ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN SISTEMA PROTOTIPO DE CÁLCULO NUMÉRICO-ALGEBRAICO INTERACTIVO PARA ESTUDIANTES CON DISCAPACIDAD VISUAL

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN INGENIERÍA ELECTRÓNICA Y REDES DE INFORMACIÓN

DANILO ISAÍAS PILACUÁN MONTALVO

danilo.pilacuan@gmail.com

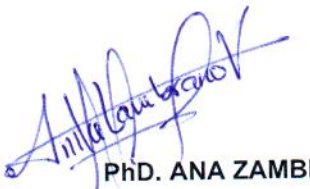
DIRECTOR: PhD. ANA MARÍA ZAMBRANO VIZUETE

ana.zambrano@epn.edu.ec

Quito, noviembre 2022

AVAL

Certifico que el presente proyecto fue desarrollado por Danilo Isaías Pilacuán Montalvo, bajo mi supervisión.



PhD. ANA ZAMBRANO
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Danilo Isaías Pilacuán Montalvo, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

DANILO ISAÍAS PILACUAN MONTALVO

DEDICATORIA

A mi madre, por su amor incondicional y todo el apoyo que me ha brindado en todas las etapas de mi vida hasta conseguir este objetivo.

A mis abuelitos, por acogernos en su hogar donde he vivido los mejores años de mi vida, rodeado de su amor y comprensión.

A mis tíos, por ser parte integral en mi desarrollo como persona de bien y no solo brindarme su apoyo, sino también su amistad incondicional.

A mis hermanas, por servirme de inspiración para ser cada vez mejor.

Danilo Isaías Pilacuán Montalvo

AGRADECIMIENTO

Un agradecimiento especial a la Dra. Ana María Zambrano Vizuete, directora del presente Trabajo de Titulación, quien no solo es una excelente docente, sino también una gran persona, por su apoyo incondicional durante el desarrollo de este trabajo, por su oportuna guía profesional, su comprensión y motivación para seguir adelante en cada obstáculo presentado. Pues gracias a ello, este proyecto ha llegado a buen puerto.

A su vez, extiendo mis sinceros agradecimientos al Dr. Felipe Grijalva, quien a través de su vasta experiencia ha sido de gran ayuda para la resolución de numerosas etapas de este Trabajo de Titulación.

A los docentes de la carrera de Ingeniería en Electrónica y Redes de la Información de la Escuela Politécnica Nacional, por la dedicación a su trabajo de formación de profesionales que contribuyan con el desarrollo del país.

A los amigos que he hecho a través de los años, con quienes he compartido no solo aulas, sino también muchas experiencias que nos pusieron a prueba y que gracias al apoyo mutuo hemos podido superar.

Mi más efusivo agradecimiento a Elsa Gabriela Túquerres Guerrero, por ser parte fundamental en mi formación, por tantos años de apoyo y enseñanzas, quien a través del cariño y comprensión me llevó a descubrir un potencial en el que no creía y que ahora abrazo con gran esperanza para ser mejor cada día.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE CÓDIGOS

ÍNDICE DE TABLAS

RESUMEN

El presente Trabajo de Titulación comprende el desarrollo de un Sistema Prototipo de Cálculo Numérico-Algebrico Interactivo para Estudiantes con Discapacidad Visual que se encuentren cursando estudios de ingeniería y ciencias exactas. El presente documento se encuentra constituido por cuatro capítulos que cubren el desarrollo del sistema, los cuales serán detallados a continuación.

El primer capítulo aborda el estudio e investigación acerca de aplicaciones y servicios web, así como también la metodología Kanban, arquitectura en capas y se definen los lenguajes de programación a utilizarse.

El segundo capítulo se conforma de los apartados de Diseño e Implementación, siendo el de Diseño en donde se definen los requerimientos funcionales y no funcionales que debe satisfacer el prototipo, así como también las historias de usuario y diagramas de casos de uso que se desprenden de estos, se definen también los módulos con los que contará el sistema y los usuarios objetivo y sus respectivos roles, así como también se maqueta la estructura con la que contará la interfaz gráfica del prototipo. Por su parte, en el apartado de Implementación se codifica los módulos de acuerdo a lo propuesto en el apartado de Diseño.

El tercer capítulo consta de los resultados de las pruebas realizadas para la validación del sistema prototipo y se constata el cumplimiento de requerimientos funcionales y no funcionales.

El cuarto capítulo expone las conclusiones a las que se ha llegado mediante el desarrollo del sistema prototipo y las recomendaciones a seguir para trabajos futuros.

PALABRAS CLAVE: Web Speech API, Kanban, Amazon Web Services, VueJS, ExpressJS, Python.

ABSTRACT

This Degree Work includes the development of an Interactive Numerical-Algebraic Calculation Prototype System for Students with Visual Impairment who are studying engineering and exact sciences. This document is made up of four chapters that cover the development of the system, which will be detailed below.

The first chapter addresses the study and research on web applications and services, as well as the Kanban methodology, layered architecture, and the programming languages to be used are defined.

The second chapter is made up of the Design and Implementation sections, being the Design section where the functional and non-functional requirements that the prototype must satisfy are defined, as well as the user stories and use case diagrams that emerge from it, the modules that the system will have and the target users and their respective roles are also defined, as well as the structure that the graphic interface of the prototype will have. On the other hand, in the Implementation section, the modules are coded according to what is proposed in the Design section.

The third chapter consists of the results of the tests carried out for the validation of the prototype system and the fulfillment of functional and non-functional requirements is verified.

The fourth chapter presents the conclusions reached through the development of the prototype system and the recommendations to follow for future work.

KEYWORDS: Web Speech API, Kanban, Amazon Web Services, VueJS, ExpressJS, Python.

1. INTRODUCCIÓN

Según la Organización Mundial de la Salud (OMS) existen aproximadamente 39 millones de personas totalmente ciegas y 246 millones con alguna deficiencia visual y que, en países como el Ecuador, un país de bajos ingresos, existe una mayor prevalencia. Según el Consejo Nacional para la Igualdad de Discapacidades, se registra que el 11.6%, correspondiente a 55.843 personas, registran un grado de discapacidad visual; y de este porcentaje, alrededor del 40% posee un grado de discapacidad mayor al 75% [1]. Toda esta población tiene una menor probabilidad de ser escolarizadas; y de la misma manera, tienen una menor probabilidad de ser empleadas. La educación inclusiva es un tema de interés mundial, el cual, a pesar de los esfuerzos realizados en la implementación de políticas de inclusión, presenta distintos desafíos que alejan a las personas con capacidades diferentes de una correcta educación al carecer de herramientas que adapten los procesos educativos a sus realidades; lo cual dificulta su ingreso, permanencia y egreso de las instituciones de educación. Como claro ejemplo tenemos que, en Ecuador, el 37,9% de la población con discapacidad no ha recibido educación formal alguna y apenas el 1,8% ha tenido acceso a la educación superior, de los cuales el 62,9% eran varones y de ellos, solamente el 1.9% logró alcanzar un título de tercer nivel [2]. El presente Proyecto de Titulación se encamina a solucionar una de las carencias actuales en cuanto a herramientas educativas existentes para personas no videntes, permitiendo dotarles de una aplicación que les permita realizar cálculos numérico-algebraicos a través de una interfaz interactiva que trabaje con traducción de texto a voz y viceversa, adaptándose así a sus capacidades y brindándoles un soporte para su formación académica en ámbitos de ingeniería.

1.1 OBJETIVOS

El objetivo general de este proyecto es desarrollar un sistema prototipo de cálculo numérico-algebraico interactivo para estudiantes con discapacidad visual.

Además, el presente Proyecto de Titulación cuenta con los siguientes objetivos específicos:

- Analizar tecnologías, herramientas y lenguajes necesarios que se acoplen de manera adecuada a cada módulo del Sistema.
- Analizar los requerimientos funcionales y no funcionales de un sistema software para personas con discapacidad visual.

- Diseñar los módulos que componen cada una de las capas correspondientes al sistema.

- Implementar el sistema en función al diseño planteado.

- Analizar los resultados obtenidos en las pruebas funcionales y no funcionales realizadas con ayuda de estudiantes voluntarios con discapacidad visual.

1.2 ALCANCE

El presente Proyecto de Titulación propone el desarrollo de un sistema interactivo alojado en la nube con una arquitectura de capas y estructurado en servicios, el mismo que se desarrollará con el objetivo de realizar cálculos numérico-algebraicos orientado al uso por parte de estudiantes con discapacidad visual. Para manipular el sistema se utilizarán una consola de comandos y una herramienta de scripting para comunicarse con el Motor Matemático MAXIMA, con el fin de manipular expresiones simbólico-numéricas como diferenciación, integración y transformadas dando paso a la inclusión de personas con discapacidad visual a la educación superior ecuatoriana.

El sistema será alojado en uno de los Proveedores Cloud líderes en la industria, como lo es ***Amazon Web Services***, tomando como referencia el modelo ***Infraestructure as a Service (IaaS),*** albergándolo en un servidor **Ubuntu 20.04 LTS** en el cual se instalarán todos los paquetes necesarios para el funcionamiento de cada capa del sistema [3].

El sistema será estructurado en base a cuatro capas, que serán: **Capa de Datos, Capa de Negocio, Capa de Comunicación con el Motor Matemático y Capa de Presentación**.

Para la realización del sistema se utilizará un lenguaje de **Programación Orientado A Objetos**, lo que permitirá desarrollar la lógica de negocio y que trabajará con la Capa de Datos realizada en ***MongoDB*** para su almacenamiento.

Una vez terminado el Proyecto de Titulación se contará con un producto final demostrable implementado en el proveedor cloud escogido y accesible a través de un navegador web.

Se definirán dos tipos de usuarios en el sistema: **Profesor y Estudiante**; además de un usuario **Administrador** encargado de la gestión de los anteriores usuarios.

Para satisfacer los requisitos funcionales y no funcionales del sistema se han definido los siguientes módulos:

- Módulo de usuarios.
- Módulo de profesor.
- Módulo de estudiante.
- Módulo de cálculo.

Módulo de usuarios:

- Existirá un usuario Administrador que será el encargado de gestionar el sistema.
- El usuario Administrador podrá crear usuarios tanto Profesores como Estudiantes

Módulo de profesor:

- Los usuarios de tipo Profesor podrán crear cursos.
- Los usuarios de tipo Profesor podrán crear lecciones enfocadas a describir el uso del sistema a los estudiantes.
- Los usuarios de tipo Profesor podrán crear actividades orientadas a evaluar el desempeño de los estudiantes.
- Los usuarios de tipo profesor podrán calificar actividades o brindar retroalimentación

Módulo de estudiante:

- Los usuarios de tipo Estudiante podrán acceder a un curso.
- Los usuarios de tipo Estudiante podrán tomar lecciones dictadas por el profesor.
- Los usuarios de tipo Estudiante podrán realizar actividades a evaluarse por el profesor.
- Los usuarios de tipo Estudiante podrán revisar el resultado de las actividades realizadas una vez sean revisadas por los usuarios de tipo Profesor.

Módulo de cálculo:

- Los usuarios de tipo Profesor podrán cargar demostraciones del uso del módulo.
- Los usuarios de tipo Estudiante podrán escuchar las instrucciones detalladas por los usuarios de tipo Profesor para el uso del módulo del sistema.
- Los usuarios de tipo Estudiante podrán crear un script a través de comandos de voz.
- Los usuarios de tipo Estudiante podrán ejecutar los scripts que han realizado y escuchar sus resultados.
- Los usuarios de tipo Estudiante podrán crear un script a través de teclas de acceso rápido.
- Los usuarios de tipo Estudiante y Profesor pueden evaluar cálculos matemáticos a través de consola de comandos.

En la Figura 1.1 se puede visualizar la utilización de cada uno de los módulos de acuerdo a cada uno de los usuarios:

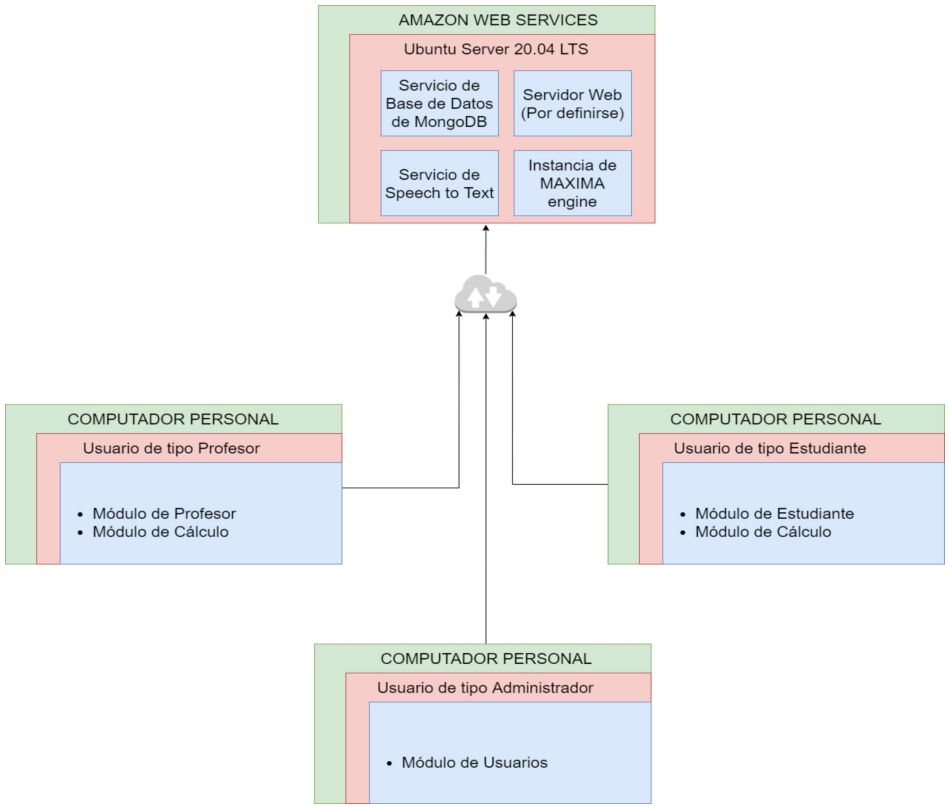


Figura 1.1 Descripción de módulos y usuarios del sistema

1.3 MARCO TEÓRICO

En esta sección se presentarán los antecedentes que dan origen a la necesidad del desarrollo del sistema prototipo, además de los conceptos utilizados para el desarrollo del presente Proyecto de Titulación, detallándose la arquitectura de capas de la que consiste el mismo, así como también, las herramientas utilizadas para el diseño de estas y la metodología Kanban la cual es la base para la planificación de este proyecto.

1.3.1 ESTADO DEL ARTE EN EDUCACIÓN INCLUSIVA

En Ecuador, el Gobierno Nacional es el ente encargado de gestionar el cumplimiento del proceso educativo mediante herramientas como la Ley Orgánica de Educación Intercultural (LOEI), el Plan Nacional de Desarrollo y la Estrategia de Desarrollo del Plan Toda una Vida, Ecuador 2030 [4]; sin embargo, y a pesar de los esfuerzos particulares en cuanto a la capacitación de los docentes, se presenta un problema mayor frente al avance en la educación inclusiva y es la falta de herramientas didácticas que favorezcan y faciliten la práctica docente, lo que ocasiona que la "inclusión" sea poco efectiva y quede relegada únicamente a una "integración" de los estudiantes en los centros educativos [5].

El presente proyecto de Titulación aborda una de las carencias actuales de las herramientas educativas existentes para personas invidentes al brindarles una aplicación que les permita realizar cálculos numéricos y algebraicos a través de una interfaz interactiva que funciona con traducción de texto a voz y viceversa, y contribuye al desarrollo de habilidades que permita su formación académica y su posterior participación en campos técnicos.

Actualmente, los trabajos o proyectos vinculados a la Ingeniería que involucran la inclusión de personas no videntes son escasos y limitados. La gran mayoría de proyectos encontrados se basan en mejorar la vida social de un no vidente; sin embargo, en el ámbito matemático y de álgebra compleja, que incluya derivadas, integrales, etc. la falta de investigación e innovación es clara y certera.

Sobre una búsqueda exhaustiva sobre el Estado del Arte, existen herramientas que ya han saltado el límite de solamente presentar y editar contenido matemático para no videntes. En un estudio del Estado de Arte se han encontrado los siguientes sistemas:

- ***SZSLatex Editor* [6]**: Es un editor de látex y ofrece una versión simplificada para ingresar este tipo de código. La principal desventaja es que los usuarios deben

tener algún conocimiento previo de la sintaxis de *Latex* y la versión actual de *SZSLatex* no tiene una interfaz de audio nativa para interactuar. Esto, a su vez, hace que este editor dependa de lectores de pantalla de terceros que no están diseñados para leer código *Latex*. Además, no se envía ningún comentario al VIP cuando se producen errores de sintaxis.

- **El Editor de Ecuaciones Accesible (AEE) de Pearson [7]:** Permite crear expresiones matemáticas a través de una aplicación web que se basa en lectores de pantalla externos como NVDA o JAWS. Admite entradas y salidas de Braille para una amplia gama de notaciones matemáticas. Para las entradas de Braille, AEE emplea una pantalla Braille externa actualizable, y para la salida convierte de MathML (lenguaje de marcado basado en XML cuyo objetivo es expresar notación matemática) a formato Braille.
- **El sistema *L-MATH* [8]**: Permite la edición e inspección de fórmulas matemáticas. La escritura y la lectura de expresiones matemáticas se logran mediante los módulos *BlindMath* y *TalkingMath*, respectivamente. Con *BlindMath*, el estudiante con discapacidad visual puede ingresar fórmulas matemáticas usando un teclado de computadora que luego se puede convertir a código *LaTeX.* Por otro lado, *TalkingMath* utiliza un algoritmo adaptativo original para leer fórmulas según la lectura humana habitual [8].
- **El sistema DOSVOX [9]**: Abarca más de 80 herramientas de código abierto a las que se puede acceder a través de menús hablados que permiten a los usuarios no videntes realizar varias actividades como enviar correos electrónicos, reproducir música y crear documentos y hojas de cálculo. DOSVOX es un sistema autónomo diseñado por la Universidad Federal de Río de Janeiro y actualmente se utiliza ampliamente en Brasil. Dentro del sistema DOSVOX existen dos herramientas que permiten la ejecución de operaciones matemáticas: MATVOX y FINANVOX.
- **MATVOX [10]**: Es un intérprete de algoritmos informáticos que ayuda a escribir y compilar pseudocódigo desde un editor de texto llamado EDIVOX. La segunda versión de esta herramienta agrega soporte a nuevos tipos de variables como números complejos y matrices.
- **La calculadora financiera FINANVOX [11]**: Permite realizar cálculos financieros y estadísticos. Esta herramienta informática, permite realizar operaciones como interés compuesto, amortización, depreciación, media, desviación estándar entre otras. Esto realiza emulando y ampliando las funciones de la popular calculadora financiera *Hewlett-Packard* HP-12C [12].

- ***LAMBDA-Linear Access to Mathematics for Braille Device and Audio-Synthesis* [13]**: Se basa en una notación matemática lineal (muy similar a *MathML*) que permite el acceso a expresiones matemáticas a través del código Braille y voz sintética. Este sistema cuenta con una versión de Braille con 256 caracteres únicos (Código LAMBDA) basada en la representación en Braille de 8 puntos, que incluye nuevos símbolos que permiten la representación de las matemáticas en forma lineal. Estos símbolos se pueden representar de forma visual y en Braille. Entre las características distintivas de LAMBDA está su capacidad para resolver operaciones matemáticas básicas (por ejemplo, operaciones de suma, factoriales, trigonométricas).

En primera instancia, en un país como el Ecuador, un país en desarrollo, el cual no ha priorizado la inclusión, el tener estos dispositivos (pantallas Braille, por ejemplo) en cada aula de clase no es, ni (posiblemente) sea una realidad. Una ventaja de esta propuesta frente a las anteriores es el mínimo costo al idear un sistema sin elementos y dispositivos de hardware externos. El sistema podrá ser instalado y ser funcional en cualquier computador con características suficientes como para correr un software matemático como *MATLAB*, *OCTAVE* o *MAXIMA*; pudiendo ser desplegado a cualquier número de personas que lo necesiten. De las citadas antes, solamente un par de herramientas *LAMBDA* y *FINANVOX* van más allá de presentar y editar fórmulas matemáticas permitiendo incluso resolverlas.

1.3.2 TECNOLOGÍAS Y LENGUAJES DE PROGRAMACIÓN UTILIZADOS EN EL PROTOTIPO

Este Apartado presenta un estudio básico de aquellas tecnologías y lenguajes a utilizar en la implementación del prototipo. Además, se estudiará la metodología a utilizar. Esto ayudará al entendimiento total en las etapas de diseño e implementación.

El sistema seguirá una arquitectura basada en capas, siendo estas, cuatro, las cuales se denominan: Capa de Datos, Capa de Negocio, Capa de Comunicación con el Motor Matemático y finalmente, la Capa de Presentación.

Para el desarrollo del sistema prototipo se utilizará la pila de desarrollo conocida como MEVN [14]; siendo este el acrónimo de *MongoDB, Express, VueJS y NodeJS* las cuales son tecnologías de código utilizadas para el desarrollo de aplicaciones web.

1.3.2.1 Capa de Datos

MongoDB [15] es un gestor de bases de datos documentales no relacionales conocidas como *NoSQL*; para este propósito *MongoDB* almacena colecciones de documentos.

Estos documentos almacenan a su vez colecciones de datos de pares *clave-valor* que pueden almacenar varios tipos de datos. Las colecciones son el mecanismo que utiliza *MongoDB* para organizar los datos asignándoles un *Object Identifier*, el cual es un identificador autogenerado único para cada documento permitiendo el acceso al mismo. Los datos son almacenados nativamente en formato *BSON* (*Binary JavaScript Object Notation*); este formato permite a *MongoDB* insertar varios tipos de datos volviéndolo agnóstico del lenguaje de programación y permitiendo así un almacenamiento de datos más eficiente con respecto a gestores de bases de datos tradicionales como *SQL*.

1.3.2.2 Capa de Negocio

NodeJS [16] es un entorno de ejecución de aplicaciones escritas en *JavaScript*. NodeJS surge debido a que *JavaScript* por sí solo es únicamente un lenguaje de programación computacional incapaz de ejecutarse en un servidor, para lo cual es necesario un entorno de ejecución como NodeJS. Este añade funcionalidades como el acceso al sistema de archivos, el acceso a la red, entre otras.

NodeJS ha sido desarrollado utilizando el motor ***Chome V8*** lo cual le permite ejecutarse en múltiples plataformas como pueden ser *Linux, Windows* o *Mac OSX*.

ExpressJS [17] o simplemente conocido como ***Express*** es un marco de trabajo (*framework*) perteneciente a *NodeJS* que simplifica el desarrollo de aplicaciones escritas en *JavaScript* del lado del servidor. A pesar de existir la posibilidad de desarrollar aplicaciones del lado del servidor nativamente en *NodeJS, Express* ayuda al desarrollador a alcanzar resultados más eficientes y en menor tiempo añadiendo además funcionalidades adicionales a través el uso de paquetes conocidos como *Middlewares*.

1.3.2.3 Capa de Comunicación con MAXIMA

Python [18] es un lenguaje de programación multiparadigma de alto nivel de propósito general. Además *Python* es un lenguaje de programación multiplataforma y de código abierto, cuya filosofía hace énfasis en la facilidad de escritura del código y la rápida

implementación de prototipos; características que lo han dotado de gran popularidad en el mundo del desarrollo [18].

Por otro lado, ***SnuggleTex*** [19] es una librería Java de código abierto que permite convertir fragmentos de código escrito en *Latex* a XML. *SnuggleTex* dispone de distintos modos de trabajo que le permiten además de convertir de *Latex* a XML, escoger otros formatos de salida como *MathML, CMathML*, *PMathML* e incluso páginas web independientes que muestran los resultados de la conversión.

MathML o Mathematical Markup Language [20] es un lenguaje de etiquetado basado en *XML* que define un método de representación estructurada de expresiones matemáticas. *MathML* ha permitido una estandarización de la representación de expresiones matemáticas, por lo que es usado en múltiples softwares matemáticos, así como también en navegadores web.

MAXIMA [21] es un sistema de álgebra computacional utilizado para el desarrollo y resolución de expresiones matemáticas simbólicas y numéricas, el cual incluye diferenciación, integración, expansión de series de Taylor, transformadas de Laplace, ecuaciones diferenciales, resolución de sistemas de ecuaciones, vectores y matrices. MAXIMA es un software de código abierto que se encuentra disponible para *Windows*, *Linux* y *MacOS X.*

Entre los aspectos más relevantes de MAXIMA se encuentran su naturaleza libre, comportamiento enmarcado por su licencia de distribución GPL, que brinda al usuario las ventajas de:

- Libertad de uso
- Libertad de modificación y adaptación en función de las necesidades del usuario
- Libertad de distribución
- Libertad de estudio y aprendizaje de su funcionamiento

El uso de la licencia GPL y sus correspondientes libertades han hecho que MAXIMA se convierta en una formidable herramienta pedagógica, de investigación y de realización de cálculos técnicos accesible a toda la población, tanto para uso individual como institucional.

1.3.2.4 Capa de Presentación

VueJs [22] es un *framework* de desarrollo web progresivo basado en el lenguaje de programación *JavaScript* cuyo propósito principal es facilitar la creación de interfaces web dinámicas conocidas como *SPAs (Single Page applications)*. Al contrario de otros framework de desarrollo web que adoptan un enfoque monolítico encapsulando todas sus capacidades en un núcleo grande como es el caso de Angular, *VueJS* adopta un enfoque modular que puede ser extendido si así se requiere, lo que permite a las aplicaciones objetivo ser livianas y eficientes.

Web Speech API [23] tiene como objetivo permitir la adición de características de entrada y síntesis de voz a aplicaciones web que normalmente carecen de las mismas y cuya implementación no es viable a través de lectores de pantalla estándar. Esta *API* es agnóstica de la plataforma y soporta procesamiento tanto en el servidor como en el cliente de la aplicación web que la utilice.

1.3.2.5 Metodología Kanban

KANBAN es una metodología ágil de desarrollo de software que reduce el tiempo de iteración entre actividades y mejora la calidad de desarrollo de cada proceso evitando cuellos de botella. La Metodología Kanban se aplica en proyectos de desarrollo de software para visualizar el flujo de trabajo y limitarlo a etapas medibles con un ciclo de vida corto, lo cual permite abordar el proceso de desarrollo de una manera eficaz [24].

Para la organización de un proyecto siguiendo la metodología Kanban, se plantea un tablero conocido como Tablero Kanban, el cual se encarga de mostrar el trabajo designado a cada desarrollador, comunica las prioridades y evita iteraciones obligatorias para concentrarse solamente en los elementos que se solicitan en la etapa de desarrollo en la que se encuentra el proyecto. Este tablero debe actualizarse constantemente durante la etapa de desarrollo, para así concentrar los esfuerzos únicamente en la tarea asignada.

En la Tabla 1.1 se observa la estructura simplificada a seguirse para la realización de un tablero Kanban:

Tabla 1.1. Estructura simplificada de un tablero Kanban.

POR HACER	EN PROGRESO	REALIZADO
En esta sección se indica las actividades que se realizarán en el futuro.	En esta sección se ubican las actividades que el desarrollador se encuentra realizando actualmente.	En esta sección se informa de las actividades que ya han sido realizadas.

2. METODOLOGÍA

En el presente Capítulo se detallará la planificación, los requerimientos y la arquitectura en capas empleada en el desarrollo del prototipo. En la Apartado 2.1 se muestran las principales tareas a realizar, las tareas en progreso y las tareas completadas mediante el Tablero Kanban, mientras que en la Apartado 2.2 se describe la instalación y codificación del sistema en detalle.

2.1 PLANTEAMIENTO DEL TABLERO KANBAN

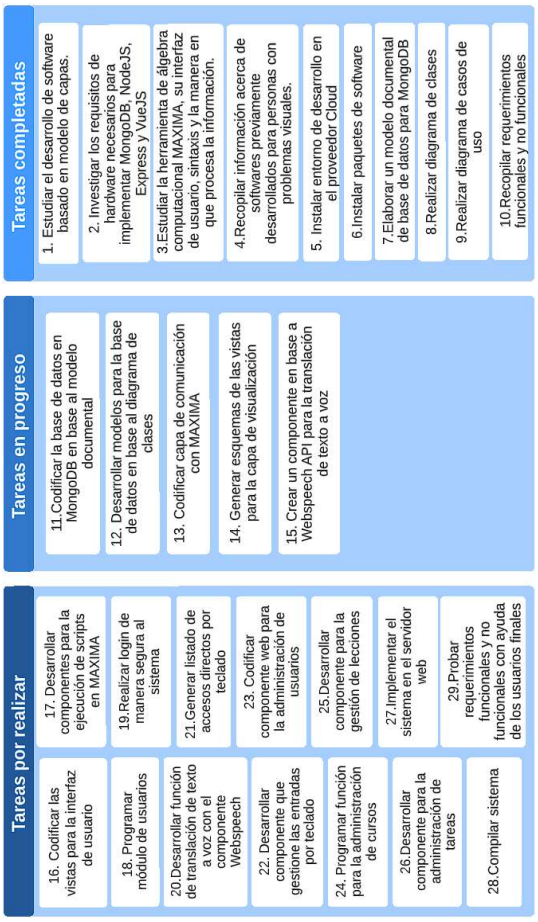


Figura 2.1. Planteamiento de Tablero Kanban en etapa de diseño.

2.2 DISEÑO

En el transcurso de este apartado se mostrará las actividades realizadas en la etapa de Diseño, lo que implica: la realización de diagramas en cada una de las capas. Se hace notar que un buen diseño puede llevar al éxito del desarrollo para cumplir los objetivos finales.

2.2.1 DIAGRAMAS DE CASOS DE USO

Los Diagramas de Casos de Uso constituyen una herramienta guía durante el proceso de desarrollo, pues detallan las necesidades de los usuarios del sistema y permiten establecer los roles que cumple cada actor y el acceso a las funciones del sistema que cumple cada uno [25].

2.2.1.1 Definición de Actores del Sistema

Se define como Actor al rol que un usuario determinado desempeña sobre el sistema y son éstos los que emplean las funciones definidas en el Diagrama de Casos de Usos.

Con base en los Requerimientos Funcionales se han determinado tres Actores para el sistema prototipo, cada uno con acceso a funciones únicas dentro del mismo. A continuación, se detalla las funciones a las que tienen acceso cada actor dentro del sistema:

- **Administrador:** Es el encargado de la gestión de usuarios del sistema, sus funciones son: registrar, editar y eliminar las credenciales para que otros usuarios puedan ingresar al sistema. También es el encargado de restituir el acceso a los usuarios que hayan olvidado sus credenciales y soliciten un cambio de contraseña.

- **Profesor:** Entre sus funciones se encuentran la administración de cursos (crear, editar o eliminar cursos), matriculación de estudiantes y la creación y edición de lecciones. También puede crear, editar o eliminar actividades, publicarlas y calificar las actividades resueltas por los alumnos, además puede acceder al módulo de cálculo para crear archivos de script para la resolución de operaciones matemáticas.

- **Estudiante:** Entre sus funciones están la posibilidad de acceder a los cursos a los que hayan sido matriculados, dentro del curso al que accedan pueden tomar

lecciones, realizar actividades asignadas por los profesores y publicarlas para recibir una calificación y una posterior retroalimentación.

Para la realización del Diagrama de Casos de Uso se ha utilizado la herramienta de modelamiento de diagramas a través de código *PlantUML* [26].

La Figura 2.2 muestra el diagrama de Casos de Uso para el Actor Administrador.

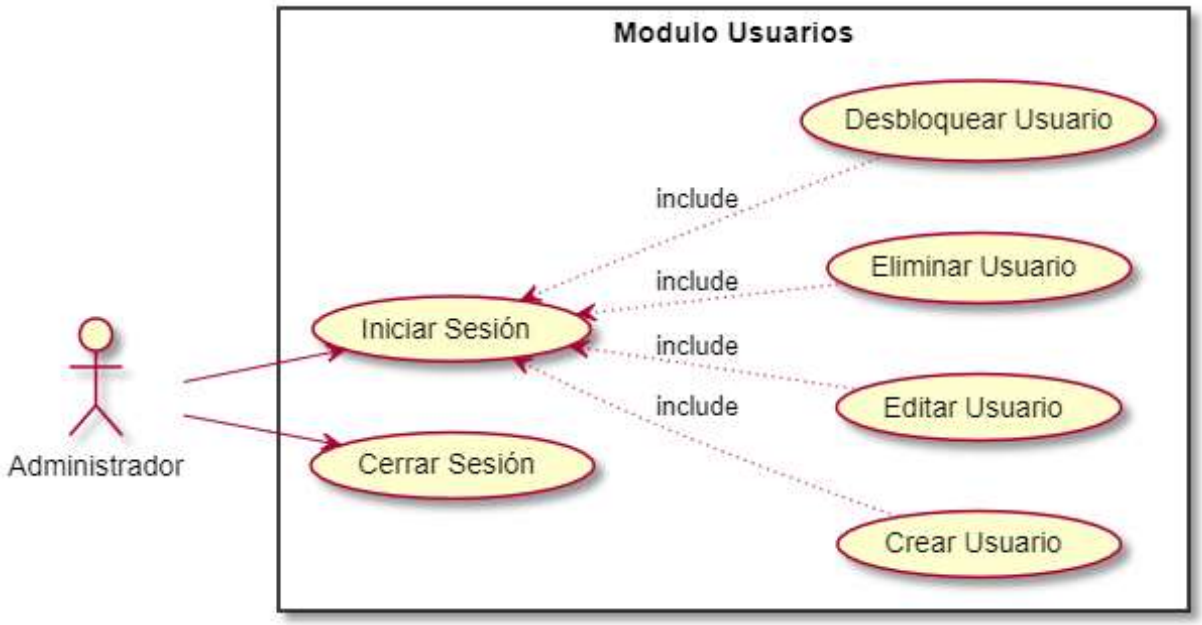


Figura 2.2. Diagrama de casos de uso para el Actor Administrador.

La Figura 2.3 el diagrama de casos de uso para el Actor Alumno.

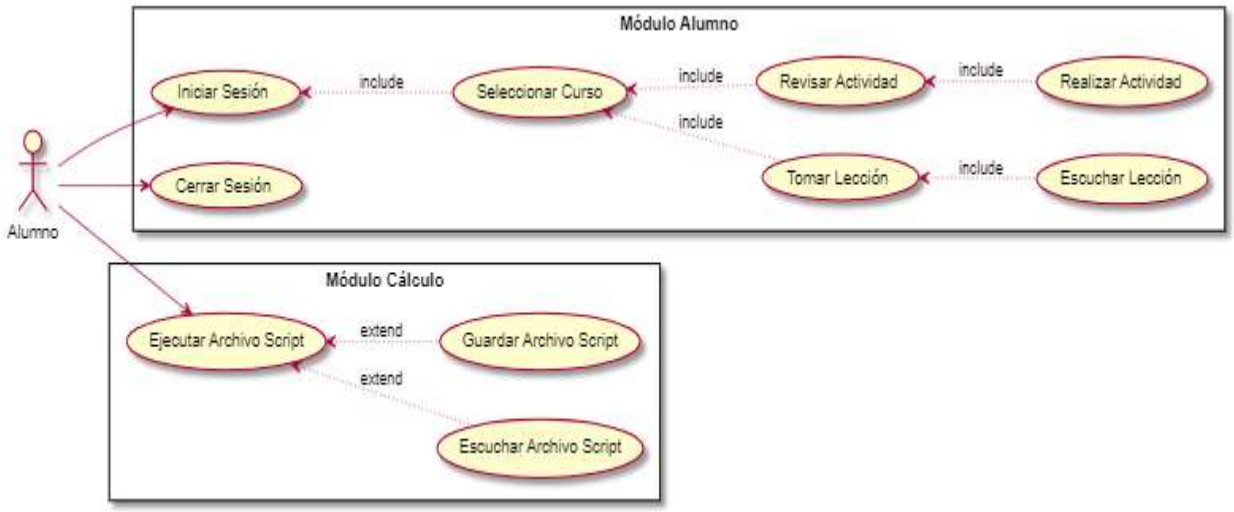


Figura 2.3. Diagrama de casos de uso para el Actor Alumno.

La Figura 2.4 muestra el diagrama de casos de uso para el Actor Profesor.

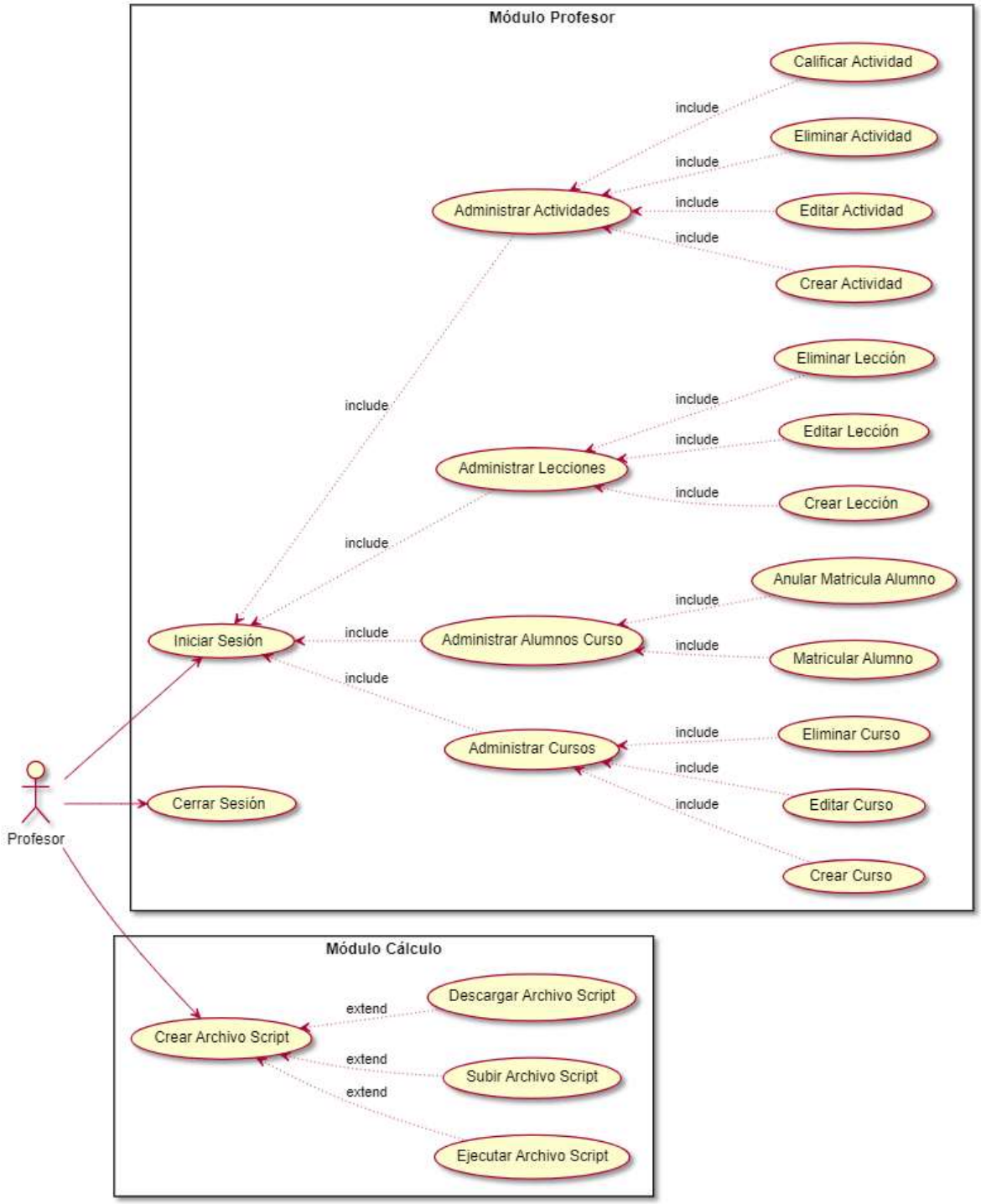


Figura 2.4. Diagrama de casos de uso para el Actor Profesor.

2.2.2 REQUERIMIENTOS FUNCIONALES

Los Requerimientos Funcionales son aquellos que describen las funcionalidades que son de vital importancia y que se deben cumplir necesariamente una vez se haya terminado el desarrollo del prototipo.

Los Requerimientos Funcionales planteados para el desarrollo del prototipo son los que se detallan a continuación:

- RF1. Se debe brindar una retroalimentación audible al usuario no vidente a través de todos los menús a los que tiene acceso, por lo cual se debe desarrollar un componente que asista a los usuarios con estos menesteres, dicho componente deberá dotar al sistema de la funcionalidad de síntesis de texto a voz, de modo que el usuario no vidente esté consciente del menú del sistema en el que se encuentra y sus correspondientes funciones.

- RF2. Se debe añadir la funcionalidad de captura de eventos de teclado pensados especialmente para los usuarios no videntes, y su correspondiente realimentación auditiva, pues estos no cuentan con la capacidad de realizar interacciones mediante una interfaz gráfica.

- RF3. Se debe añadir una retroalimentación auditiva a los comandos de script ingresados por el usuario no vidente.

- RF4. Se debe añadir una navegación simple que permita que los usuarios no videntes puedan desplazarse completamente a través de todo el sistema sin necesidad de la intervención de terceros.

- RF5. Se debe agregar una funcionalidad que permita al usuario controlar la síntesis de voz si así lo requiere, pudiéndose pausar, reanudar o reiniciar.

- RF6. Se debe dotar de la capacidad de dictar el resultado de operaciones matemáticas en lenguaje natural de fácil comprensión para los usuarios no videntes.

- RF7. Se debe agregar un menú de ayuda universal dentro del sistema, que sea accesible al usuario con una combinación simple de teclas.

2.2.3 REQUERIMIENTOS NO FUNCIONALES

Los Requerimientos No Funcionales son aquellos que describen funcionalidades adicionales o complementarias a las establecidas en los Requerimientos Funcionales, con ayuda de estos se asegura de contar con un sistema de calidad que cumpla con los objetivos planteados.

Los Requerimientos No Funcionales planteados para el desarrollo del prototipo son los que se detallan a continuación:

- RNF1: Se debe dotar al sistema de una interfaz gráfica amigable e intuitiva.
- RNF2: El sistema prototipo no debe tener una curva de aprendizaje muy compleja para su utilización, permitiendo así una rápida comprensión de las funcionalidades de sus módulos.
- RNF3: La ejecución del sistema prototipo debe ser fluida sin largos retardos de procesamiento o repetición de solicitudes.
- RNF4: El sistema prototipo debe ser robusto, no deben haber caídas del servicio.
- RNF5: El sistema prototipo debe ofrecer una baja latencia entre el mismo y los usuarios finales.
- RNF6: El sistema debe contar con la seguridad necesaria para que solo usuarios autenticados puedan acceder al mismo.
- RNF7: El sistema prototipo debe proveer de fidelidad matemática en el cálculo de resultados.
- RNF8: El sistema prototipo debe ofrecer una tasa de disponibilidad satisfactoria para proveer su servicio a los usuarios en cualquier momento.
- RNF9: El sistema prototipo debe brindar la capacidad necesaria para atender a un número grande de usuarios a la vez.
- RNF10: El sistema prototipo debe contar con la capacidad de escalamiento y posibilidad de actualización para brindarlo de longevidad a través del tiempo.

- RNF11: El sistema prototipo debe ser lanzado como una aplicación en estado de producción, la cual ofrezca una gran compatibilidad con distintos navegadores y su ejecución no requiera de conocimientos técnicos.

2.2.4 DISEÑO DE LA CAPA DE DATOS

Una Base de Datos Documental, también denominada base de datos orientada a documentos es un subconjunto de un tipo de base de datos construida sin el uso del paradigma Entidad-Relación. Las Base de Datos Documentales son herramientas de gran ayuda para el manejo de grandes volúmenes de información [27].

En el caso de MongoDB las agrupaciones de datos no toman el nombre de tablas como lo harían en una Base de Batos Relacional, sino que más bien toman el nombre de Colecciones.

En base a esta premisa se ha diseñado una serie de colecciones que se ilustran en la Figura 2.5 y componen el diagrama de la base de datos del presente prototipo que se ha diseñado en el software Moon Modeler [28].

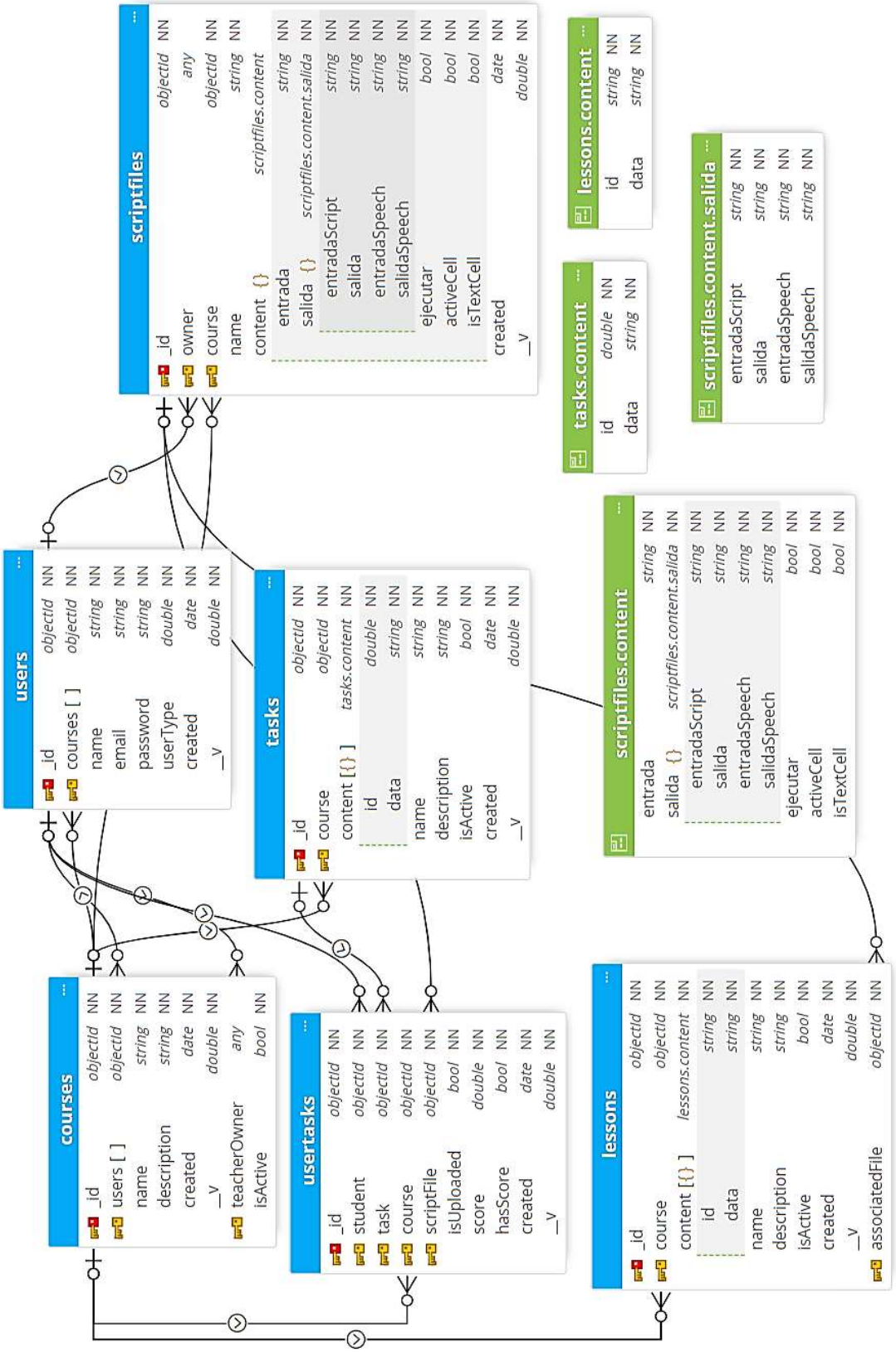


Figura 2.5. Diagrama de base de datos

2.2.5 DISEÑO DE LA CAPA DE NEGOCIO

El diseño de la Capa de Negocio estará regido en base a un Diagrama de Clase. El cual está compuesto por diversos elementos gráficos que trabajan en conjunto para describir cómo trabaja el sistema, mostrando los tipos de objetos que intervienen en el mismo y sus relaciones; convirtiéndose este Diagrama en una útil herramienta de modelado conceptual de un software [29].

2.2.5.1 Diagrama de Clases

El presente Diagrama de Clases ha sido desarrollado mediante la herramienta *PlantUML* y en él se visualizan las clases utilizadas como estructura de la Capa de Negocio. Estas son: *Courses, Lessons, LessonContent, ScriptFiles, ScriptFileContent, ScriptFileOutput, Tasks, TaskContent, UserTasks y Users*.

En la Figura 2.6 se muestra el Diagrama de Clases que rige al desarrollo de la Capa de Negocio.

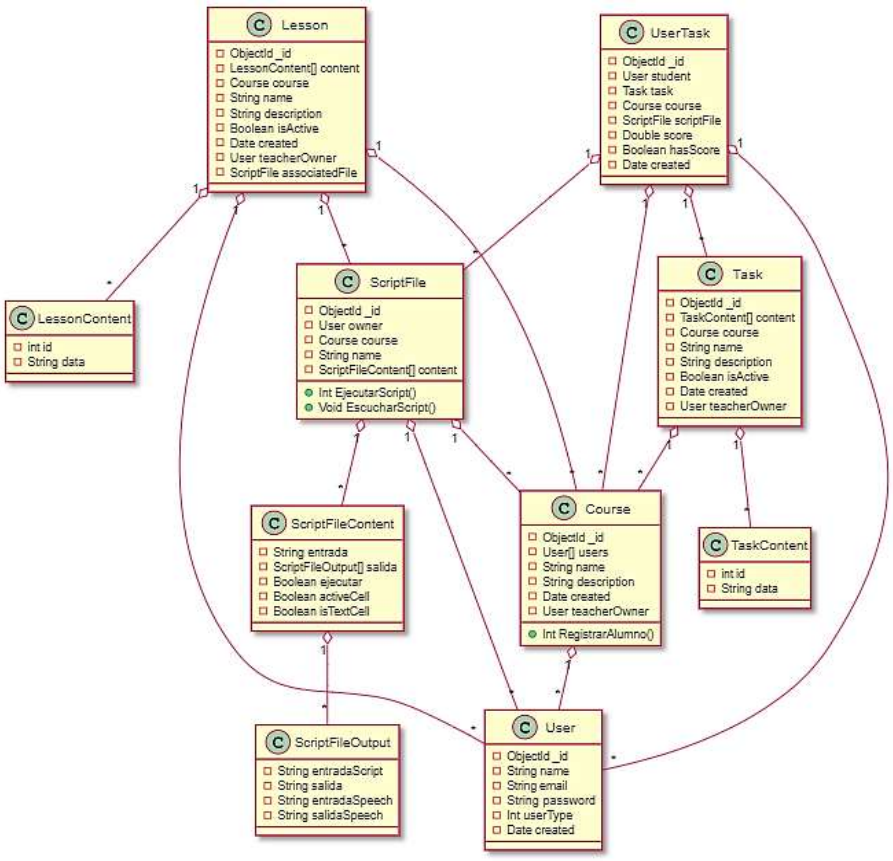


Figura 2.6. Diagrama de Clases.

2.2.6 DISEÑO DE LA CAPA DE COMUNICACIÓN CON MAXIMA

Este apartado aclara la capa más importante del Sistema. Esta capa es la responsable de hacer una conexión a MAXIMA cuando un Actor requiera realizar un cálculo matemático. Se detallarán las diferentes herramientas necesarias para lograr este objetivo.

2.2.6.1 Servicio de Intérprete de Comandos

Un Intérprete de línea de Comandos (CLI) es un software que permite la interacción entre el Usuario y un Sistema Informático. Su propósito es proporcionar al usuario una consola que permita la ejecución de las características del sistema en cuestión, mediante la interacción del teclado como interfaz de entrada y el monitor como interfaz de salida [30].

El servicio de Intérprete de Comandos de MAXIMA permite la ejecución del Sistema de Álgebra Computacional MAXIMA en una ventana de comandos en modo consola, este permite la resolución de operaciones matemáticas mediante el uso de un lenguaje de *script* propio de MAXIMA [31]. A continuación, se presenta un ejemplo de la ejecución del sistema.

La Figura 2.7 detalla un ejemplo de uso del servicio de Intérprete de Comandos de MAXIMA

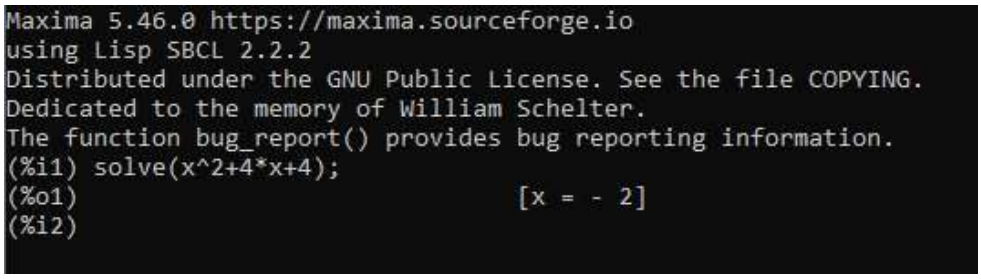


Figura 2.7. Ejemplo del uso del servicio de Intérprete de Comandos de MAXIMA.

Sin embargo el servicio de Intérprete de Línea de Comandos de MAXIMA no permite su interacción por medio de librerías o *API's* que permitan la ejecución de comandos de manera automática; es por ello que se necesita un método de comunicación entre MAXIMA y el resto del sistema a través de una conexión Cliente-Servidor, en donde se define un servidor de procesamiento y en donde MAXIMA actuará como un cliente que se conectará para resolver sus peticiones mediante el comando de inicio de MAXIMA con la opción -s (puerto) [32] la cual inicializará la conexión entre Cliente y Servidor y permitirá la ejecución de comandos de manera automática. La Figura 2.8 detalla la ejecución del sistema

MAXIMA en modo Cliente-Servidor, la cual será usada para la ejecución automática de código.

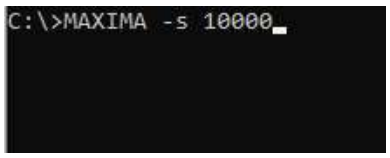


Figura 2.8. Configuración de MAXIMA en modo Cliente-Servidor.

Con la ayuda del intérprete de comandos se definirá un algoritmo para realizar el procesamiento matemático y su posterior translación a texto en lenguaje natural. En la Figura 2.9 se muestra el diagrama de flujo a seguir para la realización de este.

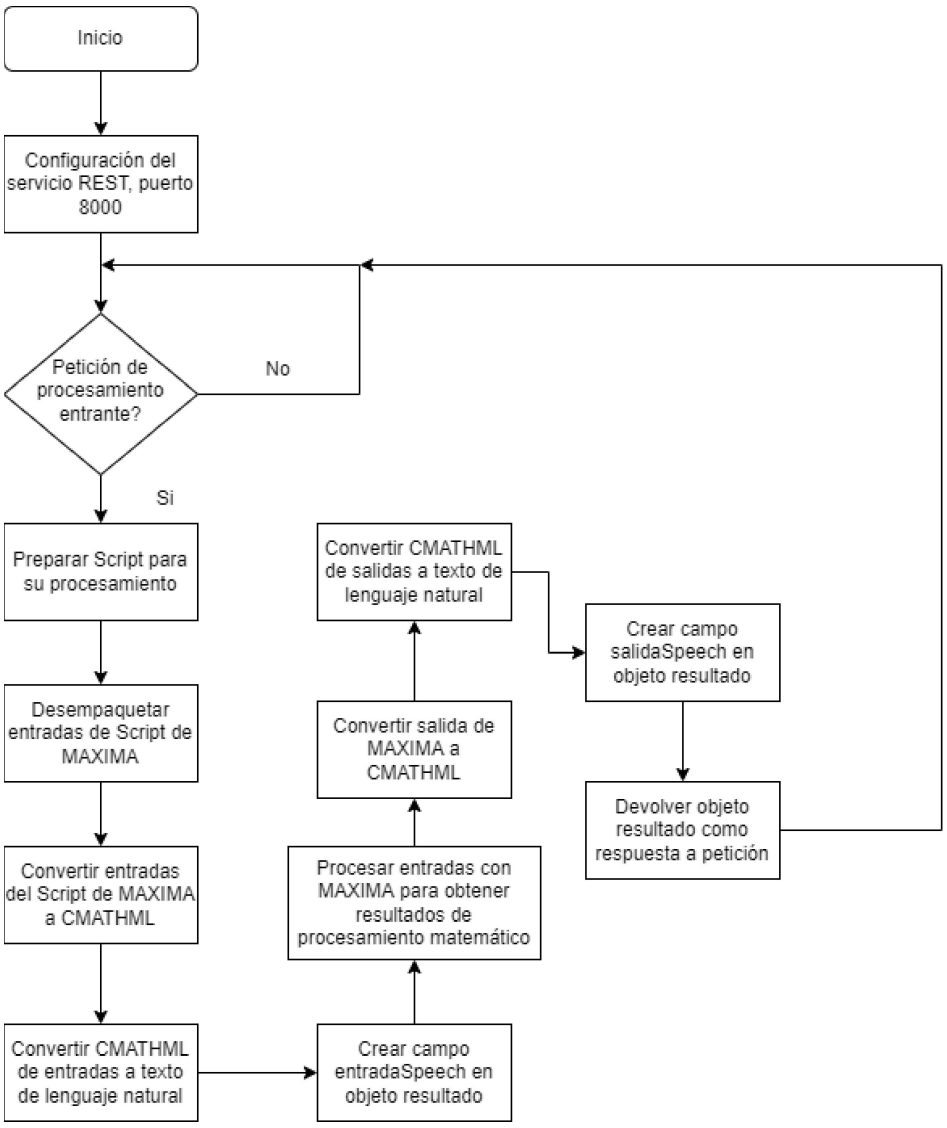


Figura 2.9. Diagrama de flujo de proceso de procesamiento y translación.

2.2.6.2 Interfaz de Programación de Aplicaciones (*API*) para Atención de Solicitudes de Translación

Una *API Rest* [33] es el conjunto de definiciones de métodos que se utilizan para ofrecer un servicio a través del protocolo HTTP, identificado por un URI que permite a varios clientes comunicarse por un protocolo estandarizado; lo cual facilita el desarrollo, centrando este en el desarrollo de la lógica del negocio en lugar del desarrollo de la comunicación.

Mediante la especificación *OpenAPI* [34] se ha desarrollado el diagrama mostrado en la Figura 2.10 que describe los métodos disponibles de la *API Rest*, así como sus direcciones y verbos HTTP necesarios para su ejecución.

Figura 2.10. Diagrama de especificación de API Rest.

2.2.6.3 Diseño de Componente de Audio

El Componente de Audio a utilizarse en el sistema se desarrollará mediante la ayuda de *Web Speech API* [23], misma que permite la incorporación de funcionalidades como reconocimiento de voz y síntesis de voz a un sistema web, lo cual permite dotar de la accesibilidad necesaria al prototipo.

El API Web Speech utiliza Javascript como lenguaje de programación y define una serie de métodos a utilizarse para proveer las funcionalidades de reconocimiento y síntesis de voz, en la Figura 2.11, se detalla los utilizados para el diseño del Componente de Audio del sistema prototipo.

```
var synth = window.speechSynthesis;

var inputForm = document.querySelector('form');
var inputTxt = document.querySelector('.txt');
var voiceSelect = document.querySelector('select');

var pitch = document.querySelector('#pitch');
var pitchValue = document.querySelector('.pitch-value');
var rate = document.querySelector('#rate');
var rateValue = document.querySelector('.rate-value');
```

Figura 2.11. Métodos necesarios para la síntesis de voz con Web Speech API.

A continuación, detallaremos el funcionamiento de cada uno de los mismos:

Window.speechSynthesis: Obtiene el contexto de síntesis de voz global del navegador, este será el encargado de proveer de los demás métodos necesarios para la síntesis de texto a voz.

VoiceSelect: Establece la lista de voces e idiomas disponibles en el navegador para realizar la síntesis de voz.

Pitch: Se define como una variable entre 0 y 1 que establece el tono de voz a utilizarse durante la síntesis de voz.

Rate: Se define como una variable entre 0 y 1 que establece la velocidad a la que se ejecutará la síntesis de voz.

En la Figura 2.12 se detalla el procedimiento para la obtención de voces disponibles en el navegador.

```
function populateVoiceList() {
  voices = synth.getVoices();

  for(i = 0; i < voices.length ; i++) {
    var option = document.createElement('option');
    option.textContent = voices[i].name + ' (' + voices[i].lang + ')';

    if(voices[i].default) {
      option.textContent += ' -- DEFAULT';
    }

    option.setAttribute('data-lang', voices[i].lang);
    option.setAttribute('data-name', voices[i].name);
    voiceSelect.appendChild(option);
  }
```

Figura 2.12. Método de obtención de voces disponibles para síntesis de voz.

Por último, en la Figura 2.13 se detallarán los métodos para control de flujo de la síntesis de voz.

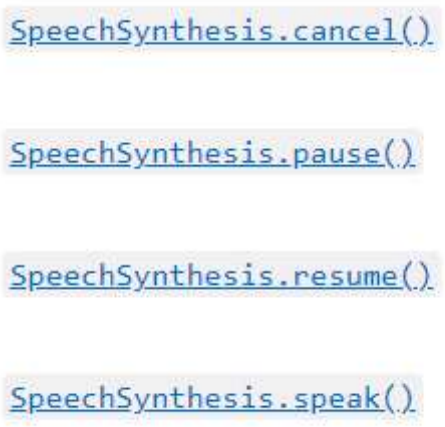


Figura 2.13. Métodos de control de flujo de síntesis de voz de Web Speech API.

A continuación, se procede a detallar las funciones que cumplen cada uno de estos métodos:

- **SpeechSynthesis.cancel():** Para la síntesis de voz y remueve de la cola de procesamiento todas las solicitudes enviadas previamente.
- **SpeechSynthesis.pause():** Pone en pausa la síntesis de voz y queda a la espera de su reanudación, solo puede ser invocado si se ha llamado previamente al método SpeechSynthesis.speak().
- **SpeechSynthesis.resume():** Continúa con la síntesis de voz luego de haberse solicitado una pausa por medio de SpeechSynthesis.pause().
- **SpeechSynthesis.speak(array colaProcesamiento):** Inicia la síntesis de voz tomando como argumentos una cola de procesamiento de solicitudes.

2.2.7 DISEÑO DE LA CAPA DE PRESENTACIÓN

Este apartado dará un pequeño panorama acerca de la visualización del sistema. Aunque parecería que no debería tener tanta importancia la parte visual (dado el usuario objetivo),

el sistema puede ser empleado por personas de diferentes características. Por tal motivo, también será importante detallar el audio a utilizar.

2.2.7.1 Vistas de la Aplicación Web

Se hace notar que las siguientes Figuras, hacen referencia a los bosquejos realizados; y estos han sido pensados para cumplir todos y cada uno de los requerimientos funcionales dictados en el Apartado 2.2.1.

Figura 2.14. Vista previa de la ventana de inicio de sesión.

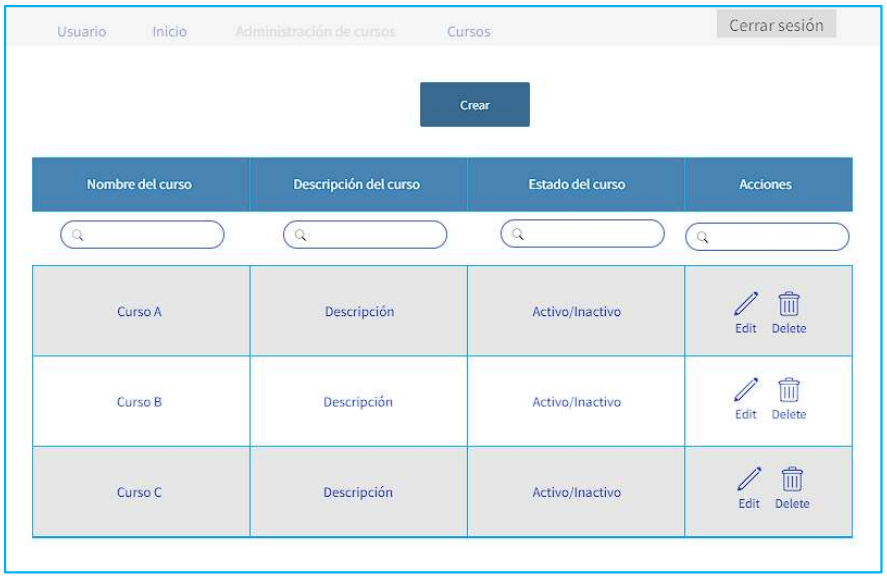


Figura 2.15. Vista previa de la ventana del módulo “Administración de cursos”.

Figura 2.16. Vista previa de la ventana módulo “Cursos”.

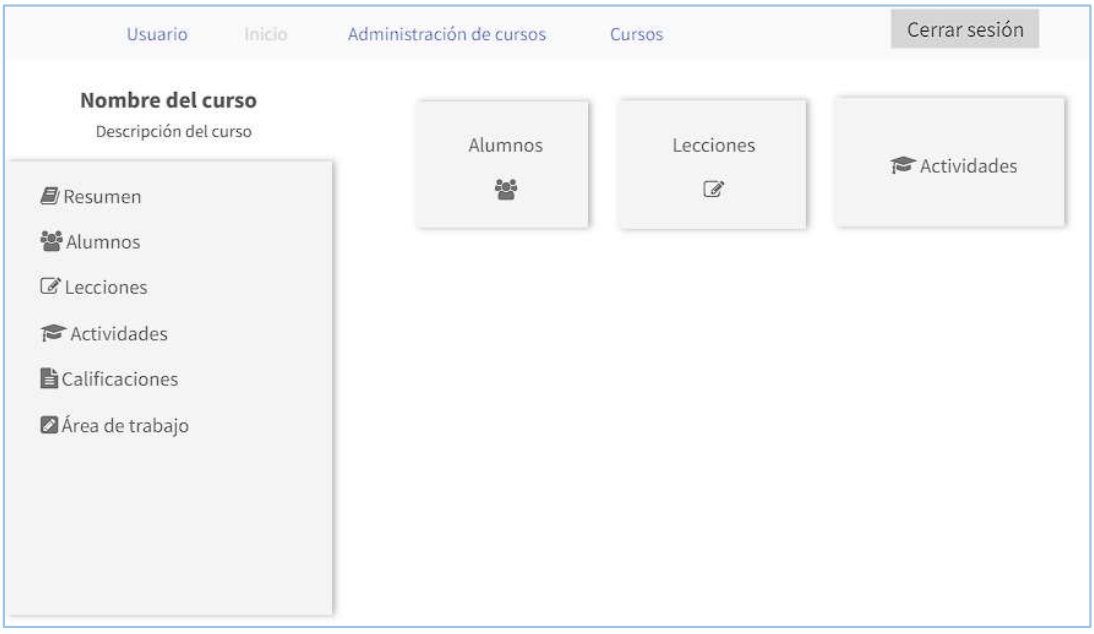


Figura 2.17. Vista previa de la ventana de gestión de curso.

Figura 2.18. Vista previa de la ventana gestión de alumnos.

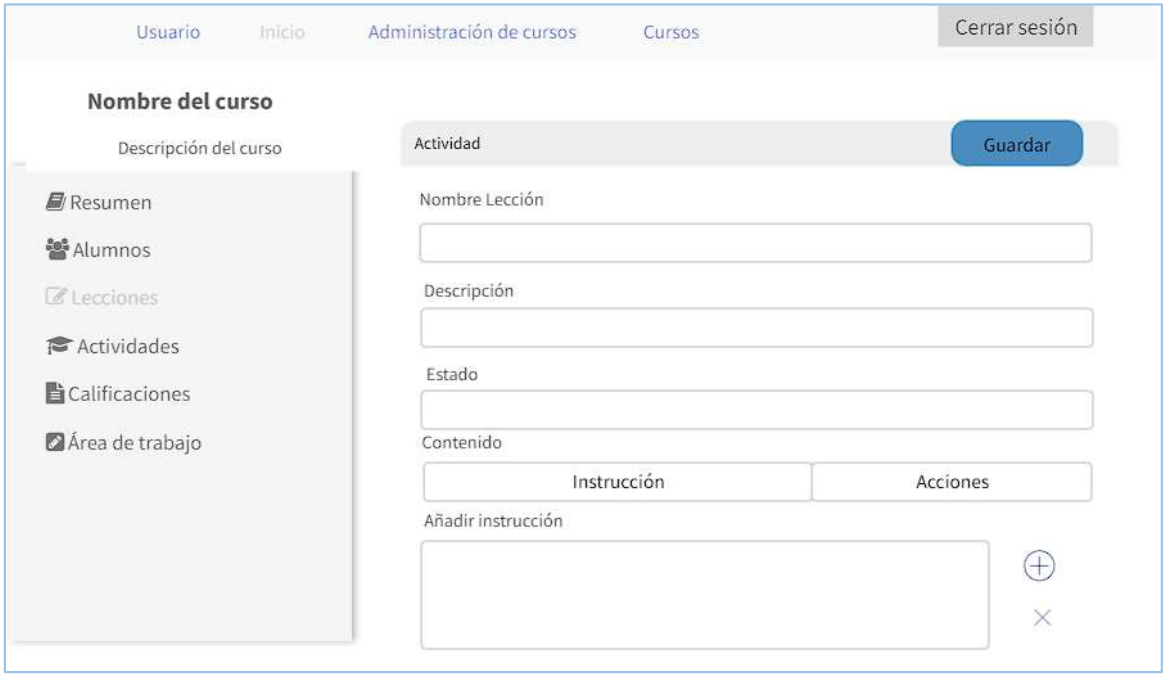


Figura 2.19. Vista previa de la ventana de administración de lecciones.

Figura 2.20. Vista previa de la ventana del área de trabajo.

2.3 IMPLEMENTACIÓN

2.3.1 ACTUALIZACIÓN DEL TABLERO KANBAN

Una vez terminada la etapa metodológica se procede a pasar las tareas concernientes a esta a la columna de terminadas y actualizar el tablero Kanban con las tareas pertinentes a la codificación de la aplicación pasando estas a la columna “En Progeso”.

La Figura 2.21 presenta la actualización del tablero Kanban pasando las tareas relacionadas a la implementación pasarán a la sección en progreso.

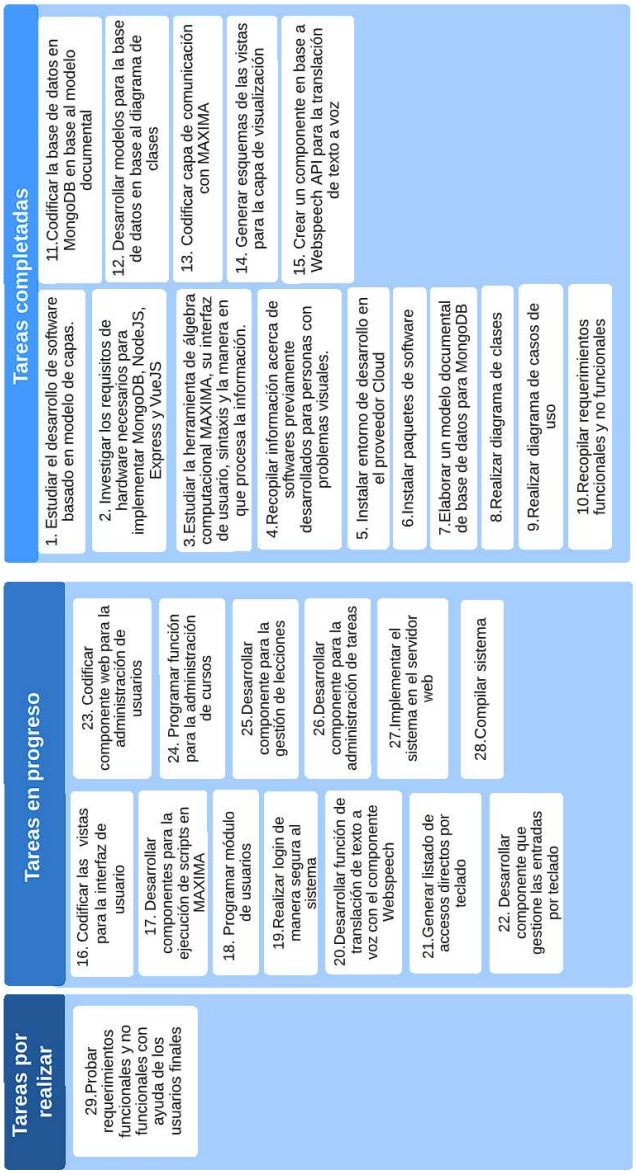


Figura 2.21. Actualización de tareas del tablero Kanban relacionadas a la implementación.

2.3.2 INSTALACIÓN Y CODIFICACIÓN DEL SERVIDOR EN AWS

En el presente Apartado se detallará la información pertinente a la instalación y configuración del sistema prototipo en un ambiente cloud; se describe a grandes rasgos su instalación, respectiva configuración y despliegue.

a) Creación de una instancia de EC2 en AWS

El uso de un servicio en la nube para el alojamiento del sistema nos permite su despliegue sin la necesidad de contar con un servidor dedicado, en este caso se ha utilizado los servicios de *AWS* [35], como primer requisito se debe registrar un usuario en su web oficial, como se aprecia en la Figura 2.22.

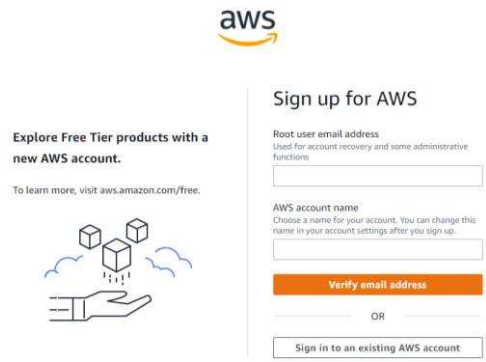


Figura 2.22. Formulario de registro en AWS.

Una vez el usuario se ha registrado tendrá acceso a los servicios de AWS, contando con una consola de administración como se aprecia en la Figura 2.23.

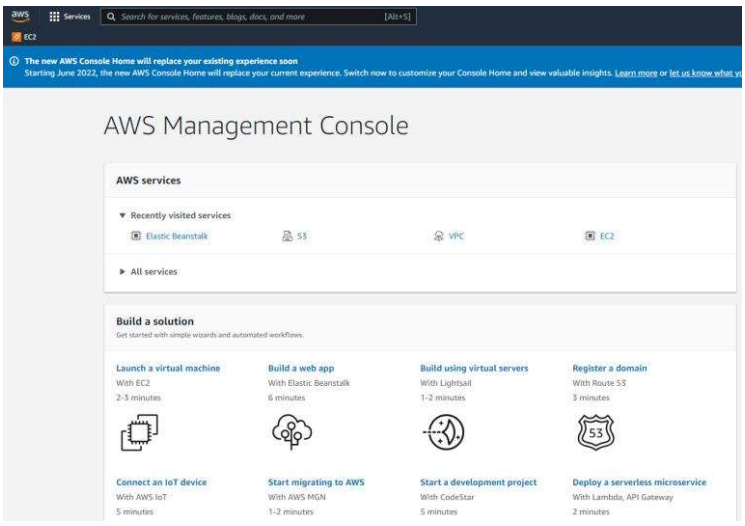


Figura 2.23. Consola de Administración de AWS.

Entre los servicios de AWS se encuentra el de IaaS (Infraestructure as a Service), el cual permite la personalización total de los servicios requeridos por nuestro entorno, este servicio se ofrece a través de la capa EC2 [35] el cual consiste en la creación de instancias de máquinas virtuales para el alojamiento de nuestros servicios.

EC2 permite la creación de instancias con distintos sistemas operativos en función de las necesidades del servicio a desplegarse, en el caso del sistema prototipo se ha seleccionado Ubuntu 20.04 LTS como sistema operativo anfitrión; En la Figura 2.24. se aprecia el lanzamiento de una instancia con el sistema operativo Ubuntu 20.04.

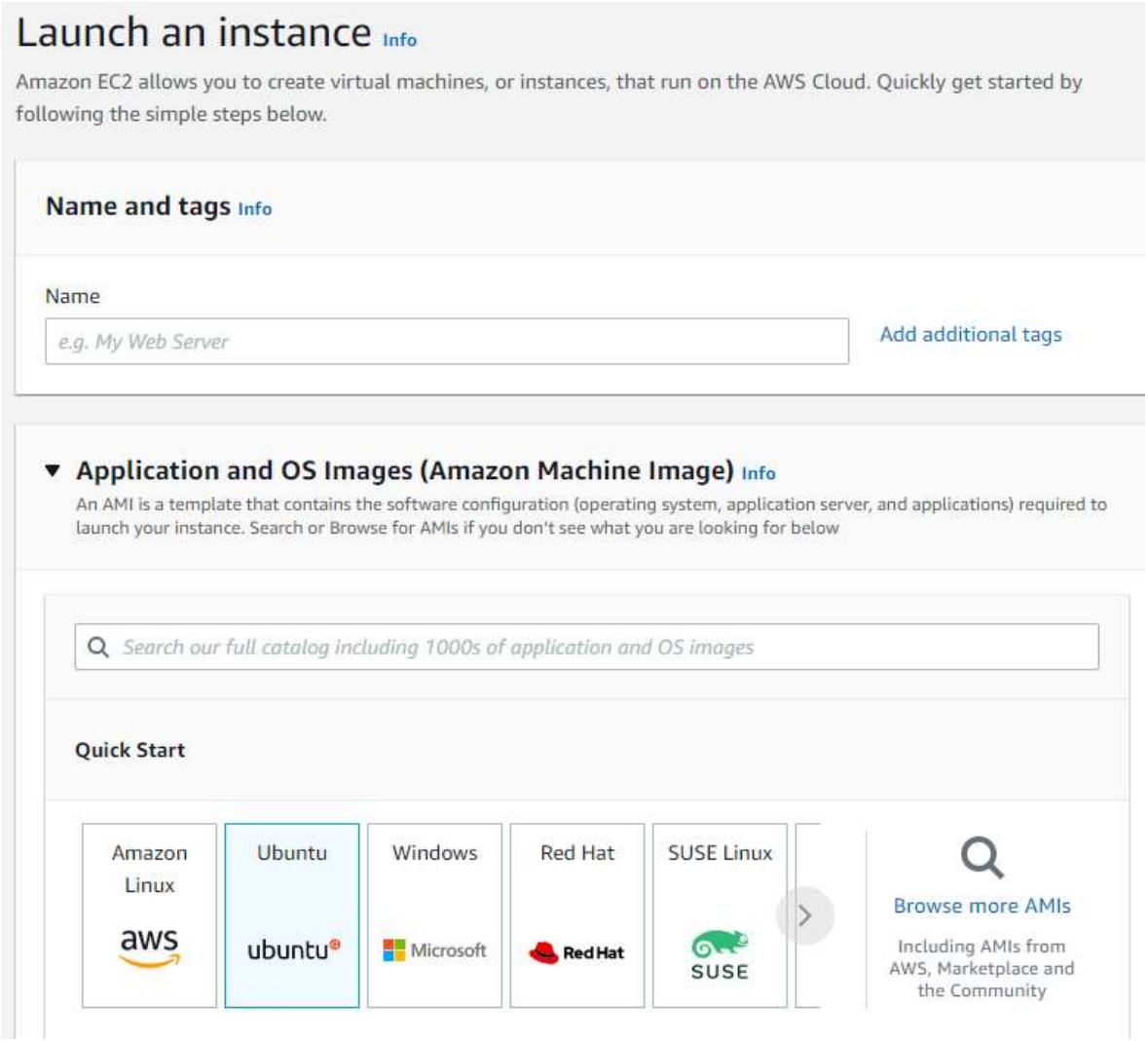


Figura 2.24. Lanzamiento de instancia en AWS EC2.

Una vez lanzada la instancia se tendrá un resumen de sus características y capacidades, como se aprecia en la Figura 2.25.

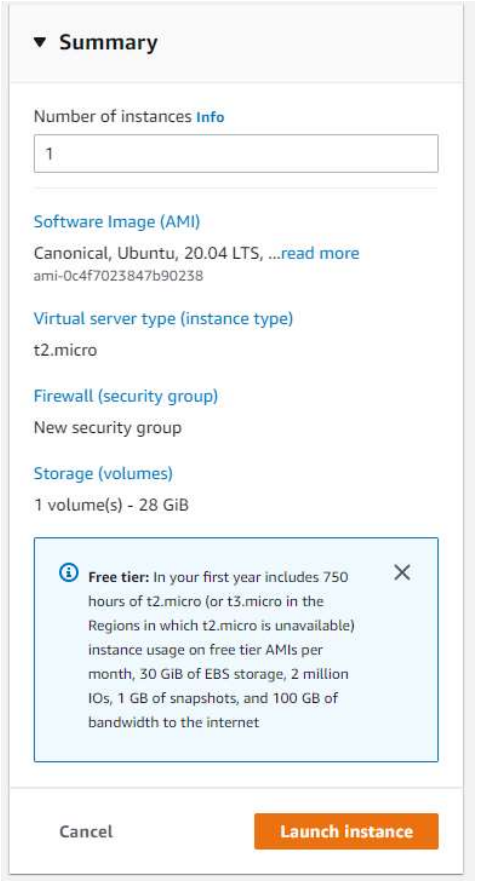


Figura 2.25. Resumen de la creación de instancia en AWS EC2.

Para comprobar que el lanzamiento de la instancia ha sido exitoso se muestra en la Figura 2.26. el resumen proporcionado por la Consola de Administración de EC2.

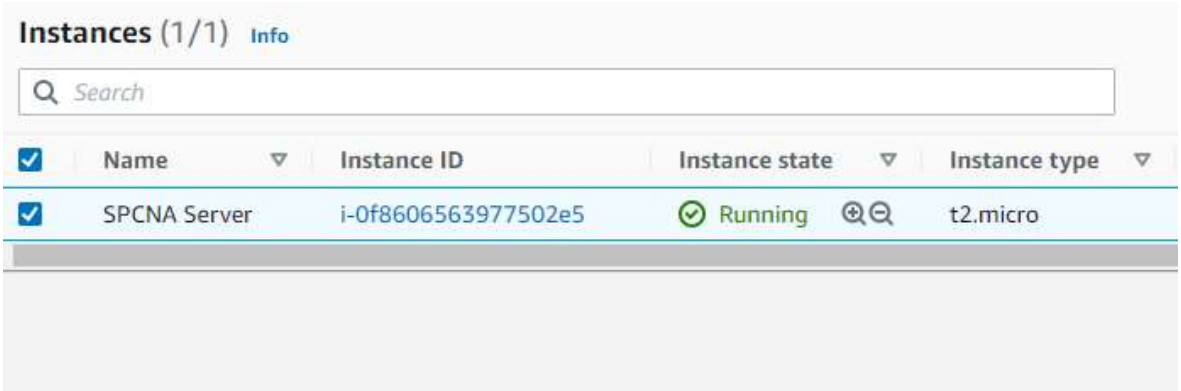


Figura 2.26. Resumen de la Consola de Administración de AWS EC2.

b) Instalación de plataforma de ejecución y herramientas de software

La instalación de la plataforma de ejecución y herramientas de software se ha efectuado a través de una conexión SSH a la consola de comandos del sistema operativo Ubuntu, mediante el software PuTTY [36].

En la Figura 2.27 se muestra los parámetros de conexión SSH con la instancia EC2 de AWS.

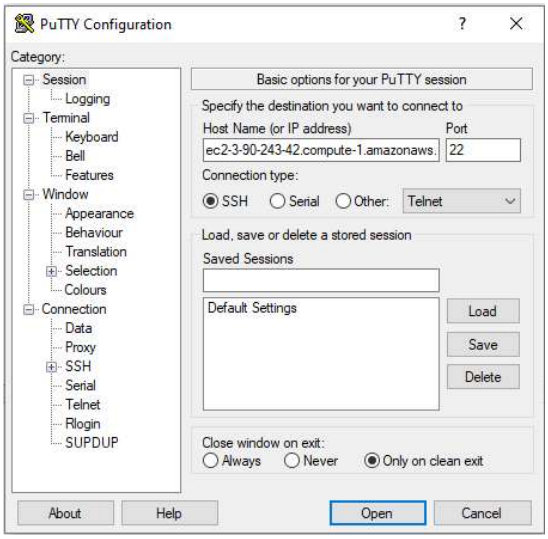


Figura 2.27. Parámetros de conexión SSH hacia instancia EC2 de AWS.

En la Figura 2.28 se muestra la conexión exitosa mediante SSH hacia la instancia EC2 de AWS.

Figura 2.28. Conexión SSH exitosa hacia instancia EC2 de AWS.

En el Código 2.1 se detallan los comandos necesarios para la instalación y configuración del servicio de MongoDB en el sistema operativo Ubuntu [37].

```
1. wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc |
   sudo apt-key add -
2. sudo apt-get install gnupg
3. wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc |
   sudo apt-key add -
4. echo "deb [ arch=amd64,arm64 ]
   https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/5.0
   multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-
   5.0.list
5. sudo apt-get update
6. sudo apt-get install -y mongodb-org
7. sudo systemctl start mongod
8. sudo systemctl enable mongod
```

Código 2.1. Instalación y configuración del servicio de MongoDB.

Al contrario de la instalación clásica de NodeJS mediante APT [38] que instala la última versión disponible de NodeJS, al ser un requisito indispensable que el sistema prototipo utilice la versión 14.5.0 debido a que esta es una versión *LTS*(Long Term Support), lo que quiere decir que contará con actualizaciones de seguridad y soporte de largo plazo; La instalación se realizará mediante NVM [39], el cual es un script bash que se utiliza para administrar múltiples versiones de NodeJS; En el Código 2.2 se muestran los comandos necesarios para la instalación de NodeJS en la instancia EC2 de AWS.

```
1. curl -o- https://raw.githubusercontent.com/nvm-
   sh/nvm/v0.35.3/install.sh
2. curl -o- https://raw.githubusercontent.com/nvm-
   sh/nvm/v0.35.3/install.sh | bash
3. source ~/.bashrc
4. nvm install v14.5.0
5. nvm use v14.5.0
```

Código 2.2. Comandos para instalación de NodeJS.

El sistema de álgebra computacional Maxima se realiza mediante APT con los comandos mostrados en el Código 2.3.

```
1. sudo apt-get install maxima
```

Código 2.3. Comandos necesarios para la instalación de MAXIMA.

Para el desarrollo de la Capa de Presentación se utilizará VueJS, el cual se instalará mediante NPM como se puede apreciar en el Código 2.4.

```
1. npm install -g @vue/cli
2. vue --version
```

Código 2.4. Comandos para instalación de VueJS mediante NPM.

El lenguaje de programación en su versión Python 3 se encuentra instalado por defecto en el sistema operativo Ubuntu 20.04 LTS, sin embargo, el sistema prototipo también utiliza módulos escritos en Python 2. En el Código 2.5 se muestran los comandos necesarios para su instalación.

```
1. sudo apt-add-repository universe
2. sudo apt update
3. sudo apt install python2-minimal
4. curl https://bootstrap.pypa.io/pip/2.7/get-pip.py --output get-
   pip.py
5. sudo python2 get-pip.py
```

Código 2.5. Comandos para la instalación de Python 2.

2.3.3 IMPLEMENTACIÓN DE LA CAPA DE DATOS

El almacenamiento de la información del sistema se alojará en una base de datos no relacional en MongoDB, esta base de datos almacenará objetos de tipo *BJSON* [40] y se almacenarán en colecciones de manera análoga a como se lo realizaría en una tabla en una base de datos relacional. Para ello se han codificado los esquemas correspondientes a cada tipo de objeto con ayuda del middleware [41] Mongoose [42] que hace las veces de ORM [43], lo que nos permite realizar un mapeo desde objetos escritos en el lenguaje de programación JavaScript hacia la base de datos sin necesidad de desarrollar una capa intermedia de conexión. Mongoose es un middleware compatible con Express, por lo cual se creó mediante NPM un proyecto que tiene como base el framework Express.

El Código 2.6 ilustra los comandos necesarios para la creación de un proyecto con el framework Express.

```
1. npm init
2. npm install express mongoose
```

Código 2.6. Comandos para la creación de un proyecto en Express.

En la Figura 2.29 se detalla la estructura básica de directorios y archivos generada al crear un proyecto en Express.

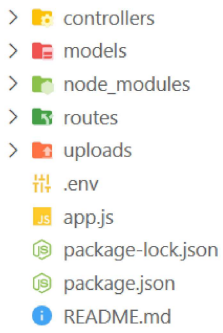


Figura 2.29. Estructura de directorios de un proyecto en Express.

El proyecto consta de los directorios Controllers, Models, Node_modules, Routes y Uploads, que son los encargados de alojar los archivos de script correspondientes a las distintas funcionalidades a desarrollarse; por otra parte, los archivos `.env`, `package-lock.json`, `package.json` y `README.md` son archivos autogenerados cuyo objetivo es almacenar la información correspondiente a la configuración del entorno de ejecución del proyecto; quedando el archivo `app.js` como archivo principal del proyecto y siendo este el punto de entrada a la aplicación, dentro del cual se incluirá la funcionalidad de Mongoose y se realizará la conexión a la base de datos.

El Código 2.7 ilustra las sentencias necesarias para agregar el middleware Mongoose a un proyecto Express y realizar una conexión a una base de datos en MongoDB.

```
1. require('dotenv').config();
2. const express = require("express");
3. const mongoose = require("mongoose");
4.
5. mongoose.connect(process.env.DB_URI, {
6.     useNewUrlParser: true,
7.     useUnifiedTopology: true,
8.     useFindAndModify: false,
9.     useCreateIndex: true,
10.      })
11.      .then(() => console.log("Connected to the database!"))
12.      .catch((err) => console.log(err));
```

Código 2.7. Conexión a la base de datos MongoDB desde Express usando Mongoose.

En base al modelo de base de datos definido de la Sección 2.2.4 se ha codificado en el lenguaje JavaScript los esquemas correspondientes a cada una de las colecciones definidas en el mismo, y que Mongoose se encargará de almacenar en la base de datos de MongoDB.

El Código 2.8 se muestra un ejemplo de las sentencias necesarias para la creación de la colección *Users*. El código correspondiente a las colecciones restantes se presenta en el Anexo B.

```
1.  const mongoose = require("mongoose");
2.  const userSchema = mongoose.Schema({
3.      name:  { type : String , unique : true, required : true },
4.      email: {
5.          type: String,
6.          unique: true,
7.          required: true
8.      },
9.      password:  { type : String , unique : true, required : true
    },
10.     userType:
11.     {
12.         type: Number,
13.         required: true
14.     },
15.     courses:
16.     [{
17.         type: mongoose.Schema.Types.ObjectId,
18.         ref: 'Course',
19.         default: null,
20.     }],
21.     created: {
22.         type: Date,
23.         default: Date.now,
24.     },
25. });
26. module.exports = mongoose.model("User",userSchema);
```

Código 2.8. Código para crear una colección utilizando el Middleware Mongoose.

2.3.4 IMPLEMENTACIÓN DE LA CAPA DE NEGOCIO

Para la implementación de la Capa de Negocio se utilizó el framework Express estudiado en el Capítulo 1 Sección 1.3.2.2. Para la codificación de un servicio web que define una API en base a la arquitectura REST que será la encargada de atender solicitudes por medio del protocolo HTTP para permitir el acceso al cliente a los recursos del sistema.

En la Figura 2.30. se detalla la estructura de directorios de los componentes del sistema codificados para dar funcionamiento al servicio web.

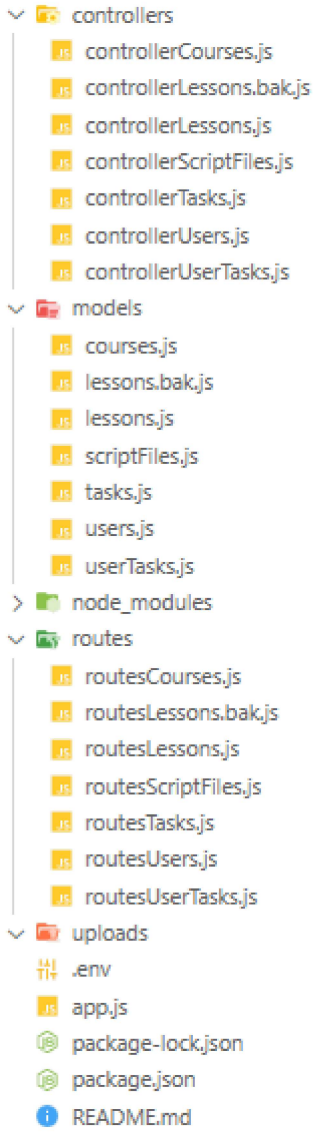


Figura 2.30. Estructura de directorios del servicio web.

Como se aprecia en el Código 2.9. se ha diseñado el servicio web de manera que trabaje con el formato JSON como el dialecto de intercambio de información, de igual manera se han añadido capacidades para el manejo de cabeceras CORS [44] para permitir el acceso a los recursos del sistema por parte de clientes externos a la red local, así como también el uso de cookies para el establecimiento de sesiones temporales entre los clientes y el servicio web.

```
1.  require('dotenv').config();
2.  const express = require("express");
3.  const mongoose = require("mongoose");
4.  const cors = require("cors");
5.  const cookieParser = require('cookie-parser')
6.  const app = express();
7.  const port = process.env.PORT || 5000;
8.
9.  app.use(cors({credentials: true,origin:
    ['http://localhost:8080']}));
10. app.use(cookieParser());
11. app.use(express.json());
12. app.use(express.urlencoded({ extended: true}));
13. app.use(express.static("uploads"));
14.
15. mongoose.connect(process.env.DB_URI, {
16.     useNewUrlParser: true,
17.     useUnifiedTopology: true,
18.     useFindAndModify: false,
19.     useCreateIndex: true,
20. })
21. .then(() => console.log("Connected to the database!"))
22. .catch((err) => console.log(err));
23.
24. app.use("/api/users",require("./routes/routesUsers"));
25. app.use("/api",require("./routes/routesCourses"));
26. app.use("/api/lessons",require("./routes/routesLessons"));
27. app.use("/api/scripts",require("./routes/routesScriptFiles"));
28. app.use("/api/tasks",require("./routes/routesTasks"));
29. app.use("/api/userTasks",require("./routes/routesUserTasks"));
30.
31. app.listen(port, () => console.log(`server running at
    http://localhost:${port}`));
```

Código 2.9. Código para la creación del servicio web.

Al tratarse de una API REST, este servicio web debe de proveer una serie de rutas de acceso a sus distintas funcionalidades. Esto se logra a través de la creación de los archivos de rutas que se han almacenado en la carpeta routes del proyecto principal.

Los archivos de rutas proveen los mecanismos de acceso a las funcionalidades del sistema a través de peticiones HTTP, haciendo uso de los métodos GET, POST, PATCH y DELETE, según sea la necesidad del caso.

El acceso a los datos de cada colección desarrollada en la Capa de Datos se define en un archivo de rutas único que engloba todos los mecanismos de acceso definidos para la misma. En el Código 2.10 se puede apreciar el contenido del archivo `routesUsers.js`, el cual lista las rutas de acceso, sus parámetros y los métodos HTTP correspondientes para su utilización.

```
1. const express = require("express");
2. const router = express.Router();
3. const controllerUsers =
   require("../controllers/controllerUsers");
4.
5. router.get("/getAll",controllerUsers.fetchAllUsers);
6. router.get("/getAlumnos",controllerUsers.fetchAllAlumnos);
7. router.get("/get",controllerUsers.getUser);
8. router.get("/:id",controllerUsers.getUserById);
9. router.post("/logout",controllerUsers.logout);
10. router.post("/",controllerUsers.createUser);
11. router.post("/login",controllerUsers.login);
12. router.patch("/:name",controllerUsers.updateUser);
13. router.delete("/:name",controllerUsers.deleteUser);
14.
15. module.exports = router;
```

Código 2.10. Contenido del archivo routesUsers.js.

Sin embargo, a pesar de existir los archivos de rutas para definir los mecanismos de acceso, estos no definen el comportamiento del sistema ante una petición, sino más bien ceden este cometido a un archivo enlazado. Este tipo de archivo se conoce como controlador y define una serie de métodos a ejecutarse cada vez que se haga una petición listada en el archivo de rutas. Estos archivos se han almacenado dentro del directorio *controllers* en el proyecto principal.

El Código 2.11 presenta el contenido del archivo `controllerUsers.js`, el cual es el encargado de atender las peticiones realizadas al servicio web por medio de las rutas definidas en el archivo *routesUsers.js*.

```
1. const User = require("../models/users");
2. const bcrypt = require("bcryptjs");
3. const jwt = require("jsonwebtoken");
4. module.exports = class controllerUsers {
5.     //fetch all Users
6.     static async fetchAllUsers(req,res){
7.         try {
8.             const users = await User.find().populate('courses');
9.             res.status(200).json(users);
10.              } catch(err) {
11.                  res.status(404).json({ message: err.message});
12.              }
13.          }
14.          //create a User
15.          static async createUser(req,res){
16.              const salt = await bcrypt.genSalt(10);
17.              const hashedPassword = await
   bcrypt.hash(req.body.password,salt);
18.
19.              const user = new User({
20.                  name: req.body.name,
21.                  email: req.body.email,
22.                  password: hashedPassword,
23.                  userType: req.body.userType
24.              })
25.              try {
26.                  const result = await user.save();
27.                  const {password, ...data} = await re-
   sult.toJSON();
28.
29.                  res.status(201).send(data);
30.              } catch(err) {
31.                  res.status(400).json({ message: err.message});
32.              }
33.          }
```

Código 2.11. Código del archivo controllerUsers.js.

Para el resto de los archivos controladores y de rutas del servicio web se ha definido un documento en formato OpenAPI [45] para listar todas la rutas de acceso y sus correspondientes métodos y parámetros que será descrito en el Anexo C.

2.3.5 IMPLEMENTACIÓN DE LA CAPA DE COMUNICACIÓN CON MAXIMA

Para el desarrollo de la Capa de Comunicación con MAXIMA se ha utilizado el lenguaje de programación Python, el cual mediante el framework Flask [46] permite desplegar un servicio web como API REST, el cual atenderá a las peticiones de procesamiento matemático y translación a texto en lenguaje natural.

2.3.5.1 Implementación del Servicio de Procesamiento y Translación.

El servicio de procesamiento y translación se basa en dos módulos externos, el primero llamado SnuggleTex, el cual se encarga de la translación de la salida procesada por MAXIMA hacia el lenguaje de etiquetado CMATHML [47] Y el segundo llamado TexToEs, el cual es una utilidad escrita en Python 2 que se encarga de producir salidas a lenguaje natural tomando como entrada un fragmento de código CMATHML.

El Código 2.12 detalla la implementación de SnuggleTex bajo el framework Spring de Java, el cual se ejecuta en el puerto 3000 y será el encargado de la translación de la salida de MAXIMA hacia CMATHML.

```
1.  @PostMapping(value = "/procesarLatex", consumes = "application/js
    produces = "application/json")
2.        public Resultado procesarLatex(@RequestBody Peticion petici
    throws IOException {
3.
4.              Resultado resultado = new Resultado();
5.              String input = peticion.getEntrada();
6.
7.              SnuggleEngine engine = new SnuggleEngine();
8.
        engine.addPackage(UpConversionPackageDefinitions.getPackage
9.
10.             SnuggleSession session = engine.createSession();
11.
12.             session.parseInput(new SnuggleInput(input));
13.             UpConvertingPostProcessor upConverter = new
    UpConvertingPostProcessor();
14.
15.             XMLStringOutputOptions xmlStringOutputOptions = new
    XMLStringOutputOptions();
16.             xmlStringOutputOptions.addDOMPostProcessors(upConvert
17.             xmlStringOutputOptions.setIndenting(true);
18.             xmlStringOutputOptions.setUsingNamedEntities(true);
19.
20.             String result =
    session.buildXMLString(xmlStringOutputOptions);
21.             result = result.replace("\n", "");
22.             String patternString = "<annotation-xml encoding=\"Ma
    Content\">.*</annotation-xml>";
23.             Pattern pattern = Pattern.compile(patternString,
    Pattern.CASE_INSENSITIVE);
24.             Matcher matcher = pattern.matcher(result);
25.
26.             if (matcher.find()) {
27.                   result = matcher.group(0);
28.             } else {
29.                   System.out.println("NO MATCH");
30.             }
31.
32.             result = result.replace("<annotation-xml encoding=\"M
    Content\">", "<math>");
33.             result = result.replace("</annotation-xml>", "</math>
34.
35.             System.out.println("Up-Conversion process generated:
    result);
```

Código 2.12. Código para conversión de MAXIMA a CMATHML con SnuggleTex.

El Código 2.13 muestra la creación del servicio web encargado de la atención a solicitudes de translación.

```
1. import socket
2. import sys
3. import json
4. import time
5. import os
6. from flask_cors import CORS
7. import signal
8. import shlex, subprocess
9.
10.     import requests
11.
12.     from flask import Flask,
   jsonify,request,make_response
13.     maximaPort=10000
14.     serverPort=8000
15.     app = Flask(__name__)
16.     CORS(app)
```

Código 2.13. Código correspondiente a la creación del servicio de translación.

El servicio de procesamiento y translación se compone por varios métodos encargados de las distintas etapas de procesamiento, estos se encargan de realizar las tareas especificadas en la Figura 2.9.

En el Código 2.14 se detalla el método wrapScript, el cual se encarga envolver en el tag “tex()” cada una de las líneas del script de entrada, su objetivo es solicitar a MAXIMA el procesamiento de los comandos del script de entrada y devolver su resultado en formato LATEX, el cual posteriormente será procesado para su translación a texto en lenguaje natural.

```
1. def wrapScript(expresiones):
2.     expWrapped=[];
3.     for exp in expresiones:
4.         expWrapped.append("tex("+exp+");")
5.     return expWrapped
```

Código 2.14. Código del método wrapScript.

Una vez se haya envuelto las entradas del script, se procederá a procesarlas, para ello en el Código 2.15. se evidencia las sentencias necesarias para esta labor, su propósito es iniciar un servidor socket y lanzar una instancia de MAXIMA en modo cliente, por lo cual se puede interactuar programáticamente con MAXIMA y procesar el script sin necesidad de intervención del usuario.

```
1. def procesarMaxima(expresiones):
2.     sock = socket.socket(socket.AF_INET,
   socket.SOCK_STREAM)
3.     server_address = ('localhost', maximaPort)
4.     sock.bind(('',0))
5.
6.     portNumber=sock.getsockname()[1]
7.     # Listen for incoming connections
8.     sock.listen(1)
9.     cont = 0
10.          aux = 0
11.          respuesta = ""
12.          maximapid =
   os.spawnlp(os.P_NOWAIT,"maxima","maxima","--server=" +
   str(portNumber))
13.          connection, client_address = sock.accept()
14.          connection.settimeout(0.1)
```

Código 2.15. Código del método procesarScript.

Una vez se haya realizado el procesamiento de MAXIMA, los resultados contendrán metadata adicional que no son de carácter relevante para el usuario no vidente, por lo cual se debe eliminar, para lo cual se utiliza expresiones regulares como se evidencia en el Código 2.16 las cuales generan los patrones necesarios para realizar esta eliminación.

```
1. def regexMetadataEliminarMaxima(entrada):
2.
   respuestaReg=re.sub(r"pid=\d+\n.*\n.*\n.*\n.*\n","",entrada)
3.
   listaRespuestas=re.findall(r"^\(%i\d+\).+?(?=\([%])",respuestaRe
   g, re.MULTILINE | re.DOTALL | re.UNICODE)
4.     for i in range(0,len(listaRespuestas)):
5.
   listaRespuestas[i]=re.sub(r"^\(%[io]\d+\)\s","",listaRespuestas[
   i])
6.         listaRespuestas[i]=listaRespuestas[i].replace("\n","")
7.     return listaRespuestas
```

Código 2.16. Código de eliminación de metadatos generados por MAXIMA.

El resultado del procesamiento de MAXIMA con los metadatos eliminados será esencialmente código Latex que presenta los resultados del procesamiento de las operaciones matemáticas, para un usuario común este sería suficiente para conocer el resultado de lo que se deseaba procesar, sin embargo, para un usuario no vidente el uso de un lenguaje complejo como Latex no es posible, es por ello que se transformará este código Latex a texto en lenguaje natural, para este propósito, el Código 2.17 describe el proceso de conversión del mismo.

```
1. def latexToEs(entrada):
2.     latexTxt  = entrada
3.     url = 'http://localhost:8080/snuggle/procesarLatex'
4.     myobj = {'entrada': latexTxt}
5.     response = requests.post(url, json = myobj)
6.     salida=response.json()
7.     cmathml=salida['salida']
8.     result = subprocess.run(['python2', 'module.py' ,'--
   cmathml',cmathml], stdout=subprocess.PIPE)
9.     respuesta=result.stdout.decode("utf-8")
10.         return respuesta
```

Código 2.17. Código para la conversión de Latex.

Todo el proceso debe seguir una secuencia, la cual debe ser comandada por una función adicional, esta se describirá en el Código 2.18, que es la encargada de recibir la petición de procesamiento de Scripts por medio de una solicitud HTTP Post.

```
@app.route('/procesarScript', methods=['GET', 'POST'])
def procesar():
    if request.method == 'POST':

        scriptTxt  = request.form.get('scriptTxt')
        scriptTxt=translateScript(scriptTxt)
        scriptTxt=scriptTxt+"\n"
        scriptTxt=scriptTxt.replace("\r","")
        expresiones=scriptTxt.split("\n")
        entradaNoEcho=expresiones[:-1]
        echoExpresiones=echoEntrada(expresiones)
        expresiones=wrapScript(expresiones)
        echoExpresiones=wrapScriptEntrada(echoExpresiones)
        respuesta=procesarMaxima(expresiones)
        respuestaEchoEntrada=procesarMaximaEntrada(echoExpresiones)
        listaRespuestas=regexMetadataEliminarMaxima(respuesta)
        lrEchoEntrada=regexMetadataEliminarMaxima(respuestaEchoEntrada)
        echoSpeech=echoToSpeech(lrEchoEntrada,entradaNoEcho)
        salidaSpeech=salidaToSpeech(listaRespuestas)
respuestaJson={"entradaScript":entradaNoEcho,"salida":listaRespuestas,"e
ntradaSpeech":echoSpeech,"salidaSpeech":salidaSpeech}
        response = make_response(
            jsonify(
                respuestaJson
            ),
            200,
        )
        response.headers["Content-Type"] = "application/json"
        return response
    else:
        pass
```

Código 2.18. Código final de procesamiento de Script.

2.3.5.2 Implementación del Componente de Audio

El Componente de Audio ha sido diseñado utilizando el API Web Speech, mismo que se basa en JavaScript para su programación, sin embargo, para su implementación en el sistema prototipo se han reinterpretado sus métodos para acoplarlos al framework VueJS, mismo que establecerá la capacidad de ofrecer el servicio de síntesis de voz como un componente global dentro del prototipo.

Para este cometido se ha implementado un componente llamado VoiceComponent.vue que contiene la estructura HTML y el código JavaScript necesario para su funcionamiento.

A continuación, se detallará la estructura del componente:

En el Código 2.19. se detalla la sección Template del componente, misma que permite declararlo y establecer sus argumentos de entrada.

```
1. <template>
2.     <div>
3.
4.       <vue-web-speech-synth
5.       v-model="dataReproducir"
6.       :voice="synthVoice"
7.       :rate="0.7"
8.       @list-voices="listVoices"
9.     />
10.    </div>
11. </template>
```

Código 2.19. Sección Template del archivo VoiceComponent.vue.

Como se evidencia en el Código 2.19. para la creación del componente se debe establecer una etiqueta HTML <vue-web-speech.synth>, la cual lo instanciará, posteriormente se detallan los argumentos del componente:

- v-model: Se refiere al texto a trasladar a voz.
- Voice: Indica la voz que se utilizará para la síntesis, en el caso del sistema prototipo se utilizará la voz Español-MX.
- Rate: Indica la velocidad de reproducción de la síntesis de voz.
- List-voices: Indica una lista de voces adicionales en caso de que la voz predeterminada del sistema no esté disponible en el navegador utilizado como cliente.

La sección Script del archivo VoiceComponent.vue se estructura a su vez por varias subsecciones, siendo estas Props, Computed, Data, Watch y Methods, las cuales se detallarán a continuación:

En el Código 2.20. se detalla la subsección Props del archivo VoiceComponent.vue, misma que define los argumentos de entrada a recibirse en la instanciación del componente.

```
1. props:
2.   {
3.     texto:{
4.       required: true
5.     },
6.     reproducir:{
7.       required: true
8.     }
9.   },
```

Código 2.20. Subsección Props del archivo VoiceComponent.vue.

El Código 2.21. establece la subsección Computed del archivo VoiceComponent.vue, la cual se encarga de computar los cambios realizados a los argumentos de entrada del componente.

```
1. computed:{
2.         authenticated() {
3.         return this.$store.state.authenticated;
4.         },
5.         userType() {
6.         return this.$store.state.userType;
7.         },
8.         selectedCourse() {
9.         return this.$store.state.selectedCourse;
10.             },
11.             currentUser() {
12.             return this.$store.state.currentUser;
13.             },
14.             voiceList() {
15.             return this.$store.state.voiceList;
16.             },
17.             synthVoice() {
18.             return this.$store.state.synthVoice;
19.             },
20.         },
```

Código 2.21. Subsección Computed del archivo VoiceComponent.vue.

La sintaxis presente en el Código 2.22 detalla la subsección Data del archivo VoiceComponent.vue, esta establece las variables internas a utilizarse dentro del componente.

```
data(){
        return {
            voicesSpeech: [],
            play: false,
            synthText: 'Inicio sintesis de voz',
            dataTexto:this.texto,
            dataReproducir:this.reproducir,
        }
    }
```

Código 2.22. Subsección Data del archivo VoiceComponent.vue.

El Código 2.23 detalla la subsección Watch, misma que se encarga de vigilar los cambios realizados a las variables internas del componente y notificarlas hacia los componentes padres del mismo mediante el método $emit.

```
watch:
  {
    dataTexto () {
       this.$emit('onTexto', this.dataTexto)
     },
     dataReproducir (newValue) {
       this.$emit('onReproducir', this.dataReproducir)
     },
     texto() {
             this.dataTexto=this.texto
             },
             reproducir() {
             this.dataReproducir=this.reproducir
             },

       }
```

Código 2.23. Contenido de la subsección Watch del archivo VoiceComponent.vue.

A continuación, se describe la subsección Methods del archivo VoiceComponent.vue, la cual describe una serie de métodos públicos del componente disponibles para ser llamados

desde cualquier otra parte del sistema. Sin embargo, al ser de gran importancia para la utilización del componente de voz estos se detallarán uno a uno

En el Código 2.24 se describe el código necesario para el método quickSpeak, mismo que cumple la función de realizar la síntesis de voz con el menor retraso de procesamiento posible.

```
1. methods:{
2.       quickSpeak(texttospeak) {
3.         if (texttospeak !== "") {
4.           var utterThis = new
   SpeechSynthesisUtterance(texttospeak);
5.           utterThis.voice = this.synthVoice;
6.           utterThis.rate = 0.75;
7.           utterThis.pitch = 1;
8.           var synth = window.speechSynthesis
9.           synth.speak(utterThis);
10.             }
11.           },
12.
```

Código 2.24. Método quickSpeak.

El método startSpeak se encuentra detallado en el Código 2.25. este será el encargado de comenzar a síntesis de voz una vez se hayan provisto todas las órdenes necesarias a la cola de procesamiento.

```
1. startSpeak()
2.  {
3.      this.dataReproducir=true
4.  }
```

Código 2.25. Método startSpeak.

Para el control de flujo de la síntesis de voz también se ha agregado el método stopSpeak, el cual es el encargado de detener completamente la síntesis de voz y eliminar la cola de procesamiento, este se encuentra detallado en el Código 2.26.

```
1. stopSpeak()
2.  {
3.     this.dataReproducir=false
4.  }
```

Código 2.26. Método stopSpeak.

Además de los métodos de control de flujo de la síntesis de voz detallados en el Apartado 2.2.6.2 se ha desarrollado uno adicional que permite no solo la pausa o reanudación de la síntesis de voz, sino que también se ha añadido la posibilidad de reiniciar la cola de procesamiento, misma que es útil para no tener que volver a escuchar una serie de comandos de voz sin necesidad de eliminar e instanciar una nueva cola de procesamiento, este método se detalla en el Código 2.27.

```
1. restartSpeak()
2.  {
3.      var synth = window.speechSynthesis;
4.      synth.cancel()
5.  }
```

Código 2.27. Método restartSpeak.

De misma forma que la detallada en el Apartado 2.2.6.2 se han agregado los métodos de control de flujo estándar de Web Speech Api, como son pausar y reanudar, estos se detallan en los Códigos 2.28 y 2.29.

```
1. pauseSpeak()
2.  {
3.      var synth = window.speechSynthesis;
4.      synth.pause();
5.  }
```

Código 2.28. Método pauseSpeak.

```
1. resumeSpeak()
2. {
3.     var synth = window.speechSynthesis;
4.     synth.resume();
5. }
```

Código 2.29. Método resumeSpeak.

Por último, el Código 2.30. detalla método listVoices, el cual es el encargado de listar las voces disponibles en el navegador que está ejecutando el sistema y establecer la voz predeterminada según el identificador provisto por el mismo.

```
1.  listVoices (list) {
2.        if(this.voiceList)
3.        {
4.          if(this.voiceList.length<1)
5.          {
6.                this.$store.dispatch("setVoiceList", list)
7.          }
8.        }
9.        else
10.       {
11.         this.$store.dispatch("setVoiceList", list)
12.       }
13.       if(this.synthVoice!=null)
14.       {
15.             if(this.voiceList.length>1)
16.             {
17.                   if(this.$browserDetect.isEdge)
18.                   {
19.                     this.$store.dispatch("setSynthVoice",
    this.voiceList.find(element=>element.lang.includes("es-
    MX")==true))
20.                   }
21.                   if(this.$browserDetect.isChrome)
22.                   {
23.                     this.$store.dispatch("setSynthVoice",
    this.voiceList.find(element=>element.lang.includes("es-
    ES")==true))
24.                   }
25.                   if(this.$browserDetect.isFirefox)
26.                   {
27.                     this.$store.dispatch("setSynthVoice",
    this.voiceList.find(element=>element.lang.includes("Spanis
    h (Latin America)")==true))
28.                   }
29.             }
30.       }
31. },
32. },
```

Código 2.30. Método listVoices.

Una vez diseñado el componente de voz este no puede trabajar por sí mismo, es por ello que debe ser instanciado en un componente de mayor orden y a su vez realizada la invocación de sus métodos.

El Código 2.31. detalla el procedimiento utilizado para la importación e instanciación del componente.

```
1. import VoiceComponent from
   '/src/components/VoiceComponent'
2. export default {
3.   props:
4.   {
5.     fileInput:{
6.       required: false
7.     },
8.   },
9.   components: {
10.    VoiceComponent: VoiceComponent,
11.  },
```

Código 2.31. Importación e instanciación del VoiceComponent.

A su vez, los métodos del componente deben ser invocados haciendo referencia a este, en el código 2.32 se detalla una sección de código que hace uso intensivo del método quickSpeak del componente de voz.

```
1. for (var i = 0; i < salidaCurrentCell['salida'].length; i++) {
2.   console.log("Execute For")
3.   this.$refs.componenteSpeak.quickSpeak("Celda
   "+(this.currentCell+1)+" Entrada "+(i+1))
4.   this.$refs.componenteSpeak.quickSpeak(auxEntradaScript)
5.
6.   this.$refs.componenteSpeak.quickSpeak("Celda
   "+(this.currentCell+1)+" Salida "+(i+1))
7.   this.$refs.componenteSpeak.quickSpeak(auxSalidaScript)
```

Código 2.32. Llamado y uso del método quickSpeak.

2.3.6 IMPLEMENTACIÓN DE LA CAPA DE PRESENTACIÓN

Se ha utilizado el framework VueJS para la codificación de la Capa de Presentación, en base a los bosquejos de la aplicación realizados en el Capítulo 2 Apartado 2.2.7.1 se ha codificado las vistas de la aplicación.

La sintaxis de VueJS indica que para la creación de cada ventana se deben incluir tres secciones distintas de código, Template, Script y Style, a continuación, se detallará como ejemplo el código necesario para la creación de la ventana Login almacenado en el archivo Login.vue.

El Código 2.33 detalla la sintaxis necesaria para la creación de la estructura de la ventana, misma que se encuentra en la sección Template del archivo.

```
1.  <template>
2.    <div id="login" class="home">
3.      <div class="container">
4.        <div class="block">
5.          <div class="columns">
6.            <div class="column is-4"></div>
7.            <div class="column">
8.              <section class="section">
9.                <form @submit.prevent="submit">
10.                 <b-field label="Usuario">
11.           <b-input v-model="username" maxlength="30"></b-input>
12.                 </b-field>
13.                 <b-field label="Contraseña">
14.               <b-input v-model="password" type="password"></b-input>
15.             </b-field>
16.             <b-button @click="submit" label="Iniciar Sesión" />
17.              </form>
18.           </section>
19.         </div>
20.       </div>
21.     </div>
22.   </div>
23. </template>
24.
```

Código 2.33. Sección Template del archivo Login.vue.

La sección Script del archivo Login.vue se detallará en el Código 2.34 y es esta la que se encarga de proveer la lógica necesaria para la interacción con el usuario por parte del código HTML correspondiente a la sección Template.

```
<script>
export default {
  data() {
    return {
      username: "",
      password: "",
      loginFailed: false,
      validatePassword: false,
      validateUsername: false,
          };
      },
      methods: {
        submit() {
          fetch("http://localhost:5000/api/users/login", {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            credentials: "include",
            body: JSON.stringify({ name: this.username,
  password: this.password }),
          })
            .then((response) => response.json())
            .then((data) => {
              var resp = data.message;
              console.log(resp);
              if (resp == "success") {
                this.authLogin();
                this.$router.push("/");
              } else this.loginFailed = true;
            });
        },
      },
    };
    </script>
```

Código 2.34. Sección Script del archivo Login.vue.

La parte estética de la estructura HTML se detallará en la sección Style del archivo Login.vue, esta será descrita en el Código 2.35.

```
<style>
.textinput {
  float: left;
  width: 100%;
  min-height: 75px;
  outline: none;
  resize: none;
  border: 1px solid grey;
  font-size: 18pt;
    }
    .el-row {
      margin-bottom: 20px;
    }
    .el-col {
      border-radius: 4px;
    }
    .bg-purple-dark {
      background: #99a9bf;
    }
    .bg-purple {
      background: #d3dce6;
    }
    .bg-purple-light {
      background: #e5e9f2;
    }
    .grid-content {
      border-radius: 4px;
      min-height: 36px;
    }
    .row-bg {
      padding: 10px 0;
      background-color: #f9fafc;
    }
    </style>
```

Código 2.35. Sección Style del archivo Login.vue.

De manera similar se han codificado las ventanas para el resto del sistema, mismas que presentaremos a continuación:

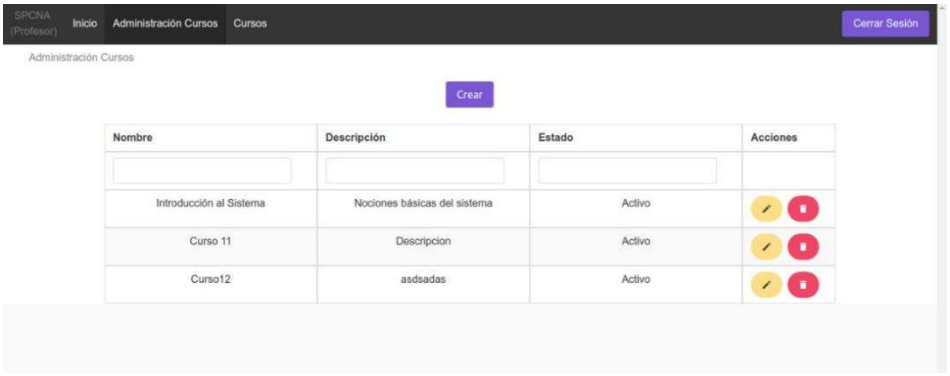


Figura 2.31. Vista de la ventana de administración de cursos.

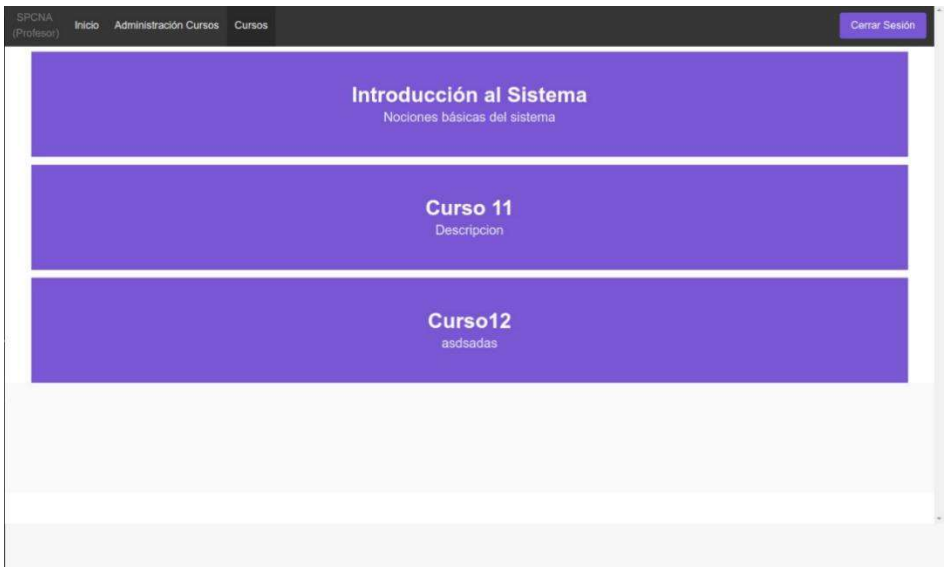


Figura 2.32. Vista de la ventana de selección de cursos.

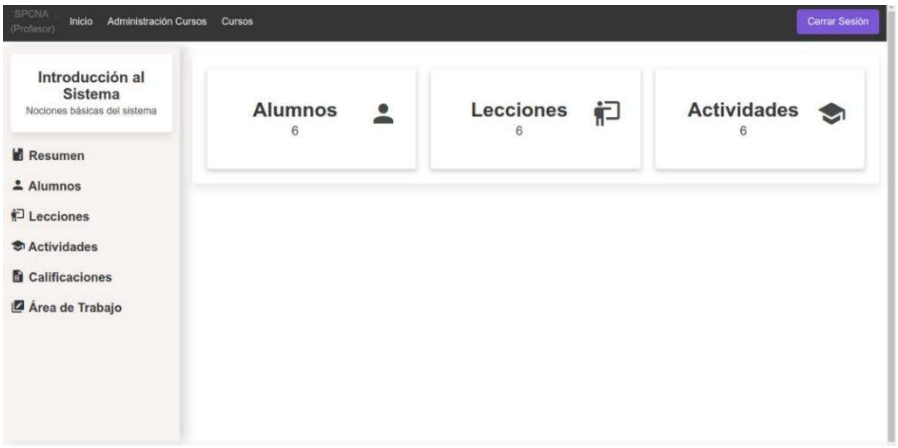


Figura 2.33. Vista de la ventana de resumen de curso.

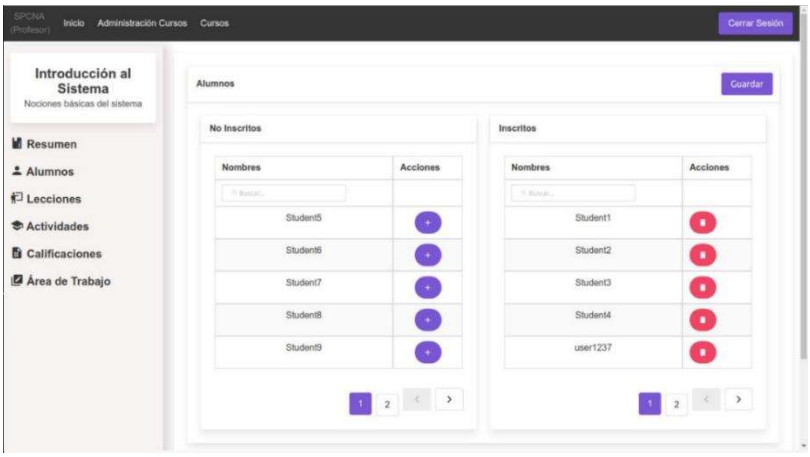


Figura 2.34. Vista de la ventana de matriculación de estudiantes.

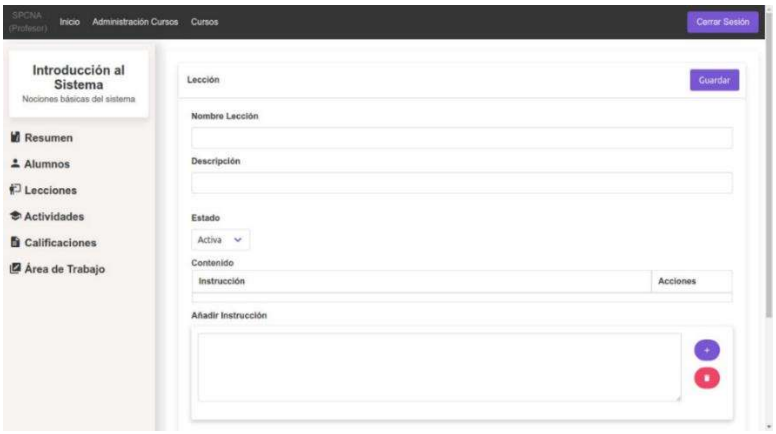


Figura 2.35. Vista de la ventana de creación de lecciones.

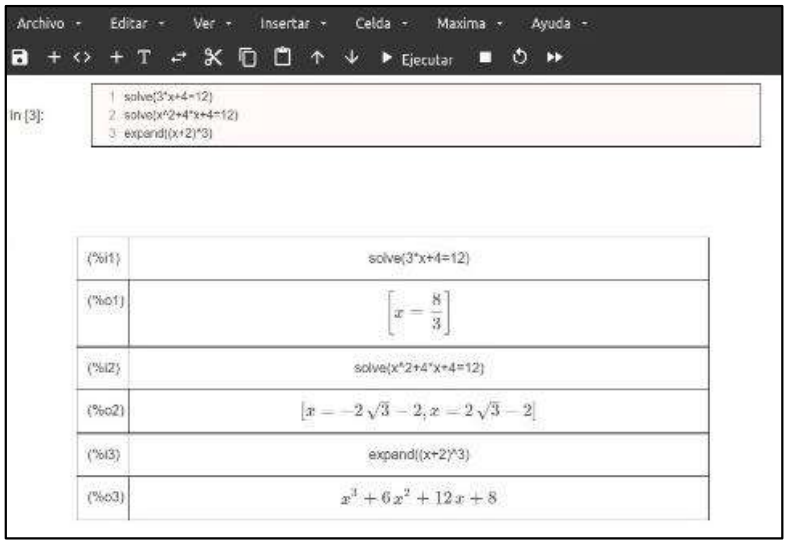


Figura 2.36. Vista de la ventana del área de trabajo.

3. RESULTADOS Y DISCUSIÓN

El presente capítulo presenta la ejecución de pruebas en cuanto a la implementación del sistema prototipo, las mismas que serán utilizadas para verificar el cumplimiento, tanto de requisitos funcionales, como no funcionales; estos detallados en Capítulo 2, Apartado 2.2.2 y Apartado 2.2.3 respectivamente.

3.1 ACTUALIZACIÓN DEL TABLERO KANBAN

Se actualizará el tablero Kanban, quedando como tarea final la realización de pruebas de requerimientos funcionales y no funcionales. Una vez finalizada esta tarea, se dará por terminado el tablero Kanban.

En la Figura 3.1 se puede apreciar el tablero Kanban con la tarea final en progreso.

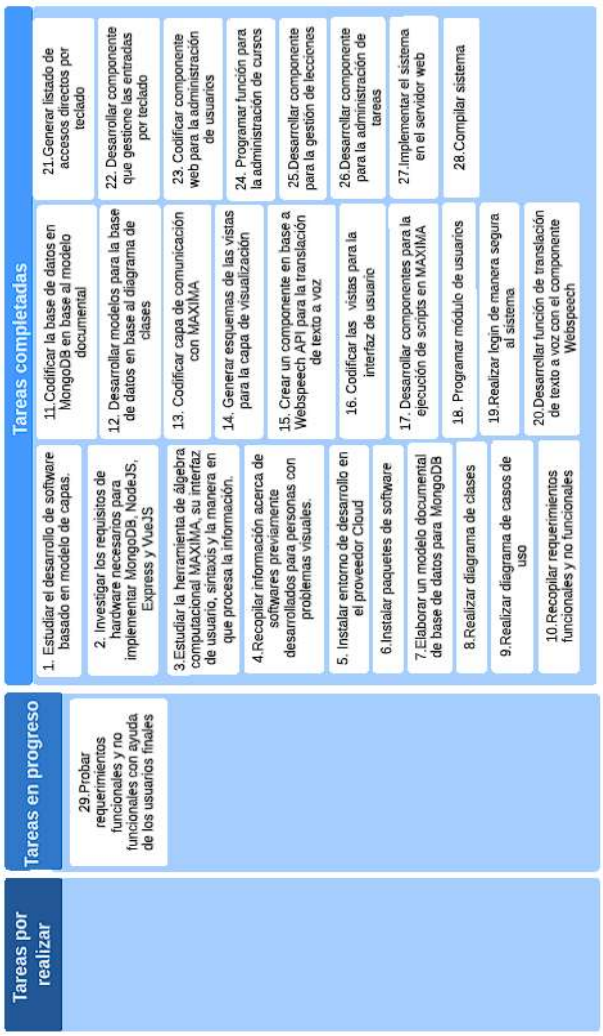


Figura 3.1. Tablero Kanban Previo a la Realización de Pruebas.

3.2 PRUEBAS DE VALIDACIÓN DE REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

3.2.1 DEFINICIÓN DEL ENTORNO DE PRUEBAS

Para la comprobación del funcionamiento del sistema prototipo, se ha procedido a definir un entorno de pruebas controlado comprendido por un equipo de cinco personas que ejercerán como los usuarios objetivo del sistema con distintas funciones cada uno y una serie de condiciones detalladas a continuación:

- Se han designado 3 usuarios a trabajar con el módulo estudiante, y dos usuarios para trabajar con los módulos profesor y administrador.
- Se ha realizado una inducción previa del funcionamiento del sistema para tratar todas sus funcionalidades, sus restricciones, comandos de teclado y acceso a los distintos menús.
- Las pruebas han sido supervisadas para llevar un control de posibles errores o eventos no controlados.
- Los usuarios encargados de los módulos profesor y administrador realizaron una exploración de la interfaz tanto gráfica como auditiva, de forma que puedan evidenciar las funciones de accesibilidad implementadas para dar retroalimentación a los usuarios del módulo estudiante.
- Los usuarios encargados del módulo estudiante realizaron una exploración de la interfaz auditiva con el objetivo de conocer los distintos menús y los respectivos atajos de teclado que permiten su desplazamiento en los mismos.
- Una vez terminadas las pruebas de funcionalidad del sistema, se realizó una encuesta con una serie de preguntas a los usuarios, de manera que ofrezcan una retroalimentación con su experiencia de uso; adicionalmente se realizó una encuesta escrita para llevar un registro de cumplimiento de los requerimientos.
- Se realizaron dos sesiones de pruebas, la primera con el prototipo en marcha en un servidor local para identificar incidencias en la implementación del sistema, y la segunda en el servidor cloud con las incidencias resueltas para la validación de estas correcciones.

3.2.2 VALIDACIÓN DE REQUERIMIENTOS FUNCIONALES

La validación de Requerimientos Funcionales se ha realizado mediante la toma de respuestas a la encuesta de los usuarios tras haber probado las distintas funcionalidades y módulos del sistema prototipo.

En la Tabla 3.1 se detalla el estado de las pruebas de los Requerimientos Funcionales.

Tabla 3.1. Tabla de cumplimiento de Requerimientos Funcionales.

ID	Descripción	Estado	Observaciones
RF1	Retroalimentación audible al usuario no vidente a través de todos los menús.	**Primera Sesión de pruebas:** Parcialmente Cumplido	**Problemática:** Existen menús que no cuentan con una descripción correcta de todas sus funcionalidades
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** Se implementó las descripciones faltantes a los menús, de manera que el usuario no vidente sepa exactamente qué hacer en cada uno de ellos.
RF2	Captura de eventos de teclado pensado en los usuarios no videntes	**Primera Sesión de pruebas:** Parcialmente Cumplido	**Problemática:** Los eventos de teclado no se detectan de manera global en toda la ventana del sistema.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** Se ha implementado un componente global de captura de eventos de teclado.
RF3	Retroalimentación auditiva ante los	**Primera Sesión de pruebas:** Cumplido	**Problemática:** No existe.

	comandos de script ingresados.	**Segunda Sesión de pruebas:** Cumplido	**Medida Correctiva:** No existe necesidad.
RF4	Navegación simple a través de todo el sistema	**Primera Sesión de pruebas:** Parcialmente Cumplido	**Problemática:** La navegación se interrumpe cuando existen actualizaciones forzadas de la página mediante el comando F5.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** Se añadió un componente para el almacenamiento persistente de las variables de trabajo del sistema que llevan control de la ubicación dentro del mismo.
RF5	Funcionalidad de control de la síntesis de voz	**Primera Sesión de pruebas:** Parcialmente Cumplido	**Problemática:** La síntesis de voz se desencadena a causa de eventos de teclado, sin embargo, los eventos de pausa, reanudación y reinicio no funcionan ante eventos de teclado, funcionando únicamente ante eventos visuales.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** Se implementó la funcionalidad de control de la síntesis de voz en el componente global de captura de eventos de teclado.

RF6	Dictado del resultado de operaciones matemáticas en lenguaje natural	**Primera Sesión de pruebas:** Cumplido	**Problemática:** No existe.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** No existe necesidad.
RF7	Menú de ayuda universal accesible a través de teclado.	**Primera Sesión de pruebas:** Parcialmente Cumplido	**Problemática:** Al igual que la captura de eventos de teclado, este solo funciona cuando la ventana tiene el foco del teclado, dejando de funcionar correctamente cuando este foco se pierde, siendo necesario retomar el foco mediante la tecla TAB.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** Se implementó las combinaciones de teclas necesarias para el funcionamiento del menú universal en el componente global de captura de eventos de teclado.

3.2.3 VALIDACIÓN DE REQUERIMIENTOS NO FUNCIONALES

Los requerimientos complementarios al funcionamiento del sistema detallados en los Requerimientos No Funcionales se trataron de manera similar a los Funcionales, siendo evaluados por parte de los usuarios y consultados para su correcta validación.

La Tabla 3.2 detalla el estado de las pruebas realizadas para el cumplimiento de los Requerimientos No Funcionales.

Tabla 3.2. Tabla de cumplimiento de Requerimientos No Funcionales.

ID	Descripción	Estado	Observaciones
RNF1	Interfaz gráfica amigable e intuitiva.	**Primera Sesión de pruebas:** Parcialmente Cumplido	**Problemática:** Existen ciertos menús que no ofrecen ayudas visuales para el acceso correcto a sus funcionalidades.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** Se implementó botones con esquemas de colores que provean ayudas visuales a la ejecución de las funcionalidades de cada menú.
RNF2	Curva de aprendizaje no muy compleja para su utilización	**Primera Sesión de pruebas:** No Cumplido	**Problemática:** Los menús auditivos no ofrecen una descripción suficiente de sus funcionalidades.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** Se implementó un listado de funcionalidades a ser dictadas al ingreso a cada menú.
RNF3	Ejecución fluida del sistema prototipo sin retardos ni repetición de solicitudes	**Primera Sesión de pruebas:** Parcialmente Cumplido	**Problemática:** Existen retardos de procesamiento inherentes a la ejecución de los cálculos matemáticos.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** Se ha reducido los tiempos de espera de las solicitudes de

			cálculo al Componente de Comunicación con MAXIMA
RNF4	Sistema prototipo robusto sin caídas del servicio.	**Primera Sesión de pruebas:** Cumplido	**Problemática:** No existente.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** No existe necesidad.
RNF5	Se debe ofrecer baja latencia entre el sistema y los usuarios finales	**Primera Sesión de pruebas:** Cumplido	**Problemática:** No existente.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** No existe necesidad.
RNF6	Se debe contar con la seguridad necesaria para que solo usuarios autenticados accedan al sistema	**Primera Sesión de pruebas:** Parcialmente Cumplido	**Problemática:** A pesar de existir los componentes de gestión de sesión de usuarios, las cookies almacenadas para esta tarea no se encuentran encriptadas.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** Se añadió un mecanismo de encripción de las cookies del sistema mediante JWT.
RNF7	Se debe proveer fidelidad matemática en el cálculo de resultados	**Primera Sesión de pruebas:** Cumplido	**Problemática:** No existe.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** No existe necesidad.
RNF8	Se debe proveer una tasa de disponibilidad satisfactoria para	**Primera Sesión de pruebas:** Parcialmente Cumplido	**Problemática:** Al encontrarse desplegado en un servidor local el sistema solo estará disponible

	proveer servicio en cualquier momento		cuando este se encuentre encendido.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** Al haberse desplegado en el servicio cloud el sistema se encontrará siempre disponible.
RNF9	Se debe proveer la capacidad de atender a un número grande de usuarios a la vez	**Primera Sesión de pruebas:** Cumplido	**Problemática:** No existe.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** No existe necesidad.
RNF10	Se debe contar con capacidad de escalamiento y posibilidad de actualización.	**Primera Sesión de pruebas:** Cumplido	**Problemática:** No existe.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** No existe necesidad.
RNF11	Se debe lanzar el sistema como una aplicación en estado de producción	**Primera Sesión de pruebas:** No cumplido.	**Problemática:** Al encontrarse en estado de desarrollo el sistema se encuentra en un servidor de pruebas.
		Segunda Sesión de pruebas: Cumplido	**Medida Correctiva:** Se ha desplegado el sistema en modo de producción.

3.2.4 CIERRE DEL TABLERO KANBAN

Una vez que se han realizado las pruebas con los usuarios sobre el sistema prototipo y al aplicar las soluciones necesarias a las incidencias encontradas, se procede a realizar el

cierre del Tablero Kanban, dejando así todas las tareas del mismo como completadas como se observa en la Figura 3.2.

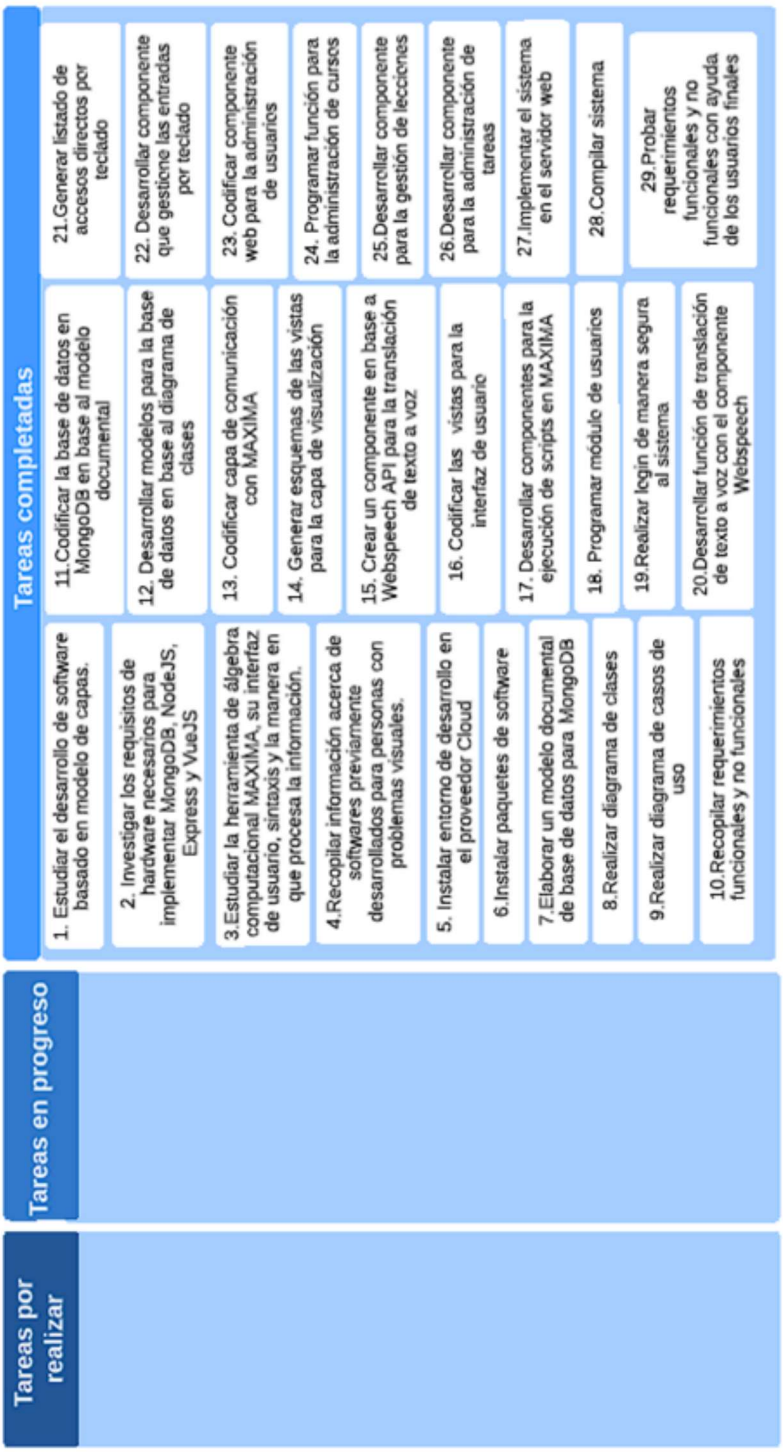


Figura 3.2. Cierre del Tablero Kanban.

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

El presente Trabajo de Titulación se ha desarrollado siguiendo una serie de etapas como son: Metodología, Diseño, Implementación y Pruebas de Funcionamiento; teniendo como resultado un sistema prototipo funcional e implementado en un servicio en la nube, del cual se han obtenido las siguientes conclusiones.

- Al concluir el presente Proyecto de Titulación se ha implementado un Sistema Prototipo de Cálculo Numérico Algebraico Interactivo para estudiantes con discapacidad visual, el cual, a través del uso de tecnologías como *MAXIMA, WebSpeech API y VueJS*, ofrece una herramienta matemática que les permite realizar cálculos numérico-algebraicos por medio de un sistema web interactivo que implementa características de accesibilidad adaptadas a sus necesidades, brindándoles un soporte en su formación académica en áreas de ingeniería.

- Para la realización de este sistema prototipo se ha estudiado una serie de tecnologías, de las cuales se ha adquirido nuevos conocimientos acerca del desarrollo de aplicaciones web interactivas, servicios en la nube, bases de datos documentales, metodologías de desarrollo ágil y síntesis de texto a voz. Además, se han obtenido habilidades de diseño de arquitecturas de software en capas y la implementación de tecnologías en estas, como: *JavaScript, Python, MongoDB y AWS.*

- Tomando en cuenta los Requerimientos Funcionales y No Funcionales planteados para el desarrollo del sistema se ha diseñado una serie de módulos que cumplen de manera satisfactoria los requisitos de accesibilidad impuestos y que permiten el correcto uso del sistema por parte de los estudiantes con discapacidad visual.

- Siguiendo la arquitectura en capas propuesta, se ha diseñado las funcionalidades a contar en cada uno de los módulos especificados en la etapa de Diseño, así como también los roles de usuarios correspondientes a cada módulo, sus funciones y los casos de uso pertinentes a las mismas.

- Mediante las tecnologías propuestas en la etapa de Diseño, y siguiendo la metodología Kanban para organizar de manera eficiente las tareas a realizarse en cada etapa del desarrollo del sistema, se ha codificado y configurado cada uno de

los módulos del sistema para su posterior implementación en un servicio en la nube, siendo en este caso particular *Amazon Web Services.*

- Para el componente de síntesis de texto a voz se ha desarrollado una serie de funcionalidades adicionales que extienden las disponibles en el *API Web Speech*, lo que ayuda a brindar una accesibilidad mejorada y a medida de los estudiantes que hagan uso del sistema, asegurando así que los mismos sientan la comodidad de utilizarlo durante el desarrollo de sus actividades académicas.

- Se realizó una serie de pruebas de funcionalidad y usabilidad, mismas que sirvieron para la comprobación del funcionamiento del sistema mediante pruebas realizadas a un grupo de usuarios objetivos en un ambiente controlado y supervisadas para su correcta consecución, las cuales derivaron en encuestas, cuyos resultados sirvieron para comprobar el correcto cumplimiento de los Requisitos Funcionales y No Funcionales propuestos para el desarrollo del Sistema Prototipo.

4.2 RECOMENDACIONES

En el transcurso del desarrollo del sistema se presentaron distintas eventualidades, las cuales se fueron resolviendo en el transcurso del tiempo, y de las cuales se desprenden las siguientes recomendaciones.

- Previo a la realización de un proyecto de software, es recomendable realizar un estudio previo de las herramientas a utilizarse, teniendo en cuenta los lenguajes de programación, frameworks disponibles y herramientas de desarrollo a utilizarse; asegurando así que estos se adapten a las necesidades del proyecto. También se debe verificar que se cuente con un adecuado soporte técnico a futuro, ya que de este depende la posibilidad de actualización y escalamiento del proyecto.

- Para todo proyecto de software, independientemente de la escala del mismo, es recomendable la utilización de un sistema de control de versiones, esto ayudará a mantener un control exhaustivo del código fuente, así como también almacenar y rastrear sus cambios en las diferentes versiones del proyecto.

- Las bases de datos documentales no guardan por defecto información relacional entre las entidades almacenadas en las mismas a través del esquema tradicional de claves foráneas, es por ello que se sugiere la inclusión de contenedores para

ObjectId en cada entidad, lo que análogamente realizará la misma función en este tipo de bases de datos.

- Es recomendable realizar *mockups* de las interfaces de usuario previos a su codificación, esto ayuda a tener un objetivo al cual se debe llegar durante su desarrollo, lo que agilizará la consecución de este objetivo al tener un esquema claro de las interacciones con el usuario que realizará cada una de las interfaces.

- A pesar de ser posible la realización de un sistema web con JavaScript puro, esto no es recomendable en sistemas de media y larga escala, ya que no existe una base de desarrollo concreta que establezca una arquitectura sólida que agilice el desarrollo, es por ello que se recomienda la utilización de un marco de trabajo con una larga trayectoria, una gran comunidad y en desarrollo activo, como ha sido el caso del presente sistema el uso de *VueJS*.

- Se considera una buena práctica de desarrollo el realizar la documentación del proyecto, para ello se recomienda el uso de herramientas que agilicen este trabajo, como puede ser en este caso el uso de documentos en el formato *OpenAPI* para indicar el funcionamiento de los métodos REST provistos por la capa de negocio del sistema.

Además de las incidencias encontradas durante el desarrollo del sistema, también se propondrán distintas recomendaciones encaminadas a extender el funcionamiento del sistema prototipo como trabajos futuros.

- En el futuro sería de gran utilidad la implementación de funcionalidades de interacción dinámica y en tiempo real con los resultados de las expresiones matemáticas resueltas en el presente sistema prototipo.

- El presente sistema prototipo implementa una síntesis de voz no lineal de las expresiones matemáticas resultantes, sino más bien se implementa una síntesis basada en el lenguaje natural y la notación matemática-algebraica estándar, sin embargo no se cuenta con ayudas auditivas para el estudiante con discapacidad visual, es por ello que se propone realizar un trabajo futuro que implemente ayudas auditivas como pueden ser la inclusión de audio espacial, modificación de la voz de acuerdo al receptor y tonos auxiliares que mejoren la experiencia de uso de la síntesis de voz.

- Se podría trasladar el procesamiento de la síntesis de voz a la parte del servidor y cambiar el motor de síntesis de voz a uno con mayor cantidad de opciones, lo que permitiría dotar a la misma con más funcionalidades de las provistas por la API WebSpeech.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] «Estadísticas de Discapacidad – Consejo Nacional para la Igualdad de Discapacidades». https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/ (accedido 2 de marzo de 2022).

[2] J. C. Ocampo y J. C. Ocampo, «Disability, Inclusiveness and Higher Education in Ecuador: The Case of Universidad Católica de Santiago de Guayaquil», *Revista latinoamericana de educación inclusiva*, vol. 12, n.º 2, pp. 97-114, nov. 2018, doi: 10.4067/S0718-73782018000200097.

[14] S. M. Johnson y J. Jakupovic, «The building of Web Pages», p. 49.

[15] «What Is NoSQL? NoSQL Databases Explained», *MongoDB*. https://www.mongodb.com/nosql-explained (accedido 16 de marzo de 2022).

[16] S. Tilkov y S. Vinoski, «Node.js: Using JavaScript to Build High-Performance Network Programs», *IEEE Internet Computing*, vol. 14, n.º 6, pp. 80-83, nov. 2010, doi: 10.1109/MIC.2010.145.

[17] A. Mardan, *Express.js Guide: The Comprehensive Book on Express.js*. Azat Mardan, 2014.

[18] A. Fern??ndez Montoro, *Python 3 al descubierto*. Madrid: RC Libros, 2012.

[19] «SnuggleTeX - Overview & Features». https://www2.ph.ed.ac.uk/snuggletex/documentation/overview-and-features.html (accedido 16 de marzo de 2022).

[20] R. Miner, «The Importance of MathML to Mathematics Communication», vol. 52, n.º 5, p. 7, 2005.

[21] «Maxima, un sistema de álgebra computacional». https://maxima.sourceforge.io/es/index.html (accedido 16 de marzo de 2022).

[22] A. Kyriakidis, K. Maniatis, y E. You, «The Majesty of Vue.js 2», p. 30.

[23] «Web Speech API - Referencia de la API Web | MDN». https://developer.mozilla.org/es/docs/Web/API/Web_Speech_API (accedido 19 de mayo de 2022).

[40] «BJSON - the binary JSON format». http://bjson.org/ (accedido 8 de junio de 2022).

[41] «¿Qué es el middleware?» https://www.redhat.com/es/topics/middleware/what-is-middleware (accedido 8 de junio de 2022).

[42] «Mongoose ODM v6.3.6». https://mongoosejs.com/ (accedido 8 de junio de 2022).

[43] «Comparing the performance of object databases and ORM tools | Proceedings of the 2006 annual research conference of the South African institute of computer scientists

and information technologists on IT research in developing countries». https://dl.acm.org/doi/abs/10.1145/1216262.1216263 (accedido 8 de junio de 2022).

[44] M. Hossain, *CORS in Action: Creating and consuming cross-origin APIs*. Simon and Schuster, 2014.

[46] M. R. Mufid, A. Basofi, M. U. H. Al Rasyid, I. F. Rochimansyah, y A. rokhim, «Design an MVC Model using Python for Flask Framework Development», en *2019 International Electronics Symposium (IES)*, sep. 2019, pp. 214-219. doi: 10.1109/ELECSYM.2019.8901656.

6. ANEXOS

Anexo A: Historias de Usuario

Anexo B: Código Sistema

Anexo B.1: Código Capa de Datos y Capa de Negocio

Repositorio digital: danilo-pilacuan/SPCNABackend (github.com)

Anexo B.2: Código Capa de Comunicación con MAXIMA

Repositorio digital: danilo-pilacuan/snuggleTexAPI (github.com)

Anexo B.3: Código Capa de Presentación

Repositorio digital: danilo-pilacuan/SPCNAFrontend (github.com)

Anexo C: Especificación API REST Capa de Negocio