con algunos inconvenientes)." [Internet]. Disponible en: `https://bit.ly/3snxz0h`.

[20] (2022), "Modelo de prototipado en ingeniería de software: Metodología, Proceso, Enfoque." [Internet]. Disponible en: `https://bit.ly/3JrMekp`.