



TECNOLÓGICO DE ESTUDIOS SUPERIORES DE COACALCO

ESTANCIAS PROFESIONALES

Alumna:

Montserrat Bedolla Cortés

Matrícula:

201221411

Carrera:

Ingeniería en Tecnologías de la Información y Comunicación

Especialidad:

Desarrollo de aplicaciones móviles

Índice

Resumen.....	5
Agradecimientos	6
CAPÍTULO I	7
Introducción	8
Descripción de la empresa	9
Descripción de la participación activa en proyectos dentro de las líneas de investigación	10
Justificación	10
Objetivo General	11
Objetivos Específicos.....	11
Alcances y limitaciones	12
CAPÍTULO II	13
Marco teórico	14
CAPÍTULO III.....	19
Metodología	20
Metodología Jenkins	20
Procedimiento y actividades realizadas	21
CAPÍTULO IV.....	23
Resultados	24
Proyectos.....	25
Casa de Terror BEG (5-7 años).....	25
Aplicación de Restaurante Cliente-Administración ADV (9-11 años).....	37
Juego de Carreras Multijugador PRO (12-15 años).....	54
App Biblioteca digital PRO (12-15 años).....	66
CONCLUSIONES	77
Conclusiones	78
COMPETENCIAS DESARROLLADAS	79
Competencias Desarrolladas	80
Fuentes de información.....	82

Índice de tablas y figuras

Figura 1 Logo BYJU'S FutureSchool.	9
Figura 2 Inicio Casa de Terror	25
Figura 3 Mensaje de entrada a la casa de terror.....	26
Figura 4 Código activado al usar vagoneta.....	26
Figura 5 Mensaje al comer filete	27
Figura 6 Código al comer filete	28
Figura 7 Tanque de arañas en Casa de Terror	28
Figura 8 Código Tanque de Arañas.....	29
Figura 9 Cueva de Murciélagos	30
Figura 10 Código del sistema de puntuación.....	31
Figura 11 Pozo de ahogados	32
Figura 12 Creación de MOBs	32
Figura 13 Cambiar comportamiento de MOB	33
Figura 14 Código del pozo de los ahogados y guardianes.....	34
Figura 15 Final del viaje	35
Figura 16 Código de la lana final.....	35
Figura 17 Añadir menú	37
Figura 18 Chatbot	38
Figura 19 Pantalla Inicio Admin	39
Figura 20 Pantalla para actualizar menú	40
Figura 21 Código pantalla de actualizar menú	41
Figura 22 Pantalla de verificar orden.....	42
Figura 23 Chat.....	42
Figura 24 Código chat.....	43
Figura 25 Pantalla Inicio Cliente	44
Figura 26 Pantalla del menú	45
Figura 27 Código conexión base de datos	46
Figura 28 Pantalla menú	46
Figura 29 Código añadir y eliminar producto.....	48
Figura 30 Pantalla menú con carrito de compra	49
Figura 31 Código del carrito de compra	50
Figura 32 Pantalla de ordenar	51
Figura 33 Código pantalla ordenar	52
Figura 34 Pantalla de chat.....	52
Figura 35 Código de chat.....	53
Figura 36 Inicio del juego	54
Figura 37 Clase Form.....	55
Figura 38 Base de datos del juego	56
Figura 39 Index del juego	56
Figura 40 Juego.....	57
Figura 41 Código de clase Player	58

Figura 42 Código de creación de sprites	59
Figura 43 Código de grupos de sprites	60
Figura 44 Código de combustible y vida de los jugadores	61
Figura 45 Código de bonificaciones y botón de reinicio	62
Figura 46 Código de tabla de puntuación	63
Figura 47 Fin del juego	64
Figura 48 Código de alertas	64
Figura 49 Subir carpeta a Github	65
Figura 50 Código app principal	66
Figura 51 Componente de navegación.....	67
Figura 52 Autenticador	68
Figura 53 Código de inicio de sesión.....	69
Figura 54 Creación de la base de datos.....	70
Figura 55 Código de pantalla de búsqueda	71
Figura 56 Código permiso de cámara	72
Figura 57 Código para obtener detalles de los libros.....	73
Figura 58 Código de devolución del libro	74
Figura 59 Código de tipo de transacción y elegibilidad	75
Figura 60 Código de componentes de pantalla transacciones	76
 Tabla 1 Competencias desarrolladas	 80
 Diagrama 1 Diagrama Ishikawa.....	 80

Resumen

Este trabajo tiene como objetivo mostrar las actividades realizadas dentro del centro de trabajo BYJU's Future School, los cuales pueden extrapolarse al tecnológico debido a que las herramientas tecnológicas y software que se utilizan pueden apoyar al desarrollo de competencias transversales de los alumnos para aumentar las habilidades y destrezas en el ámbito laboral, se utilizó la metodología Jenkins que nos permite identificar cada fase de los elementos para el desarrollo de las diferentes actividades que se realizan. Dentro de las actividades que se realizaron con las diferentes herramientas tecnológicas, podemos dividir las mismas dependiendo de la edad de cada grupo de niños, y así poder utilizar la metodología para explotar al máximo las habilidades de cada uno de los mismos y mostrarles de lo que son capaz. Las herramientas tecnológicas que se utilizaron fueron software de licencia libre como lo fueron Studio Code, Mit App Inventor, Visual Studio Code y Bluestacks los cuales pueden proporcionar nuevas competencias sobre todo en la especialidad de desarrollo de aplicaciones móviles dentro del Tecnológico, que permitan coadyuvar al posicionamiento de los alumnos, alcanzar el perfil de egreso y ubicar el Tecnológico como vanguardia del Tecnm.

Agradecimientos

A mi mamá por siempre estar alentándome a ser una mejor versión de mí, por ser maestra, madre y amiga, por ayudarme a ver la luz cuando estaba en las sombras.

A mi hermana por mantenerme en pie cuando solo quería estar en el suelo, por estar cuando nadie más estuvo y recordarme el valor que muchos me quitaron.

A mi papá por recordarme siempre el valor de la vida y luchar por aferrarme a la vida que espero tener.

A mis amigos por esas tardes de plática que te ayudan a ver siempre lo positivo en la vida cuando es difícil entender que iba mal.

CAPÍTULO I

GENERALIDADES DEL PROYECTO

No dejaremos de explorar y al final de nuestra búsqueda llegaremos a donde empezamos y
conoceremos por primera vez el lugar - Thomas Stearns Eliot

Introducción

Hoy en día la tecnología se encuentra en todas las áreas de nuestra vida, incluso en los lugares más inimaginables gracias al avance a pasos agigantados, se tienen actualizaciones de como se ve o funciona el mundo que nos rodea, por lo tanto, es necesario informarnos como es que esta tecnología puede mejorar nuestras vidas y como nos beneficiará en la forma en la que nos desenvolvemos; con lo cual es necesario que las próximas generaciones se vuelvan expertos en la utilización de esta misma.

Por ello, es necesario instruir a los más pequeños en el funcionamiento de la misma, además que ellos mismos al estar en constante utilización de la misma desde que son mucho más pequeños, son curiosos y quieren aprender sobre cómo realizar las aplicaciones o los juegos con los que se entretienen durante horas, para ello necesitan una guía pues, a pesar de encontrar algunos videos o información al respecto, no se puede entender correctamente pues la mayoría está diseñada para personas con conocimiento previo.

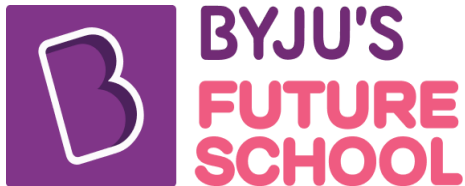
Es por esta razón que el presente trabajo nos explica algunas de las ventajas del aprendizaje a una temprana edad, así como algunos softwares necesarios para realizar las actividades con las que los niños aprenden sobre el mundo tecnológico, así como una muestra de sus trabajos realizados.

Descripción de la empresa

BYJU'S FutureSchool es la plataforma de tecnología educativa líder en la India, que ofrece resultados comprobados para niños y maestros. Según lo aplicado durante las clases los niños que aprenden programación durante la primera infancia (aproximadamente entre los 5-15 años) obtendrán mejores puntajes en matemáticas y durante el pensamiento lógico y abstracto de las pruebas.

Figura 1

Logo BYJU'S FutureSchool.



Nota BYJU'S Learning. (2021). *BYJU'S Learning*. <https://byjus.com/mx/futureschool/>

La misión de BYJU'S FutureSchool es empoderar a toda una generación para que se conviertan en creadores tecnológicos en lugar de consumidores. Durante las clases se enseñan los fundamentos de la programación, pensamiento lógico, estructural, secuencial y algorítmico, lo que permite a los niños producir resultados creativos como sitios web, animaciones, apps, etc.

BYJU'S FutureSchool tiene presencia en diferentes países alrededor del mundo como lo son India, Reino Unido, Estados Unidos, Australia, Indonesia, Brasil, México, Colombia, Chile, Perú y Argentina, adaptando sus contenidos tanto en el idioma específico del país donde se encuentra.

Descripción de la participación activa en proyectos dentro de las líneas de investigación

Dentro de la organización existen diferentes objetivos los cuales están enfocados en la mejora en el aprendizaje para los niños, por ello mi participación es ser guía para el desarrollo de las diferentes actividades, explicando los diferentes conceptos que conlleva, así como la resolución de dudas.

Justificación

El propósito de este trabajo es poder aportar algunas opciones de software libre que puede ser utilizado para las clases dentro del tecnológico con la intención de fortalecer los conocimientos proporcionados; así mismo, se busca explicar de una manera concreta el cómo se realizan las actividades dentro del centro de trabajo y con ello conseguir el logro de la titulación por estancias.

En la actualidad dentro del tecnológico hay carencia de algunos recursos tecnológicos como por ejemplo el internet, pc para el alumnado y adquisición de licencias para poder realizar prácticas que permitan alcanzar el perfil de egreso, por otro lado se puede observar que hacen falta cursos de capacitación especializados tanto al área pedagógica como al área de expertis de parte de los profesores aunado a esto la resistencia al cambio por utilizar nuevas tecnologías o lenguajes de programación y la falta de estímulos en el desarrollo de competencias transversales.

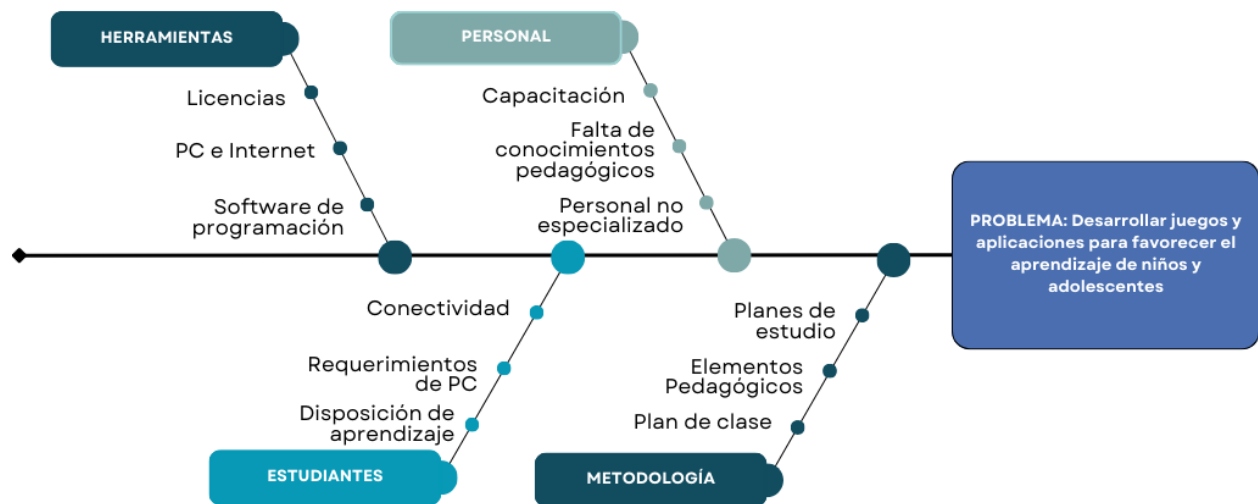
De acuerdo a la actualización constante que se vive en el mundo empresarial se requiere una paridad en el desarrollo de nuevas especialidades que permitan a los estudiantes cumplir con las competencias laborales que se requieren para enfrentar la competitividad y requerimientos que existen en el mercado, por consiguiente se requiere actitud y disposición por parte del alumnado considerando que muchos de ellos no cuentan con los recursos económicos y tecnológicos para poder enfrentar este tipo de situaciones en su hogar y el acceso restringido por parte de la institución.

Finalmente, todos estos aspectos también se ven afectados por la falta de hábitos de estudio y desconocimiento del tipo de aprendizaje en el que ellos podrían obtener una significancia en los

temas y tópicos de actualización dentro del mundo empresarial por lo que se requiere de la tríada de institución, alumno y profesor para poder llevar a cabo el aprendizaje significativo.

Diagrama 1

Diagrama de Ishikawa



Nota Autoría propia

Objetivo General

Desarrollar juegos y aplicaciones para favorecer el aprendizaje de niños y adolescentes

Objetivos Específicos

- Definir los pasos para la creación de app móviles en MIT App Inventor
- Establecer los elementos de interconexión de la base de datos en Firebase a través del index con la conexión en JavaScript
- Relacionar elementos de videojuegos para el desarrollo de código en JavaScript dentro de Minecraft Education Edition

- Generar una aplicación en React Native con base de datos para el manejo eficiente de una biblioteca

Alcances y limitaciones

Dentro de los programas dentro del centro de trabajo se determina por la edad de los pequeños cual es el más adecuado para ellos, teniendo 4, el plan Principiante que engloba a pequeños de entre 5 y 7 años, algunos de ellos aún no han aprendido a leer o a escribir por lo que se tiene el reto de enseñarles programación con diferentes técnicas para que puedan entender correctamente los conceptos. Después se tiene el programa Intermedio donde se encuentran niños de entre 7 y 9 años, las actividades siguen siendo mucho más dinámicas en comparación con cursos adelantados, pero se desarrollan tareas más complejas; el siguiente es Avanzado en el cual están chicos de entre 9 y 11 años, aquí se desarrollan aplicaciones más complejas tanto de lógica como de código, se enlaza información adaptada para que sea más llamativa para ellos. Por último, se tiene el programa Profesional, donde están adolescentes de entre 12 y 15 años, se comienza por el desarrollo de juegos con código escrito, y se avanza a aplicaciones con comandos especializados.

Las actividades son secuenciales, por lo tanto, se busca que los niños no pierdan ninguna clase para evitar que puedan atrasarse en el conocimiento, al utilizar software libre (en la mayoría de los casos) los niños pueden seguir utilizándolo fuera de la clase para realizar los proyectos que más les llame la atención, ya que muchos de ellos, tienen la intención de crear algún videojuego o aplicación que ya han tenido en mente.

Algunas limitaciones dentro de los programas pueden ser tanto la edad como el presupuesto, ya que en el primer caso, como las actividades están desarrolladas para ciertas edades, si se coloca a un niño en un nivel inferior o superior al que debería, pueden desarrollar frustración por no entenderle a las tareas o que sean muy fáciles para ellos; en el caso del presupuesto se debe a que algunos software deben ser utilizados en computadoras con ciertas características de hardware que para algunos padres no pueden permitirle a sus hijos.

CAPÍTULO II

MARCO TEÓRICO

Observa detenidamente tu entorno y comprenderás mejor el funcionamiento de las cosas - Albert
Einstein

Marco teórico

Según la Real Academia de la Lengua Española (Real Academia Española, 2023), el aprendizaje es la adquisición de conocimiento de algo por medio del estudio o de la experiencia, sin embargo, el ganar esta información valiosa puede hacerse de muchas maneras y por ello con el pasar de los años se han desarrollado diferentes técnicas que pueden servir para este propósito (Maudson, 2022). En el caso de los niños, se ha resuelto que a partir de los primeros 5 años de su vida, la capacidad en el desarrollo del lenguaje, pensamiento y habilidades tanto sociales como emocionales es exponencialmente mayor.

El desarrollo cognitivo es un proceso mediante el cual el ser humano va adquiriendo conocimiento (INVANEP, 2020), por lo cual el psicólogo Jean Piaget dedicó varios de sus estudios en ello, afirmando que los principios de la lógica se desarrollan antes que el lenguaje y que esta misma es base del pensamiento, y por ende de la inteligencia. Piaget definió 4 etapas para este desarrollo las cuales son:

- Menores de 3 años - Los niños buscan el conocimiento de sí mismos, explorando su propio cuerpo y la relación con sus figuras de apego
- Mayores de 3 años y menores de 7 años - Se busca el conocimiento del mundo más allá de ellos, utilizando las habilidades construidas en la etapa anterior, de manera intuitiva el infante decide que comportamientos imitar y cuales evitar
- Entre 7 y 12 - Se continúa con el proceso del conocimiento del mundo a su alrededor, sin embargo, el desarrollo de su lógica es mayor, utilizándola para comprender lo que percibe a su alrededor
- A partir de los 12 años - Se termina las etapas anteriores y se desarrolla el pensamiento hipotético, preguntando por sus propias capacidades

En una época en la cual la tecnología a modificado el modo de ver la vida, los niños tienen acceso a la tecnología desde una temprana edad, lo cual propicia la curiosidad de saber cómo funciona el mundo a su alrededor, enfocado especialmente en los aparatos que han reemplazados los juguetes tradicionales, por ello, algunos padres son mucho más afectos a que aprenden sobre ellas al considerarlas necesarias para la vida adulta de los niños.

Dentro de este desarrollo de habilidades se encuentra la enseñanza de la programación, la cual, aparte de ayudar a responder las preguntas de los pequeños sobre los temas que los inquietan referentes a la tecnología, ayuda al desarrollo de la lógica que como se mencionó con anterioridad ayuda a ver el mundo de una mejor manera, ordenando sus pensamientos y necesidades en procesos más estructurados, para lo cual se apoya en los lenguajes de programación.

El concepto de los lenguajes de programación se deriva de la necesidad de colocar nombre a las secuencias tecnolingüísticas que nos permiten introducir datos y procesos en la máquina (Ghezzi y Jazayeri, 1994); se les llama de esa manera ya que comparte utilidad con idiomas naturales, ya que nos ayuda a la comunicación de ideas y plasmarlas en los diferentes programas y editores. Tal como los idiomas, existen muchos lenguajes dentro del área de la programación, algunos mayormente enfocados en realizar tareas específicas, derivadas de su propia sintaxis y uso que se les ha otorgado a lo largo del tiempo.

El lenguaje de programación más versátil, es JavaScript que es un editor de código compatible para diferentes lenguajes de programación, entre ellos JavaScript, JSON, etc. (Gutierrez, 2009). Gracias a que es de código abierto permite agregarle diferentes extensiones que permiten utilizar al máximo todas las características el editor, así como añadir algunas extras que proporciona.

La utilización de este tiene que adaptarse naturalmente a las habilidades de los niños, avanzando de una manera más orgánica para evitar la repulsión del conocimiento específico, así como, la complicación extra de los mismos; por ello hay algunas plataformas que nos apoyan a que esta transición de lenguajes naturales a lenguajes de programación se dé de una manera mucho más sencilla para los niños.

Una de estas plataformas es MIT App Inventor 2 la cual es un entorno de programación que permite a cualquier persona crear aplicaciones funcionales para sistema operativo Android, así como iOS (Fernández, 2019). Se utiliza un motor basado en bloques que facilita la creación de aplicaciones complejas en menor tiempo que en los entornos de desarrollo tradicionales. La plataforma es impulsada por un equipo de estudiantes de MIT CSAIL liderados por el profesor Hal Abelson que se encarga de investigaciones sobre los impactos de plataformas de desarrollo de aplicaciones; este equipo mantiene de manera gratuita el entorno de desarrollo con más de 6 millones de usuarios registrados (Mit App Inventor, 2012 - 2022).

Los programas de codificación basados en bloques pueden estimular la creatividad, lo cual permite crear aplicaciones que logren crear un mayor impacto social en la comunidad, además de poder realizar los comandos de una manera más intuitiva que les permite mejorar la lógica con la que realizan las actividades, del mismo modo utilizar plataformas de juegos puede ayudar a desarrollarlos.

MakeCode es una plataforma gratuita para aprender sobre programación, su sistema basado en bloques los cuales permiten crear Mods -Modificadores- en Minecraft, cuenta con un sistema para desarrollar el código en JavaScript, así como un simulador que muestra el resultado en el juego (Microsoft, 2013). Dentro de la plataforma, existen videos o tutoriales que permiten a los niños a aprender de manera más sencilla los códigos necesarios para realizar alguna tarea. Aunado a esto, tiene algunos eventos especiales y mensuales en donde incentiva interactuar con los retos, donde se encuentran diferentes temas a tratar, desde las emociones hasta el espacio.

Como se mencionó la utilización de ciertos juegos permite el desarrollo cognitivo de los niños, por ello se utiliza uno de los juegos más populares del rango de edad de los pequeños, mezclándolo con la programación para que puedan entender como es que el mundo de su videojuego funciona, atendiendo a todas estas cuestiones, el equipo de Mojang adaptó una versión para ello llamada Minecraft Education Edition la cual es una plataforma donde se puede aprender desde programación hasta química (Minecraft Education, 2023).

Al estar conectado con MakeCode, permite probar los mods en el momento, pudiendo reproducir los resultados al instante, lo cual hace más interesante el aprendizaje. Tiene la opción de obtener diferentes compuestos químicos a partir de bloques de construcción en Minecraft y cuenta con una entidad exclusiva de la plataforma llamado Agente que nos permite el acceso a la sección de código, sin embargo, no es la única plataforma que se involucra.

Tynker es una plataforma que permite la creación de diferentes proyectos, desde Python hasta JavaScript. Enfocado a Minecraft, permite la personalización de los componentes del juego, como lo son Bloques, Elementos, Apariencia del jugador, así como Criaturas, en estos últimos se les puede agregar comportamientos diferentes, que se pueden importar no solo a la plataforma educativa de Minecraft, sino al juego en sí, por lo que puede hacer una experiencia de juego mucho más entretenida para los niños (Tynker from BYJU'S, 2023).

Al estar en constante progreso tecnológico, también la idea de las inteligencias artificiales es de la atención de los pequeños, entre las más famosas, las IAs de diálogo, por ello se utiliza DialogFlow, la cual, es una plataforma de comprensión del lenguaje que permite diseñar e integrar interfaces de usuario conversacionales en aplicaciones móviles, web, bots, etc. (Google Cloud, s. f.). Al ser enteramente entrenado por los usuarios está capacitado para adaptarse a las situaciones en las que se utilizará la información; puede analizar entrada de texto o audio ya que utilizar el procesamiento del lenguaje natural para la interacción a través del machine learning.

Como unificador podemos tener diferentes editores de código, que puede ser compatible con diferentes lenguajes de programación, de código abierto, dentro de ellos, se ve Visual Studio Code, el cual, permite utilizar extensiones adicionales para usar al máximo todas las características del editor, que proporciona al desarrollador una interfaz mucho más adaptada a sus necesidades (Johnson, 2019).

React Native es un framework que ayuda a la creación de aplicaciones nativas multiplataforma (IOs y Android), está basado en la librería de JavaScript (Paul y Nalwaya, 2019). Al ser aplicaciones nativas pueden tener acceso a las diferentes funcionalidades de hardware de los dispositivos móviles en los que se ejecuta la app. Para ver la funcionalidad del mismo en un entorno local, se tiene que utilizar las librerías disponibles.

Por ello se utiliza NPM que es un sistema de gestión de paquetes de Node.js, que nos permite en un entorno local la ejecución de JavaScript (Puciarelli, 2020). Dentro de la misma, hay varios paquetes de componentes que ayudaran a la mejoría del diseño y experiencia de usuario de las aplicaciones, para poder utilizarla se tiene que ingresar en la Terminal de la computadora.

Node.js es un entorno multiplataforma basado en JavaScript (Puciarelli, 2020), funciona como un modelo asíncrono para tener acceso a los diferentes repositorios que React pueda utilizar para la mejoría en la interfaz y experiencia de usuario de usuario. Es usada en el desarrollo de aplicaciones ya que se actualiza frecuentemente teniendo acceso a mejores herramientas.

Para poder mostrar los resultados que se tienen existen muchas plataformas que nos permiten almacenar la información, pero la más utilizada es GitHub, que es una plataforma de desarrollo colectivo utilizada especialmente para alojar proyectos dentro de repositorios (GitHub, 2023), los cuales contienen la información más importante del programa en un archivo llamado Read Me, así

como la posibilidad de acceder a los diferentes programas sin la necesidad de un compilador, lo que permite, que al estar almacenados puedan ser llamados de manera virtual ya en funcionamiento.

CAPÍTULO III

DESARROLLO

Lo que conocemos es una gota, lo que no conocemos es un océano - Sir Issac Newton

Metodología

Metodología Jenkins

Durante el proceso de realización de las actividades se utilizará la metodología Jenkins la cual ayuda al desarrollo de la creatividad y lógica durante el proceso de creación, pues al ser iterativa, permite la mejora continua de los proyectos en los que los pequeños están envueltos y la distinción de cada una de sus partes (Hanke y Wichern, 2006).

Fase 1. Análisis del problema

En esta fase, se determina cual es el propósito de las actividades a realizar, o el problema que se requiere solucionar con la misma, dentro de esta, se realizan pequeños bocetos o layouts para que se pueda entender que elementos van a complementar las actividades; esta sección engloba la idea que se va a realizar, los software o diseños que se utilizarán para el desarrollo satisfactorio de la misma.

Fase 2. Diseño del sistema

Dentro de la misma se colocan todos los elementos que se necesitarán para el uso por parte del usuario de la actividad, ya sea el desarrollo de la parte front-end en cuestión de las aplicaciones o juegos o en caso de los nuevos diseños de criaturas en el caso de Minecraft; está es toda la parte visual enfocada tanto en el propósito de la actividad como en la cuestión de UI.

Fase 3. Implementación del sistema

Durante la fase de implementación, se desarrolla toda la cuestión de codificación de la actividad; al tener ya los lenguajes de programación que se utilizarán, primero se realizará un algoritmo de las diferentes funciones que necesitará para poder utilizar todos los elementos de la actividad, tomando en cuenta la experiencia de usuario para que sea más sencillo de utilizar.

Fase 4. Operación

La fase de operación o prototipo, es donde se realizan pruebas de la utilización de las actividades, es necesario revisar que coincida con el propósito de la actividad para comprobar que la implementación halla sido correcta, de la misma manera que el diseño pueda cumplir con las

especificaciones necesarias, ya sean de color o letra que se utilizó y por supuesto que las funciones realicen las actividades que deben.

Procedimiento y actividades realizadas

Durante el desarrollo de las actividades se utiliza una guía que funciona como Plan de Estudios, en función del nivel (a partir de las edades) de los niños se mencionan los conceptos básicos de lo que se verá durante cada sesión. Dependiendo de la dificultad de las actividades se divide en diferentes sesiones que tienen una duración de aproximadamente 50 minutos por cada una.

Estos planes de estudios comprenden los objetivos de la clase, los recursos necesarios a utilizar, desde el hardware, hasta el software que comprenderá la misma, dado que se trata con niños se colocan algunas técnicas que pueden ayudar a enfocar la atención de los mismos, así como fomentar la autoestima de los mismos con algunos stickers que funcionan como recompensas por cumplir tanto con las actividades como con los objetivos del mismo.

Las clases están divididas en por lo menos 4 secciones, en la primera llamada Rompiendo el hielo, se desarrolla un repaso de la clase anterior para identificar si hay alguna duda que pueda tanto impedir el desarrollo de la clase como atrasarla; agregando esto, la conexión emocional con los niños ayuda mucho al aprendizaje tanto del maestro como del alumno, por ello se plantea también preguntar sobre el alumno como lo que se ha hecho en clase, etc.

La segunda sección es la actividad de la maestra, en ella se deben explicar todos los conceptos que se utilizarán en la clase, en caso de que sea necesario instalar algún software o iniciar sesión en alguna plataforma se indicarán los pasos para poder realizarlo. Se muestra el resultado final de lo que se realizará en la clase y se demuestra una parte de como escribir el código; dentro del documento, se añaden algunos enlaces que permitirán el acceso más rápido a las plataformas o alguna información extra que podrá ser útil para los niños.

En la sección de la actividad del alumno, se dejará que los niños desarrollen el código necesario hasta obtener lo que mostró la maestra al inicio de la sección anterior, en algunas ocasiones se permite copiar alguna parte de la pantalla que se comparte, aunque solo en caso de que se desarrollen entre varios estudiantes, en caso de que algún alumno tenga dudas, se responde

de manera asertiva y clara las veces necesarias para que la información haya quedado clara, en caso de que algún alumno termine antes la actividad, se tienen algunas sugerencias de actividades extra que puede desarrollar.

Por último, se tiene la conclusión, en ella, todos los alumnos deben volver a la sesión, dejar de compartir sus pantallas y se hace un resumen de lo que se aprendió durante la sesión con la intención de reforzar los conocimientos obtenidos, se debe premiar a los niños que hayan cumplido con las actividades, esto ayuda a mejorar tanto la conexión con los alumnos, como su propia autoestima, pues al sentirse reconocidos por la persona que los instruye, mejora su capacidad de aprender y no solo memorizar la información, cada clase cuenta con un proyecto de reforzamiento, por lo que se explica lo que se debe realizar y se concluye la clase.

Los proyectos están disponibles como actividades fuera de clase, se utilizan mayormente para poder practicar lo visto en clase, durante las sesiones con los padres, se solicita que estén junto a los pequeños mientras las realizan para que puedan ver el avance de los mismos, sin embargo, se puede entender que por la edad de algunos de los niños o las ocupaciones no siempre es posible realizar, por ello se tienen sesiones de retroalimentación donde se llenan boletas de calificación donde la maestra puntúa la disposición de aprendizaje, el avance en el desarrollo de las actividades, la realización de los proyectos, entre otros.

CAPÍTULO IV

RESULTADOS

La idea era hacer investigación buscar nuevos caminos a conquistar, nuevas montañas que escalar - Gertrude Belle Elion

Resultados

Dentro de la plataforma Minecraft Education Edition se crean diferentes proyectos para que los niños puedan aprender programación de una manera más sencilla y adaptado al juego, en el cual pueden mostrar puntuaciones e incluso mensajes para explicar las reglas de la actividad a realizar. Los pequeños muestran gran interés en este tipo de programación ya que pueden trasladar la información y los conocimientos que reciben dentro de su juego favorito lo que se traduce en mayor atención a las sesiones, así como aumentar el interés en las mismas.

Gracias a la flexibilidad que proporcionan juegos como Minecraft se pueden crear distintos escenarios que a los niños les atraen para poder dejar volar su imaginación y creatividad a la hora de crear, Minecraft Education Edition permite que, a partir de bloques de código, traducidos de lenguaje JavaScript se puedan realizar las actividades que ayudan a entender los conceptos básicos de la programación.

Proyectos

Casa de Terror BEG (5-7 años)

Durante la temporada de día de muertos, los pequeños con el curso de Minecraft desarrollaron un juego multijugador de una casa de terror, en la cual se tuvieran criaturas hostiles, y se tuviera que recorrer con una vagoneta, así como algunos objetos que pudieran cambiar el estado del jugador durante el recorrido del mismo.

Figura 2

Inicio Casa de Terror

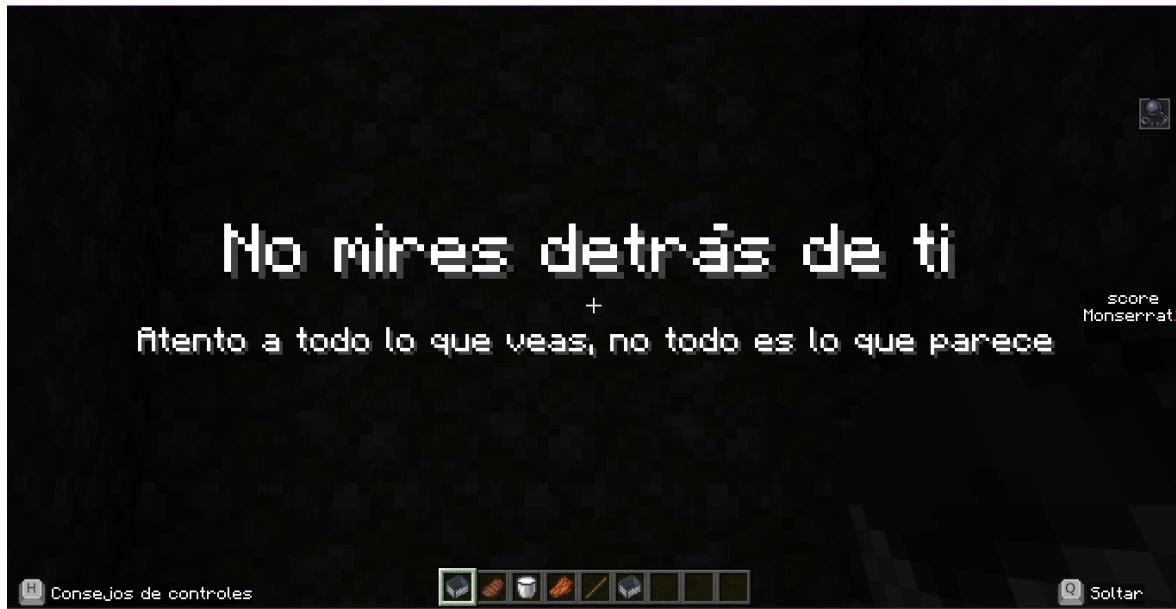


Nota Pantalla obtenida en Minecraft Education Edition

Se inicia la construcción de la casa del terror, la cual mientras sucede, el jugador es inmovilizado y cegado para que no pueda notar las distintas habitaciones con las que cuenta la estructura. Una vez terminado, manda un mensaje a los jugadores para dar la bienvenida al juego, colocando a los usuarios en la entrada de la misma.

Figura 3

Mensaje de entrada a la casa de terror



Nota Pantalla Obtenida en Minecraft Education Edition

Tras mostrar el inicio de la Casa de Terror, se colocó un cofre en donde se encuentran diferentes objetos que ayudarán en el viaje a través de la Casa de Terror; ya que la estructura está diseñada para viajar con una vagoneta, al utilizarla, el jugador es limitado de su visión para hacer aún más complicado y terrorífico el viaje a lo largo de la estructura.

Figura 4

Código activado al usar vagoneta

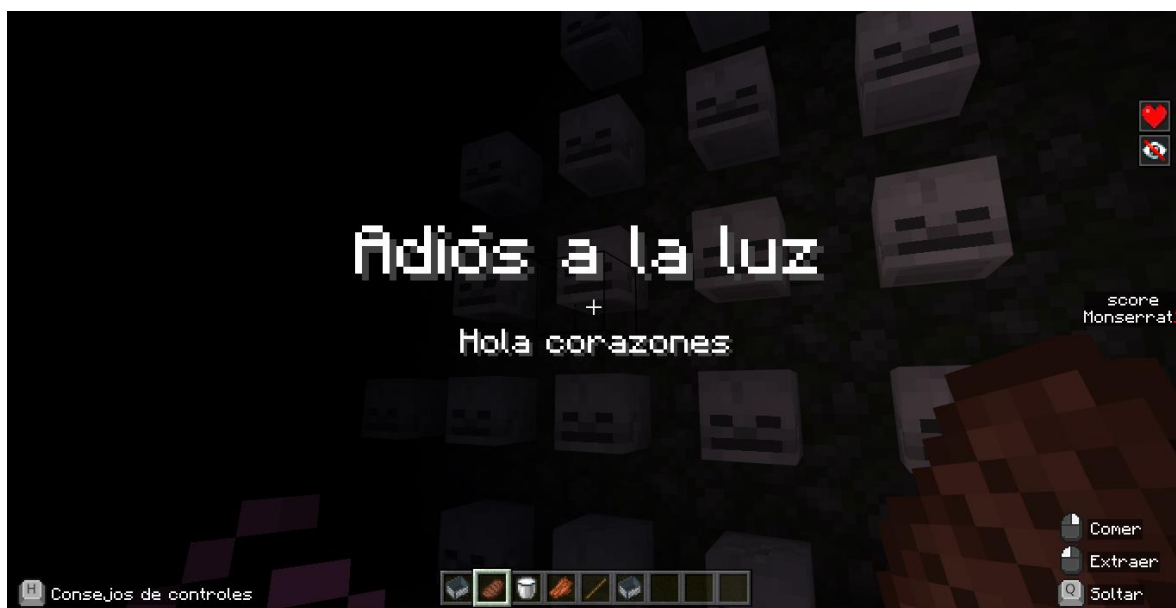
```
1328 player.onItemInteracted(MINECART, function () {
1329     mobs.spawn(mobs.monster(PHANTOM), pos(0, 5, 0))
1330     mobs.applyEffect(SLOWNESS, mobs.target(NEAREST_PLAYER), 140, 4)
1331     gameplay.title(mobs.target(NEAREST_PLAYER), "No mires detrás de ti", "Atento a todo lo que
1332     mobs.spawn(mobs.monster(ENDERMAN), pos(-5, 0, 0))
1333     blocks.fill(
1334         AIR,
1335         world(4, 5, -17),
1336         world(4, 6, -17),
1337         FillOperation.Replace
1338     )
1339 })
```

Nota Código JavaScript creado a partir de MakeCode

Dentro de Minecraft Education Edition, se encuentran algunos editores de código que proporcionan una manera sencilla de creación de código; MakeCode nos permite a partir de bloques de comandos realizar diferentes actividades, como lo son que al usar algún elemento u objeto perteneciente al juego Minecraft sucedan eventos, en este caso, se colocan efectos para disminuir la velocidad del jugador, invocar algunos MOBs hostiles (criaturas que pueden atacar al jugador) y abrir paso para avanzar durante el juego.

Figura 5

Mensaje al comer filete



Nota Pantalla obtenida en Minecraft Education Edition

Una de las mecánicas más importantes del juego original es que se debe alimentar al jugador para mantener o recuperar vida, así como aumentar la velocidad de movimiento, para ello se tienen a disposición diferentes tipos de alimento. Dentro del cofre al inicio del juego, se colocaron algunos de estos alimentos llamados “Filete asado” con la intención de que si el jugador fuera atacado por algún MOB creado con anterioridad o posterior pudiera recuperar la salud.

Figura 6

Código al comer filete

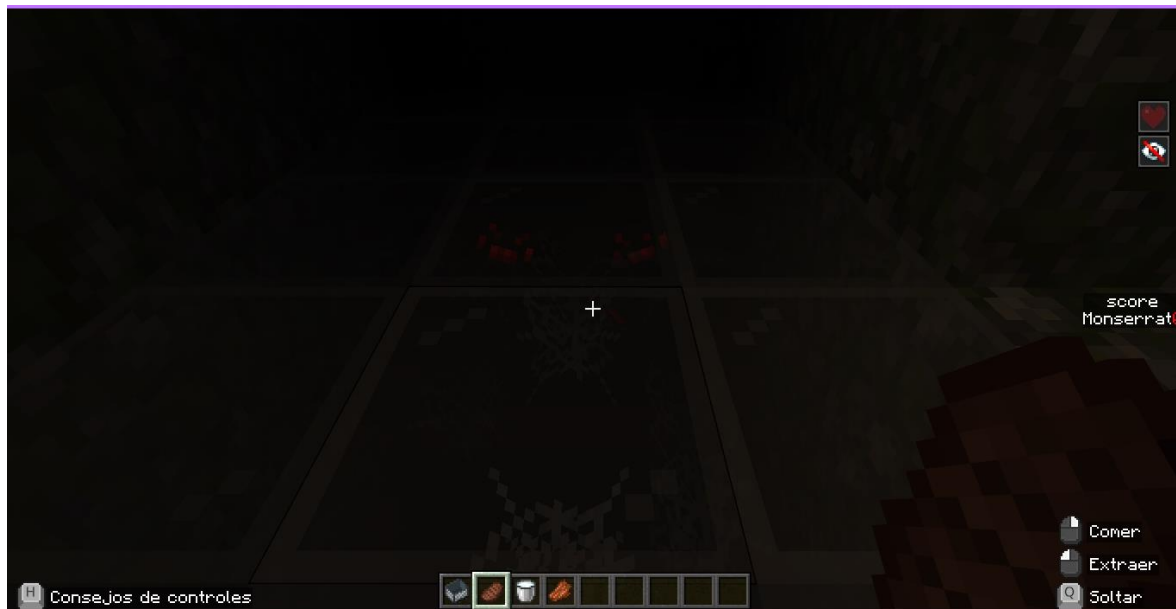
```
290 player.onItemInteracted(COOKED_BEEF, function () {  
291     gameplay.title(mobs.target(NEAREST_PLAYER), "Adiós a la luz", "Hola corazones")  
292     mobs.applyEffect(BLINDNESS, mobs.target(NEAREST_PLAYER), 50, 1)  
293     mobs.applyEffect(REGENERATION, mobs.target(NEAREST_PLAYER), 20, 1)  
294 })
```

Nota Código JavaScript creado a partir de MakeCode

La consecuencia al utilizar el alimento proporcionado por el programador del juego, fue que la visión del jugador se viera disminuida, sin embargo, se puede recuperar salud a una velocidad mucho mayor de la usual al “comer” el elemento, proporcionando también un mensaje al jugador, que le informa cual es la razón por la que no logra ver con más atención los objetos a su alrededor. Al usar efectos se tiene que determinar en primer lugar el comportamiento del mismo, el jugador o jugadores que obtendrán ese efecto, el tiempo que durará utilizando la medida tick (se refiere a la cantidad de ciclos que se ejecutan durante el juego, tick equivale a 0.05 segundos dentro del juego), así como la amplificación del efecto, esto definido en que tanto afectará al jugador la utilización de ese comportamiento.

Figura 7

Tanque de arañas en Casa de Terror



Nota Pantalla obtenida en Minecraft Education Edition

Con la intención de mantener la temática se creó un tanque de arañas de cueva, las cuales pueden trepar por las paredes y seguir a los jugadores por toda la estructura, dependiendo del nivel de dificultad en el que se determina el mundo en el que se encuentra el código que se crea; la araña de cueva tiene una media de daño de dos corazones (el jugador tiene 10 corazones de salud) así como la capacidad de envenenar lo cual reduce de nuevo la vida que se tiene, y obliga al jugador a recurrir al alimento que produce ceguera temporal.

Figura 8

Código Tanque de Arañas

```
585 function tanqueArañas () {  
586     blocks.fill(  
587         LIGHT_GRAY_STAINED_GLASS,  
588         world(8, 4, 4),  
589         world(10, 9, 4),  
590         FillOperation.Replace  
591     )  
592     for (let index = 0; index < 10; index++) {  
593         mobs.spawn(mobs.monster(SPIDER), world(9, 4, 5))  
594         for (let index = 0; index < 2; index++) {  
595             blocks.place(COBWEB, world(randint(8, 10), randint(4, 9), randint(5, 7)))  
596         }  
597     }  
598     blocks.place(COBWEB, world(9, 9, -8))  
599     blocks.place(COBWEB, world(9, 9, -2))  
600 }
```

Nota Código JavaScript creado a partir de MakeCode

Debido al daño que puede causar el MOB, se decidió colocar dentro de una zona especial rodeado de telarañas y con un vidrio frente al jugador, ya que con la oscuridad lo único capaz de ver serían los ojos de las arañas. Dentro del mismo código para la invocación de los MOB se utilizaron 10 arañas y por cada araña 2 telarañas, éstas se colocaban de manera aleatoria en la ubicación que compone el espacio donde vivirán las criaturas.

Figura 9

Cueva de Murciélagos



Nota Pantalla obtenida en Minecraft Education Edition

El jugador debía visitar algunas habitaciones llenas de MOB's terroríficos (tanto hostiles como neurales), por ello, se añadió una zona con más de 50 murciélagos utilizando un bucle FOR para la repetición de la generación de criaturas, atendiendo al principio de la programación DRY (por sus siglas en inglés Don't repeat yourself.); salas en donde se encontraban algunos objetos que ayudarían al jugador para la visita y supervivencia para las habitaciones consecuentes.

Figura 10

Código del sistema de puntuación

```
439 function SistemaPuntos () {
440     for (let index = 0; index < 10; index++) {
441         mobs.spawn(LIGHTNING_BOLT, randpos(
442             pos(-5, 0, -5),
443             pos(-5, 0, 5)
444         ))
445     }
446     BarrasInicio()
447     player.execute(
448         "/scoreboard objectives add scores dummy score"
449     )
450     player.execute(
451         "/scoreboard objectives setdisplay sidebar scores"
452     )
453     player.execute(
454         "/scoreboard players set @a scores 5"
455     )
456     player.execute(
457         "/scoreboard players set @s scores 0"
458     )
459 }
```

Nota Código JavaScript creado a partir de MakeCode

Dado que la modificación en el juego permitía tener más de un jugador se añadió un sistema de puntos en los cuales los jugadores podían obtenerlos a partir de la eliminación de los MOBs encontrados en las diferentes salas del juego. Para ello se utilizaron los comandos pertenecientes a Minecraft del Scoreboard en donde se colocó la puntuación a un costado de la pantalla del jugador, así como puntuaciones iniciales a los jugadores. Al iniciar el código automático de la tabla de puntuaciones se añadieron algunos relámpagos para marcar el inicio del viaje de los jugadores.

Figura 11

Pozo de ahogados

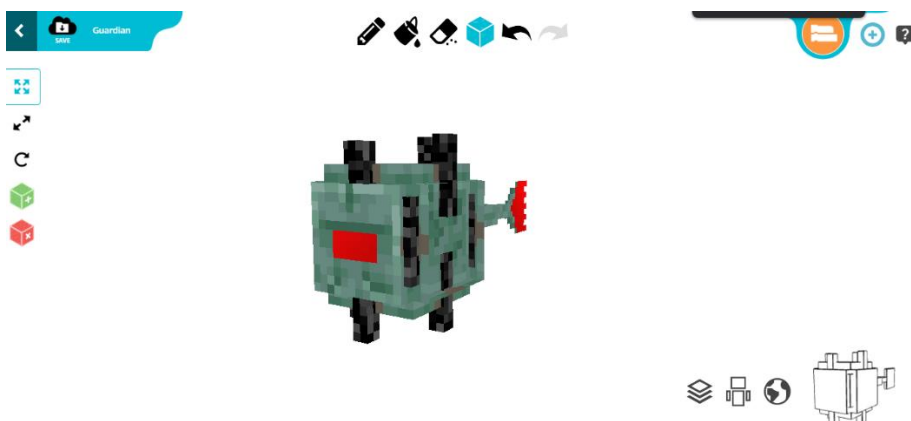


Nota Pantalla obtenida en Minecraft Education Edition

Al saber que existen algunas criaturas peligrosas que viven dentro del agua, se creó un pozo de ahogados y guardianes, los cuales pueden dañar al jugador al lanzar tridentes o efectos de fatiga (que impide romper con facilidad los bloques que proporcionan puntuación), en ellos, el jugador debía caminar por una minúscula pasarela a merced de las criaturas.

Figura 12

Creación de MOBs

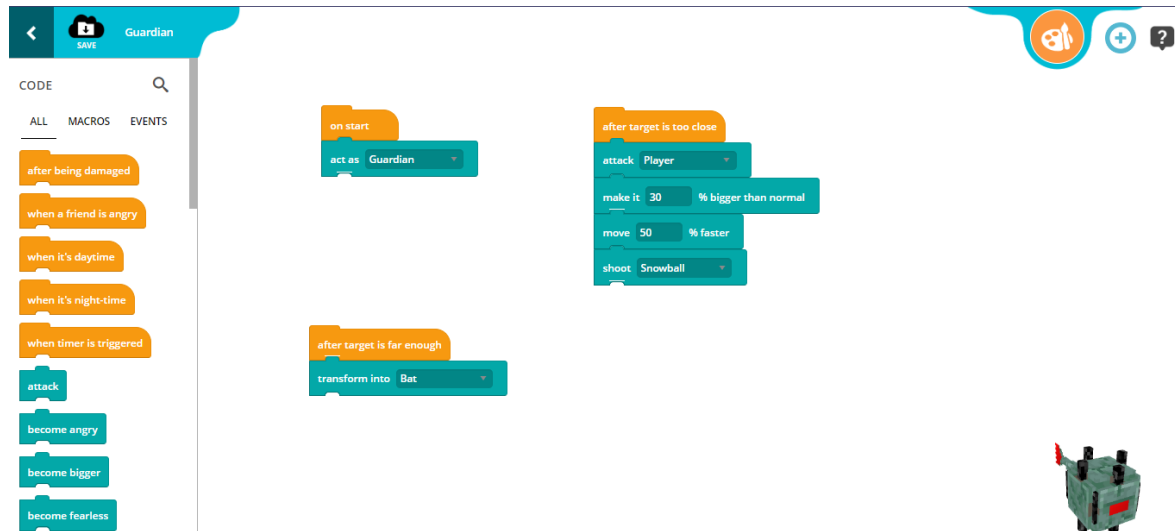


Nota Pantalla obtenida de Tynker

Para hacer más personalizable el juego, se modificó el diseño de uno de los MOBs para hacerlo mucho más terrorífico, dentro del editor de Tynker, se pueden añadir piezas nuevas, modificar su tamaño, girar piezas o incluso cambiarlas de color; se utilizó al guardián debido a sus efectos y que es otro MOB hostil que se encuentra en el agua.

Figura 13

Cambiar comportamiento de MOB



Nota Pantalla obtenida de Tynker

Tras haber modificado el diseño del MOB en cuestión, se decidió también cambiar el comportamiento, a parte del efecto de fatiga (estándar en el MOB), se añadió que cuando el jugador estuviera cerca atacará al jugador, se volviera 30% más grande de lo normal, se moviera 50% más rápido y lanzara bolas de nieve, que no causan mucho daño al jugador, sin embargo puede desestabilizarlo al pasar sobre las criaturas, además, cuando el jugador estuviera lejos, se convierte automáticamente en un murciélago, que es una criatura que no causa daño.

Figura 14

Código del pozo de los ahogados y guardianes

```
1 function Criaturas_Acuaticas () {
2     lista = [
3         DEAD_TUBE_CORAL_FAN,
4         DEAD_BRAIN_CORAL_FAN,
5         DEAD_BUBBLE_CORAL_FAN,
6         DEAD_FIRE_CORAL_FAN,
7         DEAD_HORN_CORAL_FAN
8     ]
9     for (let index = 0; index < 3; index++) {
10         mobs.spawn(mobs.monster(GUARDIAN), world(-15, 3, 0))
11         mobs.spawn(mobs.monster(DROWNED), world(-15, 3, 0))
12     }
13     blocks.place(SEA_LANTERN, world(-11, 2, 0))
14     for (let index = 0; index < 10; index++) {
15         for (let value of lista) {
16             blocks.place(value, world(randint(-19, -12), 2, randint(9, -12)))
17         }
18     }
19 }
```

Nota Código JavaScript creado a partir de MakeCode

En la zona del pozo, con la intención de decorar se ocuparon diferentes tipos de corales muertos guardados en un arreglo, después mediante un FOR, se añadían de manera aleatoria, lo que significa que, a pesar de contener los diferentes corales, podían aparecer de manera aleatoria en el rango que componía el pozo. Se añadieron 3 criaturas diferentes en el agua del pozo y se colocó una linterna marina para que el jugador lograra ver alguno de sus atacantes.

Figura 15

Final del viaje



Nota Pantalla obtenida en Minecraft Education Edition

Al llegar al final de la casa, habiendo atravesado cada una de las habitaciones que componía la estructura, había un bloque de lana que, al romperlo, lograba teletransportar al jugador fuera de la casa, proporcionaba un mensaje personalizado con la leyenda “Sobreviviste” y lanzaba algunos fuegos artificiales que indicaban el final del juego.

Figura 16

Código de la lana final

```
601 blocks.onBlockBroken(ORANGE_WOOL, function () {
602     prueba += 1
603     puntos += 1
604     puntos2 += -1
605     player.execute(
606         "/scoreboard players set " + mobs.target(NEAREST_PLAYER) + " scores " + puntos2
607     )
608     player.execute(
609         "/scoreboard players set " + mobs.target(LOCAL_PLAYER) + " scores " + puntos
610     )
611     mobs.clearEffect(mobs.target(LOCAL_PLAYER))
612     player.teleport(world(-40, 4, 0))
613     gameplay.title(mobs.target(LOCAL_PLAYER), "Felicitades" + mobs.target(NEAREST_PLAYER),
614         for (let index = 0; index < 10; index++) {
615             mobs.spawn(FIREWORKS_ROCKET, world(-40, 6, 0))
616         }
617 })
```

Nota Código JavaScript creado a partir de MakeCode

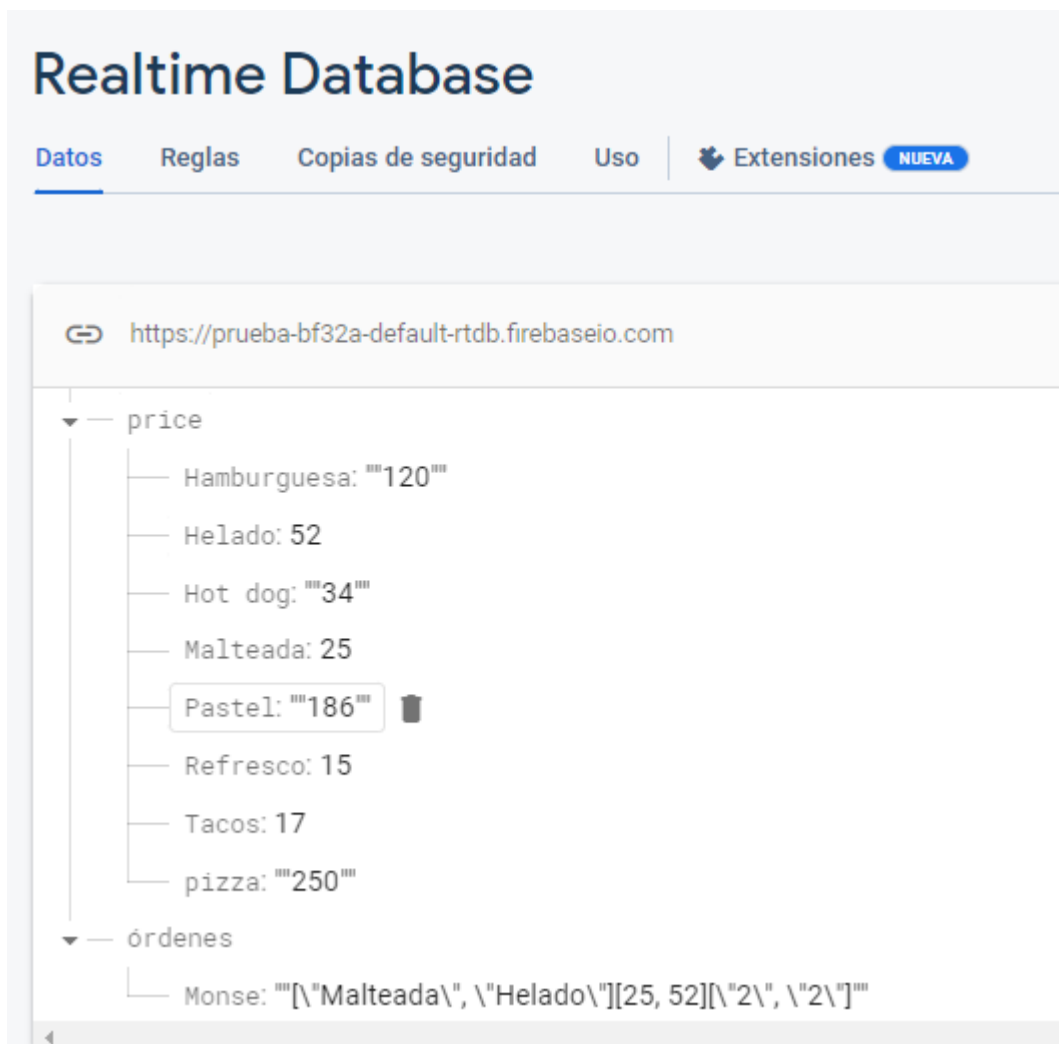
Tras completar el viaje de la casa de terror, sobreviviendo a todas las criaturas, se debía romper un bloque de lana, que fue seleccionado ya que si el guardián confería el efecto de fatiga minera (la cual hace que el jugador rompa bloques con mayor lentitud) aún se pudiera romper y escapar de los MOBs que pudieran perseguir al jugador. Se añadieron puntos por romper el bloque, así como la eliminación completa de los efectos con los que pudiera estar el jugador, añadía 10 fuegos artificiales que explotaban sobre el jugados y añadía el mensaje de felicidades con el nombre del jugador para hacer la experiencia mucho más personalizada.

Aplicación de Restaurante Cliente-Administración ADV (9-11 años)

Los niños de entre 9 y 11 años realizan una aplicación destinada a un restaurante en el cual se crea un chatbot para describir los platillos que contiene y se realizan 2 app, una de usuario en la cual se puede realizar el pedido de los diferentes alimentos y otra pantalla de administración en donde se tiene acceso al menú y los pedidos realizados, utilizando bases de datos.

Figura 17

Añadir menú

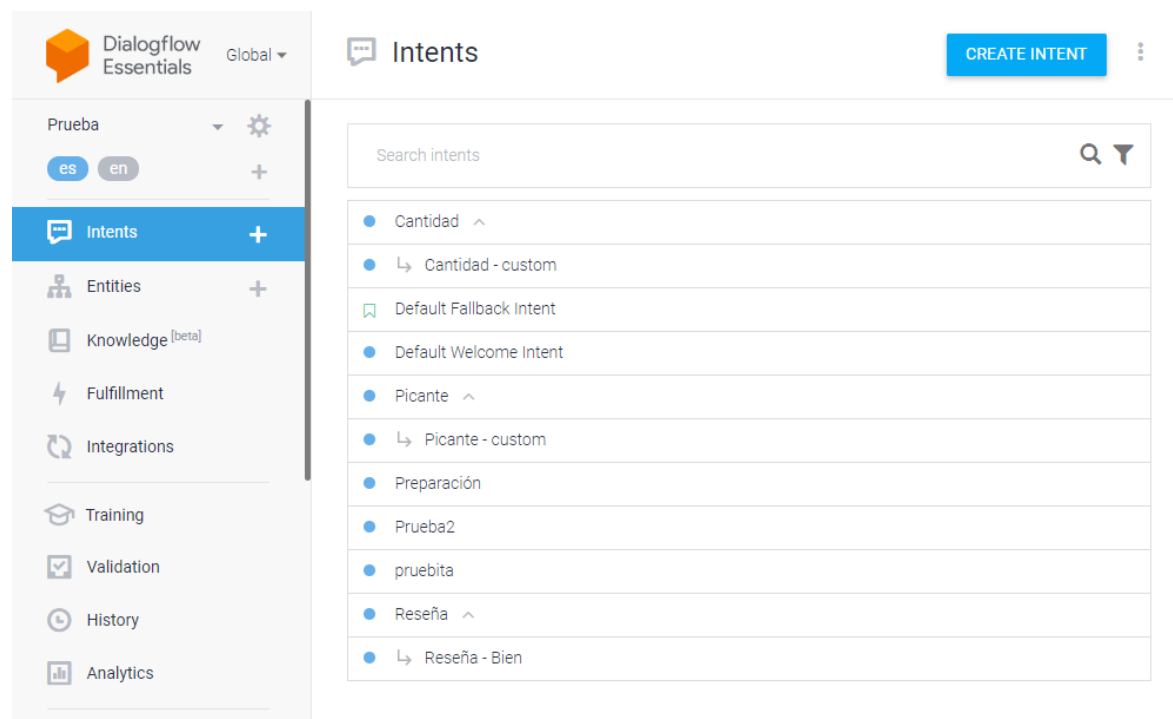


Nota Pantalla obtenida de Google FireBase

Dado que se utilizarán diferentes elementos del menú o para el registro de las órdenes, se utilizó una base de datos alojada en servidores de Google, por lo que la información se coloca de manera online para su posterior consulta o actualización de la misma, para este caso, se necesita la URL de la base de datos en tiempo real para poder tener acceso a la misma.

Figura 18

Chatbot

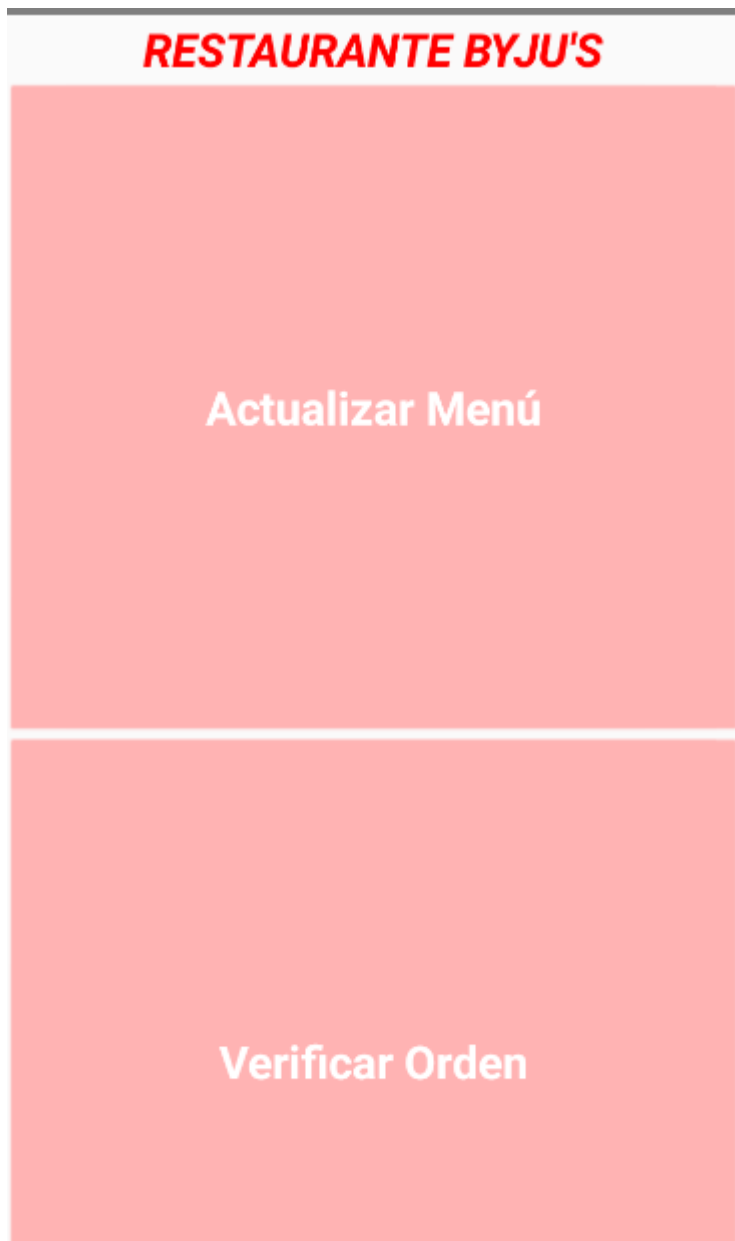


Nota Pantalla obtenida en Google Dialogflow

Previamente a la realización de la app, se creó un chatbot (sistema que ayuda a simular conversaciones) para ayudar al usuario a saber cuál es el menú que tendrá disponible el restaurante, así como algunos datos extra como lo son la cantidad de picante que cuenta cada platillo y porciones de los mismos.

Figura 19

Pantalla Inicio Admin -Administración-



Nota Pantalla obtenida en BlueStacks

Se determinó que se necesitarán 2 aplicaciones diferentes para la gestión del restaurante, una de ellas, es la app de administración, en donde el “restaurante” tiene acceso al menú y a la posibilidad de añadir nuevos platillos al mismo, así como sus precios; y una pantalla para verificar orden, en donde aparecerán los usuarios que hayan realizado su pedido en la app del cliente.

Figura 20

Pantalla para actualizar menú

Update Menu

Check Orders

Nombre del platillo :

Precio :

Añadir información del platillo

Menú :

pizza : 250

Tacos : 17

Refresco : 15

Pastel : 186

Malteada : 25

Hot dog : 34

Helado : 52

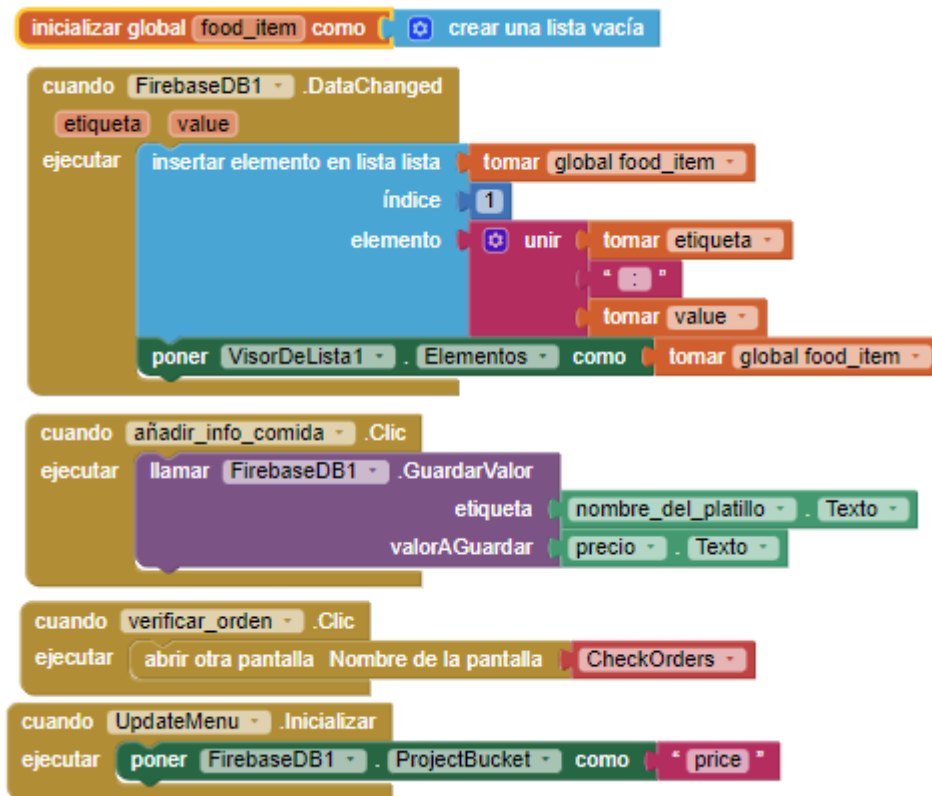
Hamburguesa : 120

Nota Pantalla obtenida en BlueStacks

Dentro de la sección de actualizar menú, se colocó en diseño la opción de añadir diferentes platillos y precios, los cuales son mostrados debajo, atendiendo a la consulta de la base de datos previamente creada. En esta pantalla también se encuentra un botón que permite movilizarnos a la sección de verificar pedido.

Figura 21

Código pantalla de actualizar menú



Nota Pantalla obtenida en MIT App Inventor 2

Una vez creado el diseño la aplicación, se debe hacer la conexión con la base de datos con el elemento Firebase, ya que se va a mostrar como un texto dentro de la pantalla para actualizar menú, colocamos un bloque que permite que cuando la base de datos sea actualizada (ya sea dentro de la página de Firebase como dentro de la app) se haga un reajuste al menú que nos aparece, separando el nombre del platillo con el precio.

Figura 22

Pantalla de verificar orden



Nota Pantalla obtenida en BlueStacks

La pantalla de verificación de orden, nos colocará en una lista los nombres de las personas que hayan hecho los diferentes pedidos dentro de nuestra app de cliente, uniendo el diseño con un código orientado a objetos de manera similar a la anterior, actualizando su contenido cada vez que se añaden los pedidos nuevos a la base de datos.

Figura 23

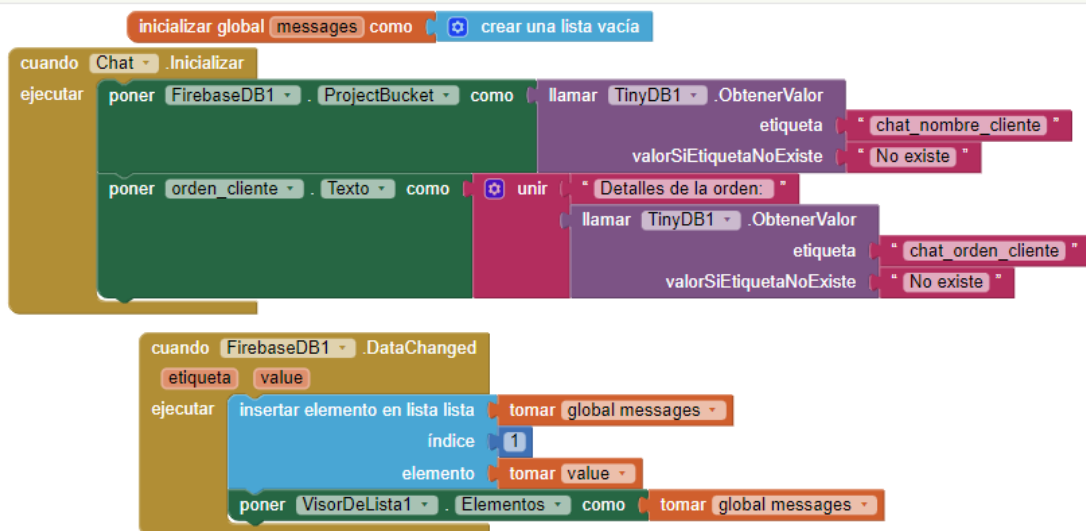
Chat



Nota Pantalla obtenida en BlueStacks

Si seleccionamos alguno de los nombres de las personas que realizaron un pedido, aparecerá la pantalla del chat con el usuario, en la parte superior, estará el detalle del pedido registro, iniciando por el/los platillo/s que seleccionó el cliente, siguiendo con los precios en orden y el número de platillos que seleccionó de cada uno de los mismos.

Figura 24 Código chat

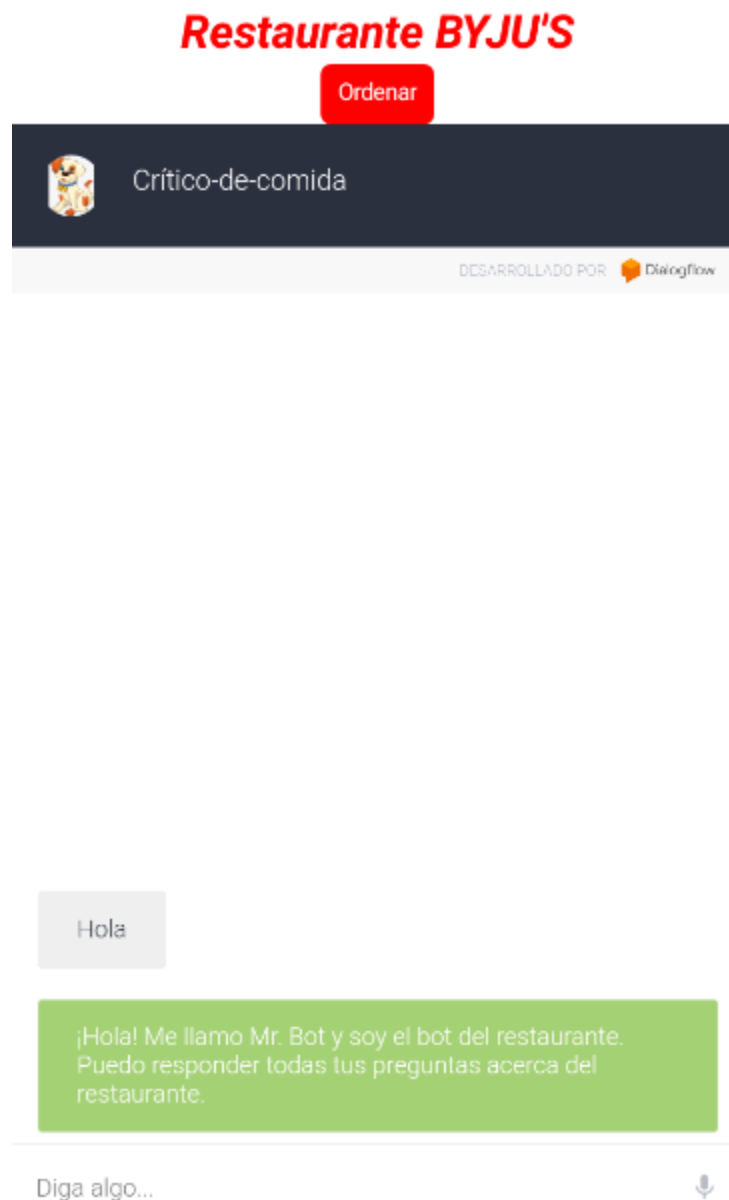


Nota Pantalla obtenida en MIT App Inventor 2

Lo primero es la conexión que se tendrá de la base de datos, una vez que se hace, al iniciar la pantalla del chat (al elegir el nombre del pedido) se imprimirá en una etiqueta los detalles del pedido realizado, así como la información del usuario que está realizando el mismo. Esta pantalla está conectada con otra con la intención de añadir comunicación entre el usuario y el chef, en caso de hacer alguna aclaración en la elaboración de los alimentos.

Figura 25

Pantalla Inicio Cliente

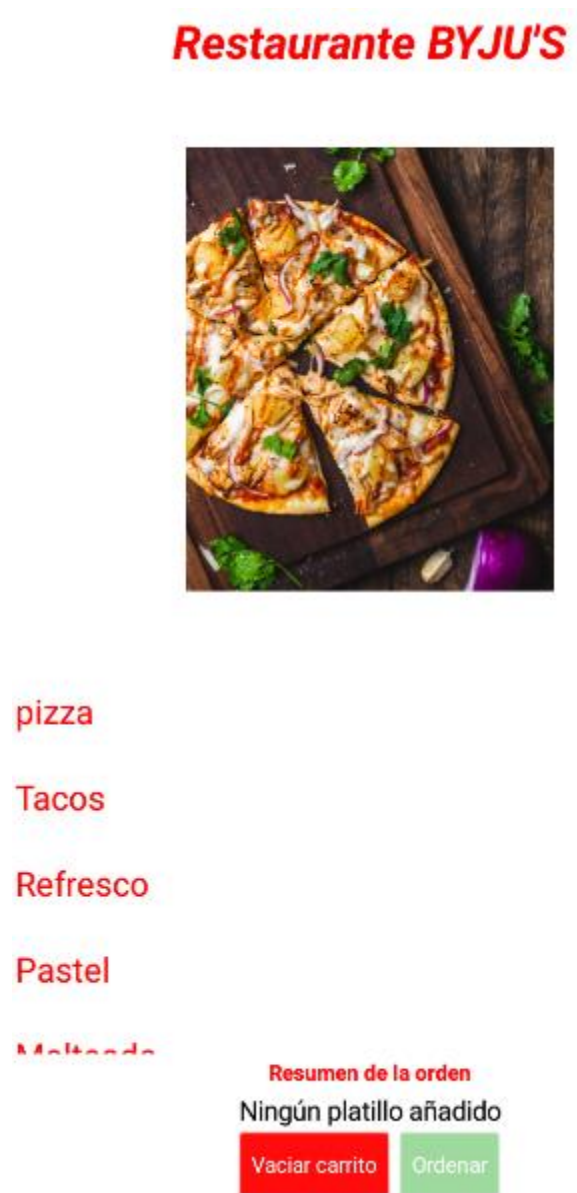


Nota Pantalla obtenida en BlueStacks

Lo primero en aparecer en la app del cliente es un visor web que nos permite tener al acceso al chatbot previamente creado, en donde mediante preguntas podremos saber cuál es el menú que tiene el restaurante, así como algunas inquietudes extra, una vez informados, el chatbot invita al usuario a utilizar el botón para ordenar que lo llevará a la sección del menú.

Figura 26

Pantalla del menú



Nota Pantalla obtenida en BlueStacks

Con la conexión a la base de datos, nos presentará el nombre del restaurante, así como alguna imagen (seleccionado por cada niño dependiendo de su creatividad), aparecerá el menú del restaurante, así como el carrito de compra que se puede eliminar al presionar el botón de vaciar carrito.

Figura 27

Código conexión base de datos



Nota Pantalla obtenida en MIT App Inventor 2

La información del carrito se guardará en un elemento de MIT App Inventor 2 llamado TinyDB el cual, es utilizado para compartir información entre las diferentes ventanas que compondrá la aplicación del cliente; en este caso se guardará el nombre, cantidad y precio de los platillos seleccionados.

Figura 28

Pantalla menú

Restaurante BYJU'S



Hamburguesa



Pesos:240

Confirma tu platillo

Regresar al menú

Resumen de la orden

Ningún platillo añadido

Vaciar carrito

Ordenar

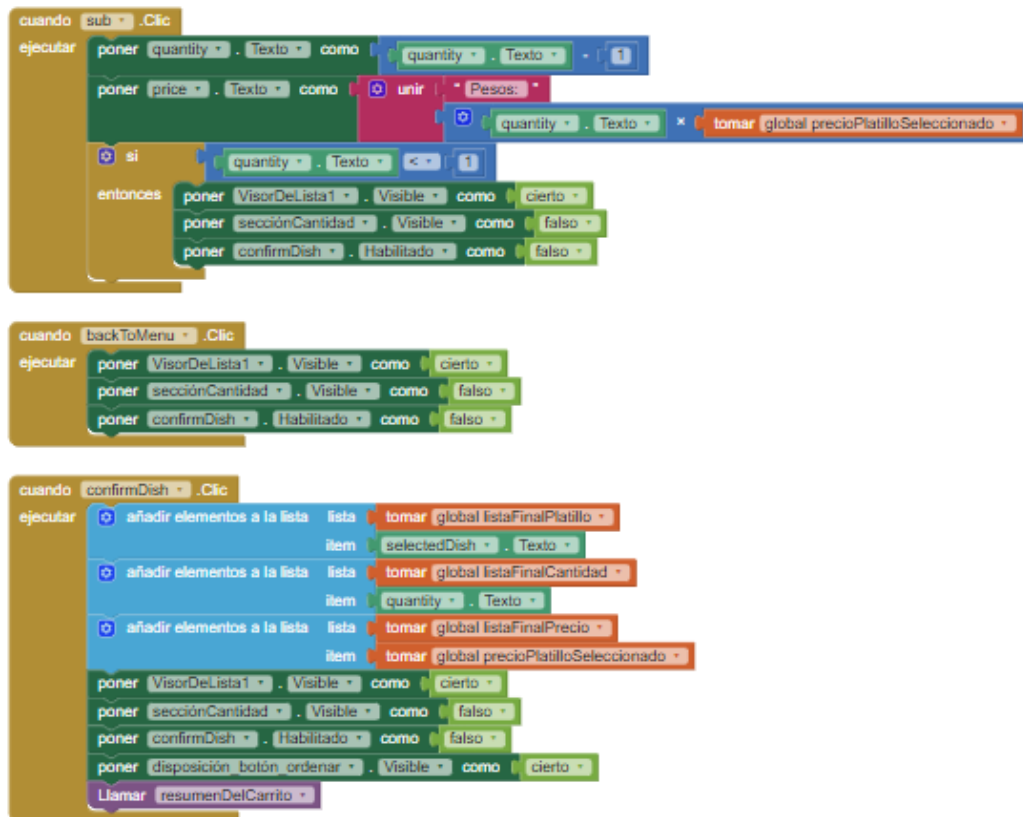
Nota Pantalla obtenida en BlueStacks

Al seleccionar alguna de las opciones de la lista del menú, se activará la sección para añadir el platillo, en ella, aparecerá el nombre del platillo, dos botones que nos permiten añadir o eliminar cantidad del platillo que se seleccionó, así como el costo total de los productos añadidos, si se está

de acuerdo con la selección, al presionar el botón de Confirmar pedido, se añadirá al carrito de compra.

Figura 29

Código añadir y eliminar producto



Nota Pantalla obtenida en MIT App Inventor 2

Para añadir o eliminar la cantidad de productos seleccionados, se utilizó un evento clic, en el cual se actualizaba dependiendo del botón que se presionaba (símbolo + para añadir productos o símbolo – para eliminarlos), al mismo tiempo la cantidad de elementos se multiplicaba por el precio de cada uno de ellos, para proporcionar el costo total de los alimentos. Si se presiona el botón para regresar el menú, se desactiva la opción para elegir la cantidad de productos y se volvía al menú, sin embargo, si se confirmaba, volvía al menú, después de añadir la lista del pedido al carrito.

Figura 30

Pantalla menú con carrito de compra

Restaurante BYJU'S



Pastel

Malteada

Hot dog

Helado

Hamburguesa

Resumen de la orden

Malteada Cantidad:1 Precio en pesos:25

Hamburguesa Cantidad:2 Precio en pesos:240

Vaciar carrito

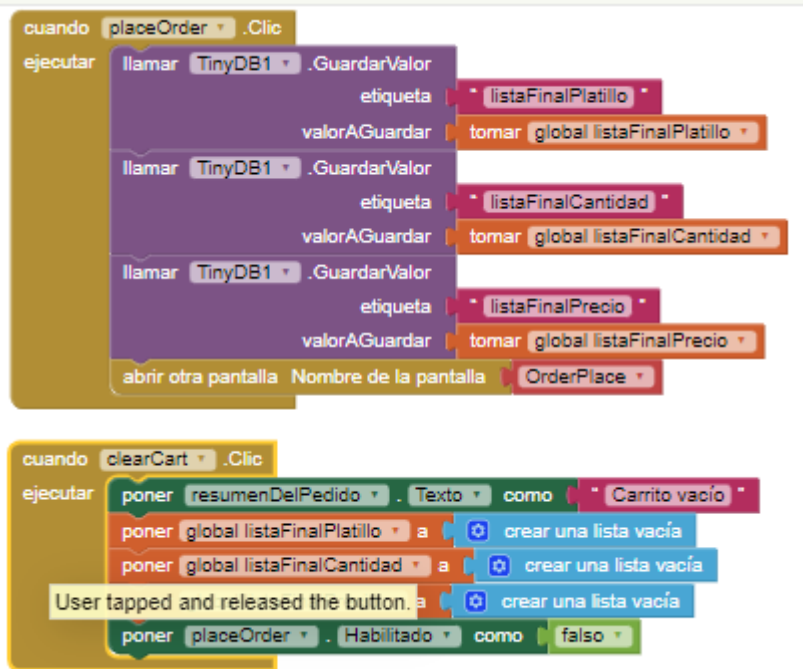
Ordenar

Nota Pantalla obtenida en BlueStacks

Al añadir los diferentes alimentos que conformarán el pedido, aparecerá un resumen con la información necesaria para estar seguros de la selección, en caso de no estarlo se debe presionar el botón de vaciar carrito y comenzar de nuevo con la selección de alimentos que se solicitará al restaurante.

Figura 31

Código del carrito de compra



Nota Pantalla obtenida en MIT App Inventor 2

Al vaciar el carrito de compra las listas donde se guarda la información del pedido son eliminadas, volviendo a quedar vacías, así como la etiqueta que muestra el mensaje “Carrito vacío”. En caso de estar seguros de la elección, el pedido es guardado en la TinyDB para su utilización y guardado en la base de datos en la siguiente pantalla.

Figura 32

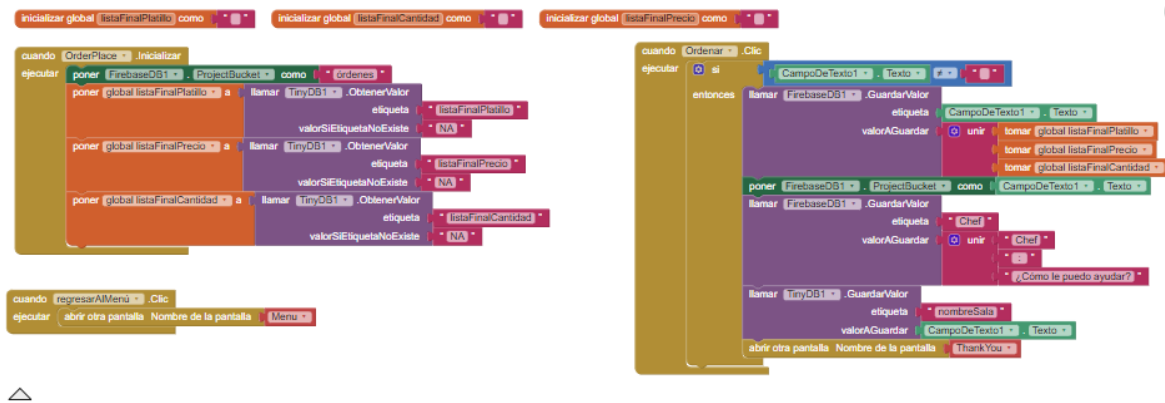
Pantalla de ordenar



Nota Pantalla obtenida en BlueStacks

Tras seleccionar los alimentos que se añadieron al pedido, el usuario será trasladado a la pantalla para ordenar, en la cual proporcionará su nombre para registrar el pedido y así aparecer en la sección de verificar pedidos en la app de administración, en caso de no estar seguros de la selección, se puede volver a la pantalla del menú,

Figura 33
Código pantalla ordenar



Nota Pantalla obtenida en MIT App Inventor 2

Dentro de variables se guarda la información almacenada en la pantalla anterior (platillo, cantidad y precio), en caso de que se perdiera alguna información se almacena como No Aplica (NA). Una vez que se presione el botón de ordenar se hará una verificación con condicional IF para verificar si la caja de texto donde va el nombre del usuario no está vacía, en caso de no estarlo, guarda la información en la base de datos para su posterior revisión.

Figura 34 Pantalla de chat

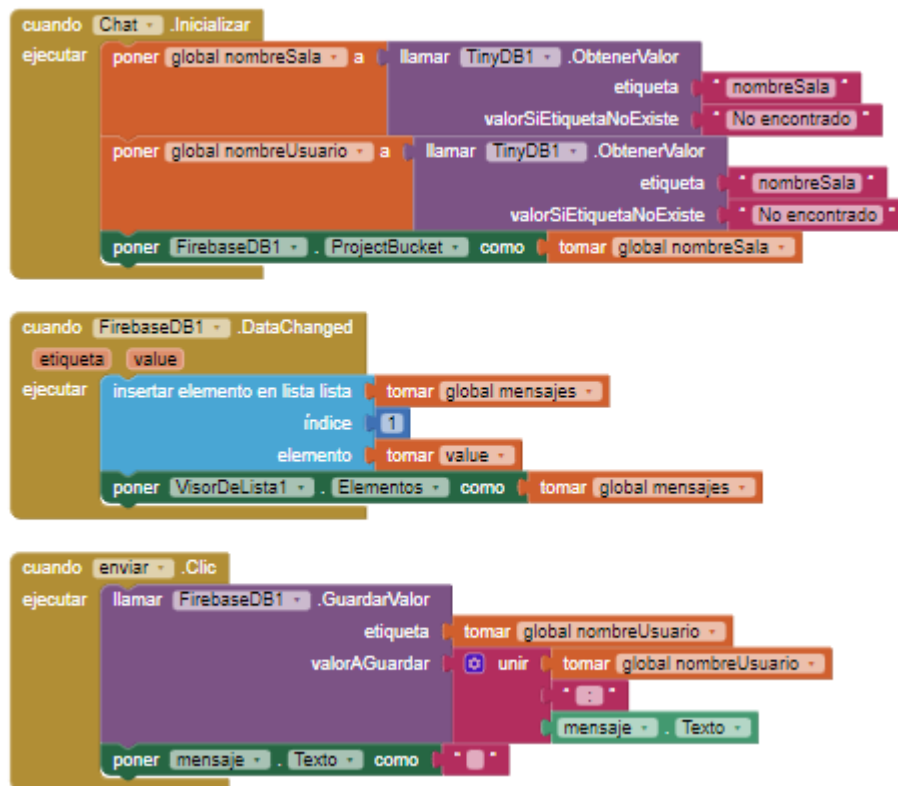


Nota Pantalla obtenida en BlueStack

Una vez registrado el pedido que realizó el usuario, se guarda en la base de datos, y se conecta con el chat con el chef, para tener comunicación directa con el restaurante, en caso de solicitar alguna preparación o la excepción de algún ingrediente, todos los mensajes se guardan en su propia sala de chat, así que los mensajes no se pueden mezclar.

Figura 35

Código de chat



Nota Pantalla obtenida en MIT App Inventor 2

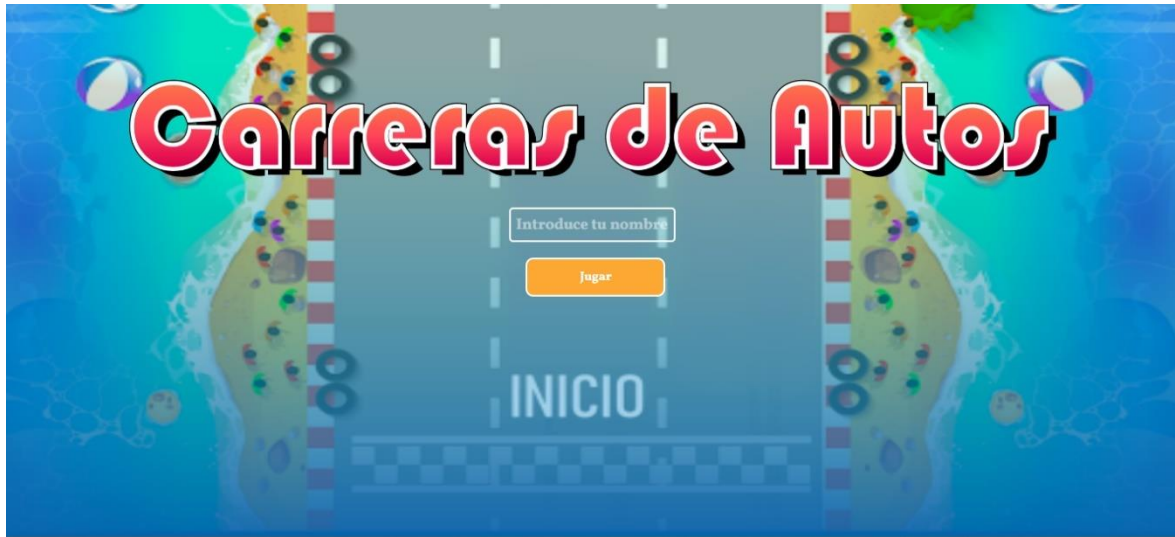
Se realiza la conexión con la base de datos, corroborando en primer lugar el nombre del usuario que realizó el pedido, en caso de que no se encuentre o haya existido un fallo al guardar esa información aparecerá como “No encontrado”. Al cambiar la base de datos, los mensajes se estarán guardando en la misma variable con la intención de mostrarlos en pantalla y al presionar el botón de enviar, se guardará el mensaje nuevo junto con los anteriores.

Juego de Carreras Multijugador PRO (12-15 años)

Durante el desarrollo de sus habilidades de programación, los adolescentes de entre 12 y 15 años realizan un juego multijugador síncrono, en el cual, en dos computadoras se hace el ingreso al juego, se añade el nombre de los jugadores y al colocarlo en la base de datos, los jugadores ven en tiempo real el avance de su contrincante y los puntos que van obteniendo.

Figura 36

Inicio del juego



Nota Pantalla obtenida gracias a Live Server

Al inicio del juego se solicitará a los jugadores registrar sus nombres para tener una mejor personalización del mismo, en juego está adaptado para dos jugadores, por lo cual, al registrar al primer jugador, esperará a que se una el segundo jugador, mientras aparecerá un mensaje con el nombre del jugador que mencionará que se espera a otro usuario.

Figura 37

Clase Form

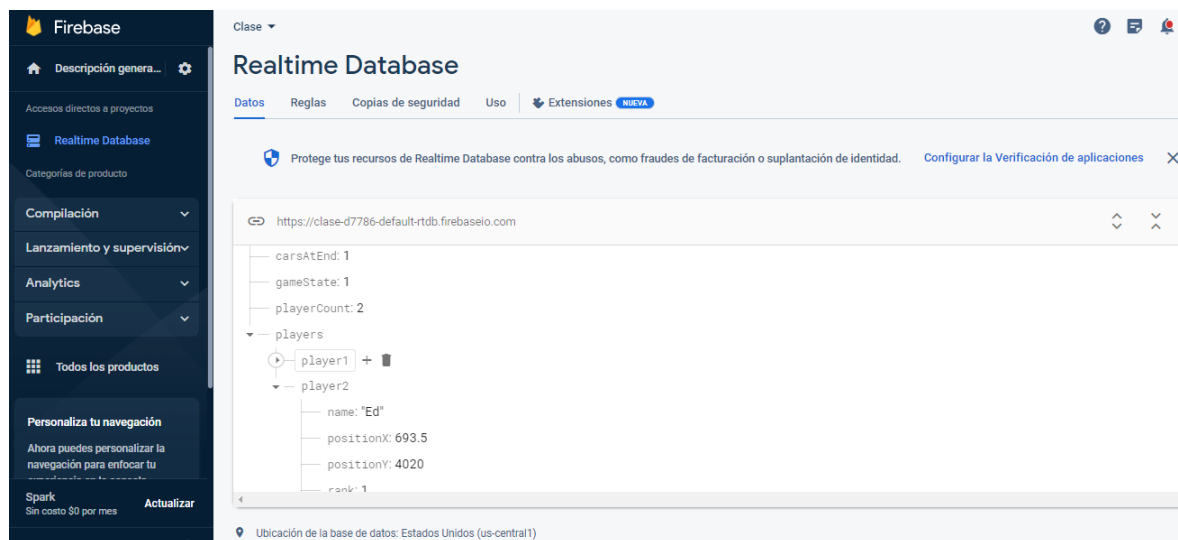
```
1  class Form {
2      constructor() {
3          this.input = createInput("").attribute("placeholder", "Introduce tu nombre");
4          this.playButton = createButton("Jugar");
5          this.titleImg = createImg("./assets/title.png", "título del juego");
6          this.greeting = createElement("h2");
7      }
8
9  >  setElementsPosition() { ...
14 }
15
16 >  setElementsStyle() { ...
21 }
22
23 >  hide() { ...
27 }
28
29  handleMousePressed() {
30      this.playButton.mousePressed(() => {
31          this.input.hide();
32          this.playButton.hide();
33          var message = `
34              Hola, ${this.input.value()}
35              </br>Espera a que se una otro jugador...`;
36          this.greeting.html(message);
37          playerCount += 1;
38          player.name = this.input.value();
39          player.index = playerCount;
40          player.addPlayer(); //aaa
41          player.updateCount(playerCount); // BP
42          player.getDistance(); //aaa
43      });
44  }
```

Nota Código realizado en Visual Studio Code

Dentro de la clase del formulario, se crea en el constructor los elementos HTML necesarios en el código JavaScript, se personaliza la posición y el estilo gracias a las diferentes funciones dentro de la clase. Se añade una función que, al presionar el botón de Enviar, se oculten los elementos como la caja de texto, título y botón, muestre el mensaje para esperar al jugador 2 y actualiza las variables necesarias para el funcionamiento del juego, como el playerCount (contador de jugadores, que evita que entren más de 2 personas) que servirán para distintas clases dentro del juego.

Figura 38

Base de datos del juego



Nota Pantalla obtenida en Google Firebase

Se utiliza una base de datos alojada en el servidor de Firebase, donde se guardará toda la información del juego, ya que al ser un juego online multijugador y síncrono, deben actualizarse las posiciones a partir de la información registrada. Añadimos las secciones necesarias como players, gameState y playerCount que pueden ser revisadas en cualquier momento por alguna de las clases del juego.

Figura 39

Index del juego

```
30 <script>
31 // Configuración de Firebase de tu aplicación web
32 const firebaseConfig = {
33   apiKey: "AIzaSyAkC-8794HRJrLKKn7G1L25J7pykRwguhI",
34   authDomain: "clase-d7786.firebaseio.com",
35   databaseURL: "https://clase-d7786-default-rtdb.firebaseio.com",
36   projectId: "clase-d7786",
37   storageBucket: "clase-d7786.appspot.com",
38   messagingSenderId: "84979067607",
39   appId: "1:84979067607:web:94dc4734472fd8f4529e92"
40 };
41
42 // Inicializar Firebase
```

Nota Pantalla obtenida en Visual Studio Code

Se añade la información de la conexión a la base de datos dentro del index del juego, se utiliza un script y la información que la plataforma nos proporciona una vez que se selecciona la opción para añadir a una app; la información se puede adaptar para conexión con IOs, Android, Web y Unity, facilitando la unión de las plataformas.

Figura 40

Juego



Nota Pantalla obtenida gracias a Live Server

Al contar con los 2 jugadores, se modifica el estado del juego, permitiendo de esta manera ver todos los elementos que componen el mismo. Al ser una carrera, se cuentan con autos que son controlados por el jugador (se marcan con un punto rojo para que el jugador sepa cuál es el que controla en todo momento), se colocaron obstáculos, así como objetos que proporcionan un aumento en la gasolina (que permite avanzar más distancia) y puntos (que influyen en la tabla de posiciones).

Figura 41

Código de clase Player

```
32  getDistance() {
33      var playerDistanceRef = database.ref("players/player" + this.index);
34      playerDistanceRef.on("value", data => {
35          var data = data.val();
36          this.positionX = data.positionX;
37          this.positionY = data.positionY;
38      });
39  }
40
41  getCount() {
42      var playerCountRef = database.ref("playerCount");
43      playerCountRef.on("value", data => {
44          playerCount = data.val();
45      });
46  }
47
48  updateCount(count) {
49      database.ref("/").update({
50          playerCount: count
51      });
52  }
53
54  update() {
55      var playerIndex = "players/player" + this.index;
56      database.ref(playerIndex).update({
57          positionX: this.positionX,
58          positionY: this.positionY,
59          rank: this.rank,
60          score: this.score,
61          life: this.life
62      });
63  }
```

Nota Pantalla obtenida en Visual Studio Code

Dentro de la clase player es donde se va a modificar toda la información que afecte directamente a los jugadores, en ella se tiene la obtención de sus posiciones en la base de datos que se guardarán para que el usuario contrario pueda ver sus movimientos, se asegura que al presionar el botón de Enviar (al inicio del juego) añade al jugador y modifique el playerCount (conteo de jugadores) que, a su vez, cambiará el estado del juego una vez llegado a 2. La función de update, nos permite añadir la diferente información necesaria para el jugador, entre ellas sus

posiciones, la posición en la tabla de puntuación, la cantidad de gasolina que tiene el auto, la salud del vehículo, así como los puntos recolectados.

Figura 42

Código de creación de sprites

```
29  start() {
30      player = new Player();
31      playerCount = player.getCount();
32
33      form = new Form();
34      form.display();
35
36      car1 = createSprite(width / 2 - 50, height - 100);
37      car1.addImage("car1", car1_img);
38      car1.scale = 0.07;
39      car1.addImage("blast", blastImage); //C42 //AA
40
41      car2 = createSprite(width / 2 + 100, height - 100);
42      car2.addImage("car2", car2_img);
43      car2.scale = 0.07;
44      car2.addImage("blast", blastImage); //C42//AA
45
46      cars = [car1, car2];
47
48      fuels = new Group();
49      powerCoins = new Group();
50      obstacle1 = new Group();
51      obstacle2 = new Group();
52      var obstacle1Positions = [
53          { x: width / 2 - 150, y: height - 1300, image: obstacle1Image },
54          { x: width / 2 + 250, y: height - 1800, image: obstacle1Image },
55          { x: width / 2 - 180, y: height - 3300, image: obstacle1Image },
56
57          { x: width / 2 - 150, y: height - 4300, image: obstacle1Image },
58          { x: width / 2, y: height - 5300, image: obstacle1Image },
59      ];
60  }
```

Nota Pantalla obtenida en Visual Studio Code

Al ser un código estructurado, se decidió separar cada elemento que influye en el juego en diferentes clases, por lo tanto, en la sección de game se añadía todo lo que involucraba al juego y al funcionamiento del mismo. En la función start, se creaban los diferentes sprites (objetos a los que se les proporciona algún comportamiento), añadiendo la imagen y las características que se necesitan; en el caso de los obstáculos se les proporciona una lista de posiciones en las que pueden aparecer al azar, para que se mantengan en la zona de la pista de carreras.

Figura 43

Código de grupos de sprites

```
86  ✓   this.addSprites(  
87      obstacle2,  
88      obstacle2Positions.length,  
89      obstacle2Image,  
90      0.04,  
91      obstacle2Positions  
92  );  
93  }  
94  
95  ✓   addSprites(spriteGroup, numberOfSprites, spriteImage, scale, positions = []) {  
96  ✓   for (var i = 0; i < numberOfSprites; i++) {  
97      var x, y;  
98  ✓   if (positions.length > 0) {  
99      x = positions[i].x;  
100     y = positions[i].y;  
101     spriteImage = positions[i].image;  
102  ✓   } else {  
103     x = random(width / 2 + 150, width / 2 - 150);  
104     y = random(-height * 4.5, height - 400);  
105     }  
106     var sprite = createSprite(x, y);  
107     sprite.addImage("sprite", spriteImage);  
108  
109     sprite.scale = scale;  
110     spriteGroup.add(sprite);  
111     }  
112     }  
113
```

Nota Pantalla obtenida en Visual Studio Code

Para añadir los diferentes sprites con los que cuenta el juego y atendiendo al principio DRY (Don't Repeat Yourself) se añade una función con la cual añadir todos los elementos. En ella se utiliza el bucle FOR para repetir la creación de los objetos, se determina si es un obstáculo o un potenciador, ya que los obstáculos cuentan con unas posiciones específicas para aparecer, en ambos casos, aparecen de manera aleatoria en las zonas de la pista de carreras, se añaden al grupo, se coloca la imagen y cambia la escala de la misma para evitar que llegue a cubrir toda la pantalla.

Figura 44

Código de combustible y vida de los jugadores

```
157 // Usa datos de la base de datos para mostrar los autos en dirección x y y
158 var x = allPlayers[plr].positionX;
159 var y = height - allPlayers[plr].positionY;
160
161 // Guarda el valor de player.life en la variable temporal "currentlife"
162 var currentlife = allPlayers[plr].life;
163
164 if (currentlife <= 0) {
165     cars[index - 1].changeImage("blast");
166     cars[index - 1].scale = 0.3;
167 }
168 cars[index - 1].position.x = x;
169 cars[index - 1].position.y = y;
170
171 if (index === player.index) {
172     stroke(10);
173     fill("red");
174     ellipse(x, y, 60, 60);
175
176     this.handleFuel(index);
177     this.handlePowerCoins(index);
178     this.handleObstacleCollision(index); //C41//AA
179     if (player.life <= 0) {
180         this.blast = true;
181         this.playerMoving = false;
182     }
183
184     // Cambiando la posición de la cámara en dirección y.
185     camera.position.y = cars[index - 1].position.y;
186 }
187 }
```

Nota Pantalla obtenida en Visual Studio Code

Se guarda la información de los jugadores en un arreglo, se les proporcionará una posición de inicio y si aún se cuenta con salud en el vehículo (no se ha destruido al chocar con los obstáculos). Se actualiza de manera individual la información de la posición de los jugadores, uniéndolos a cada una de sus secciones en la base de datos, se añade el círculo rojo que nos muestra el auto activo por cada jugador y se añade el movimiento de la cámara para que vaya siguiendo al jugador.

Figura 45

Código de bonificaciones y botón de reinicio

```
232 handlePowerCoins(index) {
233   cars[index - 1].overlap(powerCoins, function(collector, collected) {
234     player.score += 21;
235     player.update();
236     // "collected" es el sprite en el grupo de coleccionables que desencadenaron
237     // el evento
238     collected.remove();
239   });
240 }
241
242 handleResetButton() {
243   this.resetButton.mousePressed(() => {
244     database.ref("/").set({
245       carsAtEnd: 0,
246       playerCount: 0,
247       gameState: 0,
248       palyers: {}
249     });
250     window.location.reload();
251   });
252 }
253
254 showFuelBar() {
255   push();
256   image(fuelImage, width / 2 - 130, height - player.positionY - 350, 20, 20);
257   fill("white");
258   rect(width / 2 - 100, height - player.positionY - 350, 185, 20);
259   fill("#ffc400");
260   rect(width / 2 - 100, height - player.positionY - 350, player.fuel, 20);
261   noStroke();
262   pop();
263 }
```

Nota Pantalla obtenida en Visual Studio Code

Para añadir los puntos y la cantidad de gasolina por jugador, se utiliza el término `overlap` (pasar sobre el objeto) para que parezca que el jugador realmente recolecta los objetos, al hacerlo modifica la variable involucrada y actualiza la base de datos para que el jugador contrario pueda ver los cambios. Se cuenta con un botón de reinicio que nos permite eliminar toda la información en la base de datos y recargar la página para comenzar de nuevo, en este caso se utiliza una función `arrow` (función compacta y limitada) dentro de la función para contar con el acceso. Para mostrar la barra de gasolina y salud, se crea un método `push` que nos permite grabar el comportamiento de algunos objetos, se añaden las imágenes que nos ayudan a distinguir las barras y los rectángulos

que las componen, terminando con un pop que acaba con el guardado del comportamiento, lo cual permite que cada objeto funcione correctamente.

Figura 46

Código de tabla de puntuación

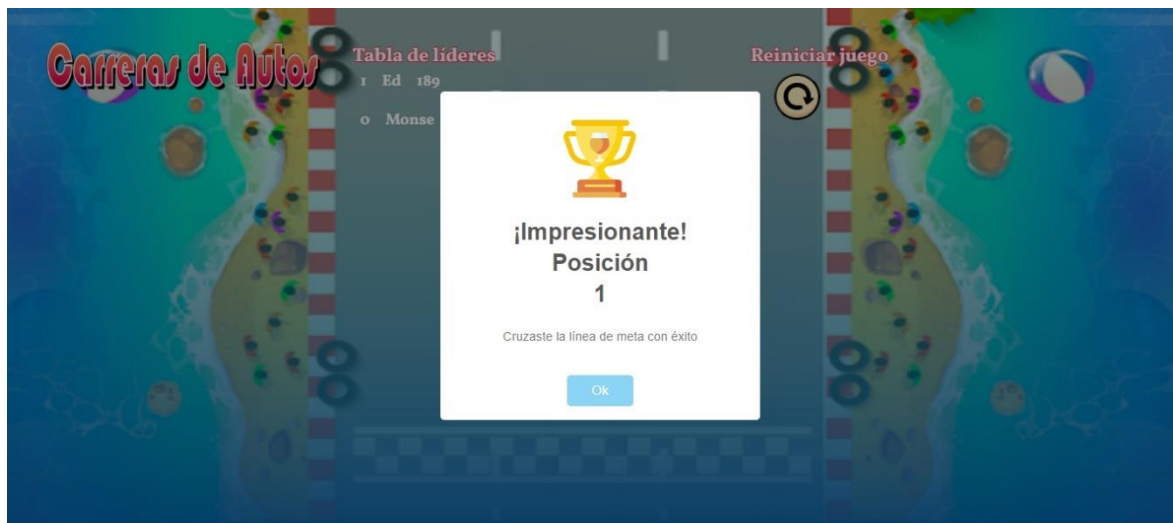
```
276 showLeaderboard() {
277     var leader1, leader2;
278     var players = Object.values(allPlayers);
279     if (
280         (players[0].rank === 0 && players[1].rank === 0) ||
281         players[0].rank === 1
282     ) {
283         // &emsp; Esta etiqueta se utiliza para mostrar cuatro espacios.
284         leader1 =
285             players[0].rank +
286             "&emsp;" +
287             players[0].name +
288             "&emsp;" +
289             players[0].score;
290
291         leader2 =
292             players[1].rank +
293             "&emsp;" +
294             players[1].name +
295             "&emsp;" +
296             players[1].score;
297     }
298 }
```

Nota Pantalla obtenida en Visual Studio Code

Para la creación de la tabla de puntuación se crean 2 variables que almacenarán a las posiciones de los jugadores, después se añade una nueva variable que contendrá la información de los jugadores como Object (devuelve una matriz de valores de propiedad numerable en mismo orden en el que fueron guardados) y así revisa cual es el jugador que se encuentra en la primera posición dependiendo del número de puntos que haya obtenido. Se añaden los objetos HTML en donde se mostrará la puntuación, utilizando una condicional en la que, si el jugador 1 tiene la primera posición o no se han conseguido puntos aún, se mostrará como el usuario con la posición uno, o en caso contrario aparecerán en orden invertido.

Figura 47

Fin del juego



Nota Pantalla obtenida gracias a Live Server

Hay tres maneras de terminar el juego para los usuarios, la primera es que se queden sin salud, la segunda acabar con la gasolina y la tercera es terminar con la carrera, para ello se comprueban todos los valores antes mencionados y proporciona una alerta dependiendo de la situación presentada.

Figura 48

Código de alertas

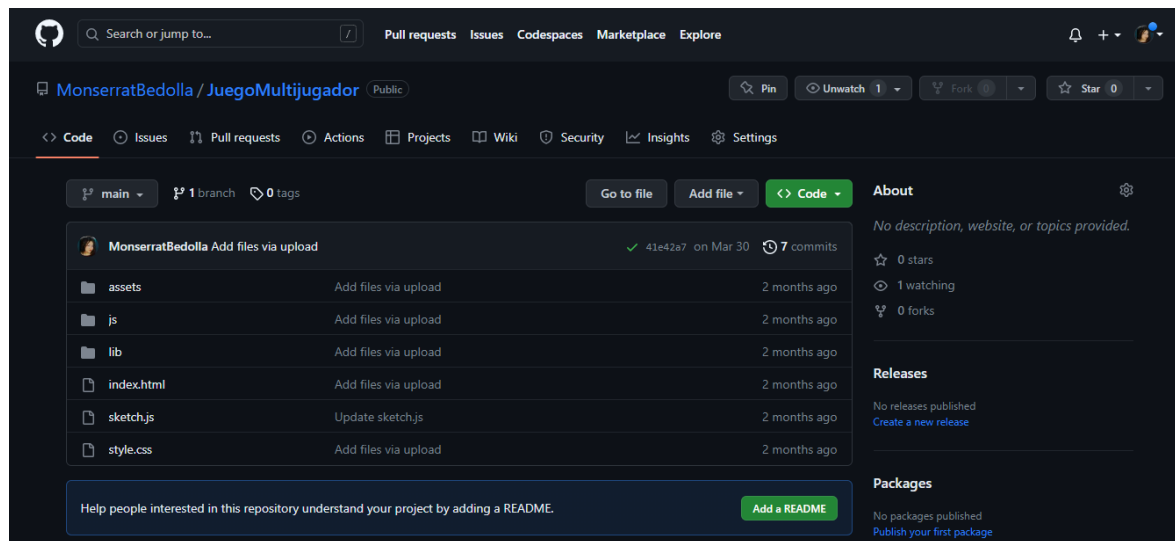
```
363 showRank() {  
364     swal({  
365         title: `¡Impresionante!${"\n"}Posición${"\n"}${player.rank}`,  
366         text: "Llegaste a la línea de meta exitosamente",  
367         imageUrl:  
368             "https://raw.githubusercontent.com/vishalgaddam873/p5-multiplayer-car-race-game/master/assets/cup.png",  
369         imageSize: "100x100",  
370         confirmButtonText: "Ok"  
371     });  
372 }  
373  
374 gameOver() {  
375     swal({  
376         title: `Fin del juego`,  
377         text: "¡Ups! ¡Perdiste la carrera!",  
378         imageUrl:  
379             "https://cdn.shopify.com/s/files/1/1061/1924/products/Thumbs_Down_Sign_Emoji_Icon_ios10_grande.png",  
380         imageSize: "100x100",  
381         confirmButtonText: "Gracias por jugar"  
382     });  
383 }  
384  
385
```

Nota Pantalla obtenida en Visual Studio Code

Se crean 2 SweetAlerts (manera de mostrar mensajes en pantalla capaz de personalizarse) para añadir los mensajes tanto de victoria como de derrota en el juego, lo primero en hacerse, es añadir el script del Swal (SweetAlert) en el index para poderlo utilizar durante el proceso, para poderlo personalizar añadimos un mensaje con el nombre del jugador y un texto que nos indique si se ha ganado o perdido la carrera, se añade una pequeña imagen para que el diseño sea mucho más estético, configuramos la imagen y añadimos un botón que nos permite cerrar la alerta y poder reiniciar el juego.

Figura 49

Subir carpeta a Github



Nota Pantalla obtenida de GitHub

Una vez que se comprueba que el juego funciona se aloja en un servidor como Github para que se pueda jugar de manera online. Se debe contar con una cuenta gratuita, creamos un nuevo repositorio donde se almacenará el juego, se suben los archivos del mismo, de la misma manera en la que se crearon, una vez realizada y los archivos en la nueva se va a configuraciones y en la sección de Pages, se añade la característica para que el link proporcionado sea ejecutable sin la necesidad de un compilador.

App Biblioteca digital PRO (12-15 años)

Se realizó una aplicación multiplataforma (disponible para IOs y Andorid) en la cual se hace el registro de una biblioteca escolar, se ingresan los libros dentro de una base de datos y se comparte la información de los alumnos a la misma; se realizan 3 pantallas, la pantalla principal que deriva en las otras dos, una para realizar el préstamo y devolución de libros y otra para la búsqueda de las transacciones realizadas.

Figura 50

Código app principal

```
componentDidMount() {  
  this.loadFonts();  
}  
  
render() {  
  const { fontLoaded } = this.state;  
  if (fontLoaded) {  
    return <AppContainer />;  
  }  
  return null;  
}  
}  
  
const AppSwitchNavigator = createSwitchNavigator(  
  {  
    Login: {  
      screen: LoginScreen  
    },  
    BottomTab: {  
      screen: BottomTabNavigator  
    }  
  },  
  {  
    initialRouteName: "Login"  
  }  
);  
  
const AppContainer = createAppContainer(AppSwitchNavigator);
```

Nota Pantalla obtenida de Visual Studio Code

Se realiza la importación de los componentes necesarios para el funcionamiento de la app, utilizamos el ciclo de vida de react para poder cargar las fuentes necesarias para el mejor diseño de la aplicación, y se llama al componente de navegación por pestañas, el cual, une (en la parte inferior de la pantalla) las pantallas, para poder ser utilizadas posteriormente.

Figura 51

Componente de navegación

```
import Ionicons from "react-native-vector-icons/Ionicons";

import TransactionScreen from "../screens/Transaction";
import SearchScreen from "../screens/Search";

const Tab = createBottomTabNavigator();

export default class BottomTabNavigator extends Component {
  render() {
    return (
      <NavigationContainer>
        <Tab.Navigator
          screenOptions={({ route }) => ({
            tabBarIcon: ({ focused, color, size }) => {
              let iconName;

              if (route.name === "Transaction") {
                iconName = "book";
              } else if (route.name === "Search") {
                iconName = "search";
              }

              // Puedes devolver cualquier componente que te guste aquí.
              return (
                <Ionicons
                  name={iconName}
                  size={size}
                  color={color}
                  //size={size}
                />
              );
            }
          })
        />
      );
  }
}
```

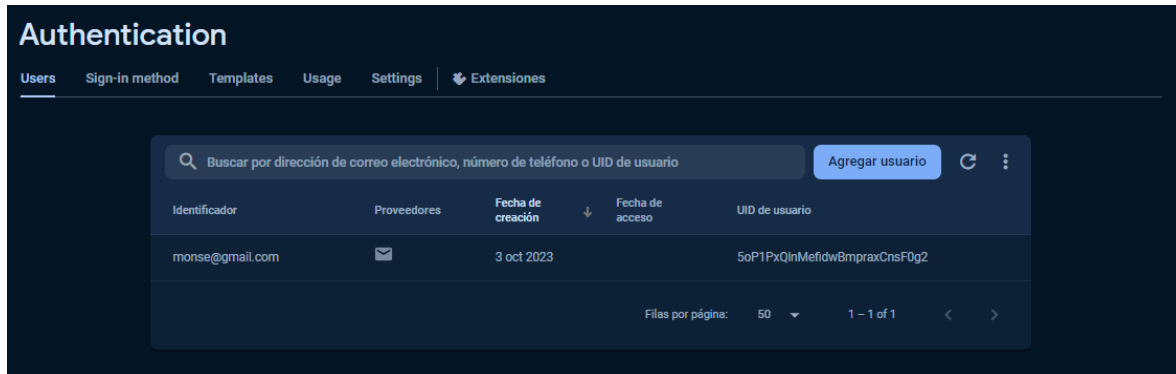
Nota Pantalla obtenida de Visual Studio Code

Dentro del componente de la navegación creamos el contenedor de las pantallas que de la misma manera nos permite modificar su diseño con iconos que nos ayuden identificar de una

manera más clara para que sirva cada pantalla, por ello, se crea la barra inferior en donde se mostrarán las pestañas que nos ayudan a cambiar de pantallas.

Figura 52

Autentificador



Nota Pantalla obtenida de Firebase

Dentro del proyecto que se crea en Firebase se tiene acceso a sus distintas herramientas, entre ellas se encuentra la sección de Authentication, la cual nos permite añadir algún correo (existente o no) y contraseña, con ello, se revisará y nos permitirá el acceso a la base de datos o no.

Figura 53

Código de inicio de sesión

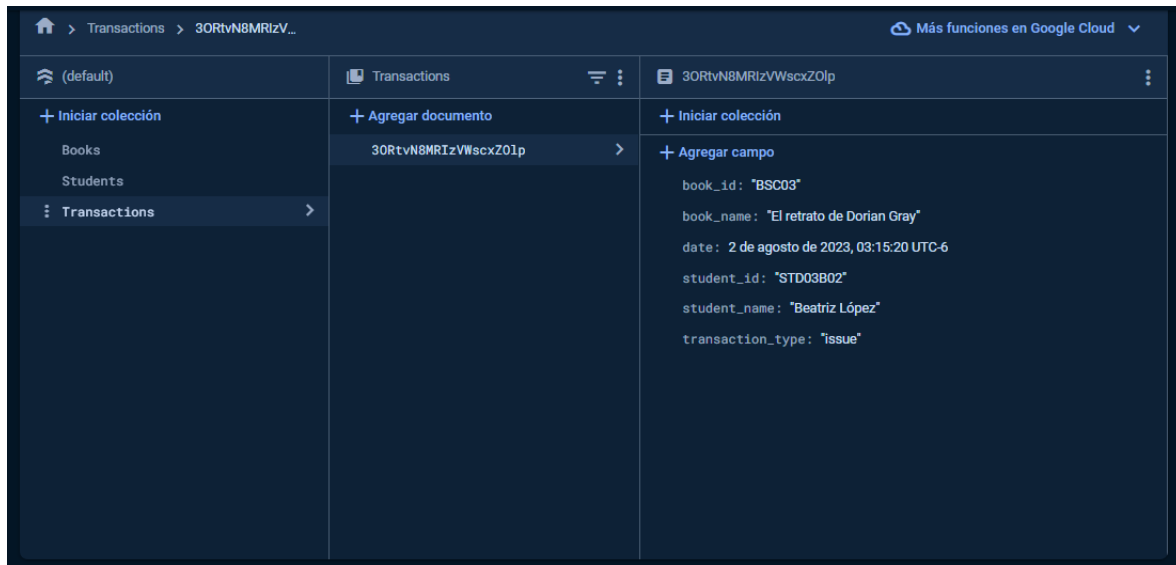
```
const { email, password } = this.state;
return (
  <KeyboardAvoidingView behavior="padding" style={styles.container}>
    <ImageBackground source={bgImage} style={styles.bgImage}>
      <View style={styles.upperContainer}>
        <Image source={appIcon} style={styles.appIcon} />
        <Image source={appName} style={styles.appName} />
      </View>
      <View style={styles.lowerContainer}>
        <TextInput
          style={styles.textinput}
          onChangeText={text => this.setState({ email: text })}
          placeholder="Escribe tu correo electrónico"
          placeholderTextColor="#FFFFFF"
          autoFocus
        />
        <TextInput
          style={[styles.textinput, { marginTop: 20 }]}
          onChangeText={text => this.setState({ password: text })}
          placeholder="Escribe tu contraseña"
          placeholderTextColor="#FFFFFF"
          secureTextEntry
        />
        <TouchableOpacity
          style={styles.button, { marginTop: 20 }}
          onPress={() => this.handleLogin(email, password)}
        >
          <Text style={styles.buttonText}>Iniciar sesión</Text>
        </TouchableOpacity>
      </View>
    </ImageBackground>
  </KeyboardAvoidingView>
)
```

Nota Pantalla obtenida de Visual Studio Code

La pantalla de inicio de sesión nos permite ingresar un correo y contraseña (previamente añadido a la autenticación de nuestra base de datos), para ello, se crean los componentes necesarios para llevar a cabo esta tarea, una vez que se realiza, se corrobora que la información sea correcta y por lo tanto tiene acceso al resto de pantallas.

Figura 54

Creación de la base de datos



Nota Pantalla obtenida de Firebase

Para tener un mejor control de la información que se tiene, se utilizará una base de datos, creada en Firestore, que ayuda a colocar los elementos en diferentes colecciones las cuales son más fáciles de modificar dentro de React Native. Se crean 3 colecciones diferentes, una donde se guarde la información de los libros, alumnos y transacciones, dentro se coloca los diferentes datos que ayudarán a identificarlos.

Figura 55

Código de pantalla de búsqueda

```
const { lastVisibleTransaction, allTransactions } = this.state;
if (enteredText[0] === "B") {
  const query = await db
    .collection("transactions")
    .where("bookId", "==", text)
    .startAfter(lastVisibleTransaction)
    .limit(10)
    .get();
  query.docs.map(doc => {
    this.setState({
      allTransactions: [...this.state.allTransactions, doc.data()],
      lastVisibleTransaction: doc
    });
  });
} else if (enteredText[0] === "S") {
  const query = await db
    .collection("transactions")
    .where("bookId", "==", text)
    .startAfter(this.state.lastVisibleTransaction)
    .limit(10)
    .get();
  query.docs.map(doc => {
    this.setState({
      allTransactions: [...this.state.allTransactions, doc.data()],
      lastVisibleTransaction: doc
    });
  });
}
};
```

Nota Pantalla obtenida de Visual Studio Code

Para una búsqueda más eficiente, se coloca la información de la caja de texto (donde se realizará la búsqueda) en mayúsculas, ya que es como fueron registrados en la base de datos, aunado a esto, se revisa la palabra seleccionada, para poder identificarlo correctamente, se utiliza la primera letra de la búsqueda para poder discernir si se tiene que ingresar en la colección de Estudiantes o en la de los Libros; una vez se realiza, se hará una lista de las últimas transacciones que involucren el elemento buscado.

Figura 56

Código permiso de cámara

```
getCameraPermissions = async domState => {
  const { status } = await Permissions.askAsync(Permissions.CAMERA);

  this.setState({
    /*status === "granted" es verdadero cuando el usuario da su permiso.
    | status === "granted" es falso cuando el usuario no da su permiso.
    */
    hasCameraPermissions: status === "granted",
    domState: domState,
    scanned: false
  });
};

handleBarcodeScanned = async ({ type, data }) => {
  const { domState } = this.state;

  if (domState === "bookId") {
    this.setState({
      bookId: data,
      domState: "normal",
      scanned: true
    });
  } else if (domState === "studentId") {
    this.setState({
      studentId: data,
      domState: "normal",
      scanned: true
    });
  }
};
```

Nota Pantalla obtenida de Visual Studio Code

La pantalla de transacciones es la que contiene el mayor peso del código, ya que se tienen que realizar varios procesos, entre ellos, el regreso y préstamo de los libros, para ello, añadimos un comando para poder leer los códigos QR o de barras que pueden contener los libros o las credenciales de los alumnos, así que se añade el permiso para utilizar la cámara con el uso del ciclo de vida.

Figura 57

Código para obtener detalles de los libros

```
getBookDetails = bookId => {  
  bookId = bookId.trim();  
  db.collection("books")  
    .where("book_id", "==", bookId)  
    .get()  
    .then(snapshot => {  
      snapshot.docs.map(doc => {  
        this.setState({  
          bookName: doc.data().book_details.book_name  
        });  
      });  
    });  
};
```

Nota Pantalla obtenida de Visual Studio Code

Para obtener la información guardada en la base de datos desde la sección de transacciones, guardamos la información proporcionada por los códigos QR o de barras y se realiza una búsqueda especializada a partir del identificador ya sea de los alumnos como de los libros, donde almacenará información para su posterior uso.

Figura 58

Código de devolución del libro

```
initiateBookReturn = async (bookId, studentId, bookName, studentName) => {  
  //Añade una transacción.  
  db.collection("transactions").add({  
    student_id: studentId,  
    student_name: studentName,  
    book_id: bookId,  
    book_name: bookName,  
    date: firebase.firestore.Timestamp.now().toDate(),  
    transaction_type: "return"  
  });  
  //Cambia el estatus del libro.  
  db.collection("books")  
    .doc(bookId)  
    .update({  
      is_book_available: true  
    });  
  //Cambia el número de libros prestados del alumno.  
  db.collection("students")  
    .doc(studentId)  
    .update({  
      number_of_books_issued: firebase.firestore.FieldValue.increment(-1)  
    });  
  
  // Actualizando el estado local.  
  this.setState({  
    bookId: "",  
    studentId: ""  
  });  
};
```

Nota Pantalla obtenida de Visual Studio Code

Una vez que se tiene la información de los libros y estudiantes se tiene que revisar el tipo de transacción a realizar, en este caso, la devolución, para ello se debe añadir a la colección de Transacciones la información necesaria, (en el caso del ID de transacción se coloca de manera automática, por ello no se añade en la actualización de la base); así mismo, se tiene que cambiar la disponibilidad del libro para que algún estudiante pueda utilizarlo más tarde y cambiar el número de libros prestados al alumno, ya que se tiene un tope de préstamos por alumno.

Figura 59

Código de tipo de transacción y elegibilidad

```
if (!transactionType) {
  this.setState({ bookId: "", studentId: "" });
  // Solo para usuarios de Android.
  // ToastAndroid.show("El libro no existe en la base de datos de la biblioteca.", ToastAndroid.SHORT);
  Alert.alert("El libro no existe en la base de datos de la biblioteca.");
} else if (transactionType === "issue") {
  var isEligible = await this.checkStudentEligibilityForBookIssue(
    studentId
  );

  if (isEligible) {
    var { bookName, studentName } = this.state;
    this.initiateBookIssue(bookId, studentId, bookName, studentName);
  }
  // Solo para usuarios de Android.
  // ToastAndroid.show("Se le prestó el libro al alumno.", ToastAndroid.SHORT);
  Alert.alert("Se le prestó el libro al alumno.");
} else {
  var isEligible = await this.checkStudentEligibilityForBookReturn(
    bookId,
    studentId
  );

  if (isEligible) {
    var { bookName, studentName } = this.state;
    this.initiateBookReturn(bookId, studentId, bookName, studentName);
  }
  // Solo para usuarios de Android.
  // ToastAndroid.show("Se devolvió el libro a la biblioteca.", ToastAndroid.SHORT);
  Alert.alert("Se devolvió el libro a la biblioteca.");
}
};
```

Nota Pantalla obtenida de Visual Studio Code

Para realizar la detección de la transacción a realizar, se realiza una búsqueda a las colecciones para ver si el libro está disponible (en caso de que no, entonces se asume que es una devolución) y si el alumno puede pedir en préstamo algún libro (se deben revisar dos cosas importantes, la primera es si el alumno puede tomar el libro al tener un tope de cantidad o si el alumno que se registra es el mismo que pidió el libro). Dado que React Native es multiplataforma, existen varias maneras de enviar alertas al teléfono que verifica la aplicación, sin embargo, se coloca el código en caso de que se quiera utilizar exclusivamente para Android.

Figura 60

Código de componentes de pantalla transacciones

```
return (  
  <KeyboardAvoidingView behavior="padding" style={styles.container}>  
    <ImageBackground source={bgImage} style={styles.bgImage}>  
      <View style={styles.upperContainer}>  
        <Image source={appIcon} style={styles.appIcon} />  
        <Image source={appName} style={styles.appName} />  
      </View>  
      <View style={styles.lowerContainer}>  
        <View style={styles.textinputContainer}>  
          <TextInput  
            style={styles.textinput}  
            placeholder="Id del libro"  
            placeholderTextColor="#FFFFFF"  
            value={bookId}  
            onChangeText={text => this.setState({ bookId: text })}  
          />  
          <TouchableOpacity  
            style={styles.scanbutton}  
            onPress={() => this.getCameraPermissions("bookId")}  
          >  
            <Text style={styles.scanbuttonText}>Escanear</Text>  
          </TouchableOpacity>  
        </View>  
      </View>  
    </ImageBackground>  
  </KeyboardAvoidingView>  
)
```

Nota Pantalla obtenida de Visual Studio Code

Algunos de los componentes que se utilizan dentro de la pantalla de Transacciones son cajas de texto en donde escribiremos o escanearemos los identificadores de los involucrados (alumnos y libros), botones que nos permiten tener el acceso a la cámara para escanear los códigos QR o de barra, así como otro que nos permite enviar la información que detonará alguna de las funciones anteriores que determina si se realiza alguna devolución o préstamo de libros.

CONCLUSIONES

El método de investigación científica no es más que la expresión del modo necesario del funcionamiento de la mente humana - Thomas Henry Huxley

Conclusiones

Como podrá observarse, el trabajo está muy relacionado al ámbito educacional debido a que es el área en la que me he desarrollado de manera profesional, esto me ofreció otro panorama para poder desenvolver el mismo y con ello relacionar mi propia experiencia escolar con la que les ofrezco a mis alumnos en este mundo de la programación y el poder plasmar los proyectos de mis estudiantes y poder hacer uso de su creatividad e inteligencia para describir mi trabajo final me hace sentir agradecida con la labor que estoy realizando.

La enseñanza es una labor muy noble que ayuda a formar las diferentes mentes del futuro, por lo cual, hay que procurar nunca apagar esa chispa que caracteriza a muchos de ellos, así como fomentar su creatividad y su capacidad de resolver problemas por sí mismos, por ello, se debe tener mucho cuidado con las palabras que se eligen tanto para elogiarlos como para corregirlos, pues al estar en una edad donde aprenden fácilmente, esto puede ser muy positivo o bien, muy negativo para su desarrollo personal y/o académico.

Durante el desarrollo de mis actividades en el centro de trabajo he aprendido a enseñar y siempre procuro tener una sonrisa en el rostro y un oído para escucharlos, pues, a pesar de que trabajamos eficientemente el tiempo de la clase, muchos de los niños, se sienten cómodos para contar sobre algo que les sucedió en el día o incluso problemas en su vida, siempre trato de poner mi granito de arena para hacer que estos pequeños que son el futuro sean una mejor versión de lo que fueron un día anterior, motivándolos a ser creativos y practicar por cuenta propia todo lo que puede suceder; sobre todo les recuerdo que un error, no es algo malo, es una nueva oportunidad para aprender algo nuevo.

COMPETENCIAS DESARROLLADAS

¡Felices de nosotros si podemos unir los límites de las distintas clases de filosofía al reconciliar la investigación profunda con claridad, la verdad con la novedad! - David Hume

Competencias Desarrolladas

Tabla 1

Competencias desarrolladas

Materia	Competencia aplicada
Base de Datos Distribuidas	Identificar la arquitectura de los sistemas de bases de datos distribuidos, niveles de abstracción y lenguajes para definir arquitecturas de un sistema de gestión de bases de datos.
	Aplicar el diseño de bases de datos distribuidas incluyendo niveles de transparencia y fragmentación de datos e implementar un diccionario de datos para bases de datos distribuidas.
	Realizar el seguimiento, supervisión, control y cierre del proyecto.
Desarrollo de aplicaciones para dispositivos móviles	Utilizar técnicas de modelado para la solución de problemas.
	Aplicar tecnologías de conectividad de bases de datos actuales y emergentes.
Ingeniería de Software	Planificar el desarrollo de un proyecto de software.
Administración y Seguridad de Redes	Implementar una intranet con todas las políticas de seguridad con base en normas establecidas.
Programación WEB	Utilizar un lenguaje de programación del lado del servidor, para la construcción de aplicaciones web

Interacción Humano Computadora	Crea interfaces humano computadora que faciliten el acceso y la interacción con las aplicaciones de cómputo.
Taller de investigación I	Elabora un protocolo de investigación en el que presenta soluciones científico - tecnológicas a problemáticas relacionadas con su campo profesional en diversos contextos.
Taller de investigación II	Consolida el protocolo para ejecutar la investigación y obtener productos para su exposición, defensa y gestión de su transcendencia.

Nota Elaboración propia

Fuentes de información

BYJU'S Learning. (2021). BYJU'S Learning. <https://byjus.com/mx/futureschool/>

Fernández, R. (2019). Programación de aplicaciones android con app inventor 2. Raúl Fernández.

Hanke, J., y Wichern, D. (2006). *Pronósticos en los negocios*. Pearson Educación.

Ghezzi, C., y Jazayeri, M. (1986). Conceptos de lenguaje de programación. DIAZ DE SANTOS.

GitHub. (2023). GitHub. <https://github.com/about>

Google Cloud. (s.f). DialogFlow. Google Cloud.
https://cloud.google.com/dialogflow?utm_source=google&utm_medium=cpc&utm_campaign=latam-MX-all-es-dr-SKWS-all-all-trial-e-dr-1605194-LUAC0014891&utm_content=text-ad-none-any-DEV_c-CRE_548115622232-ADGP_Hybrid%20%7C%20SKWS%20-%20EXA%20%7C%20Txt%20~%20AI%20&%20ML_Dialogflow-KWID_43700066578948454-kwd-473320480267&utm_term=KW_dialogflow-ST_Dialogflow&gclid=Cj0KCQjwmZejBhC_ARIsAGhCqnemo_WVHPSDxqegnI4noxSKZKPk-1whVft3XzTWuQrFiLer8YPbpiYaAIRnEALw_wcB&gclid=aw.ds&hl=es-419

Gutierrez, E. (2009). JavaScript: Conceptos básicos y avanzados (bibliotecas Prototype y Script.aculo.us). Ediciones ENI.

INVANEP. (30 diciembre 2020). Desarrollo cognitivo del ser humano. INVANEP.
https://invanep.com/blog_invanep/desarrollo-cognitivo-del-ser-humano

Johnson, B. (2019) Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers. Wiley.

Maudson, M. (2022). Aprendizaje y desarrollo en la infancia y preescolar. Amazon Digital Services.

Microsoft. (2013). Qué es Microsoft MakeCode. Microsoft. <https://www.microsoft.com/es-es/makecode/about#:~:text=Microsoft%20MakeCode%20es%20una%20plataforma,progr%20esar%20hacia%20la%20programaci%C3%B3n%20real>.

Minecraft Education. (2023). Minecraft Education. <https://education.minecraft.net/es-es>

Mit App Inventor. (2012-2022). About us. Mit App Inventor. <https://appinventor.mit.edu/about-us>

Paul, A., y Nalwaya, A. (2019). React Native for mobile development: Harness the Power of React Native to Create Stunning iOS and Android Applications. Apress.

Puciarelli, L. (2020). Node JS - Vol. 1: Instalación - Arquitectura - node y npm. RedUsers.

Ramírez, I. (23 febrero 2020). BlueStacks: guía a fondo del completo emulador de Android para PC. Xataka Android. <https://www.xatakandroid.com/aplicaciones-android/bluestacks-guia-a-fondo-completo-emulador-android-para-pc#:~:text=BlueStacks%20es%20uno%20de%20los,el%20teclado%20y%20el%20rat%C3%B3n>

Real Academia Española. (2023). Aprendizaje. En Diccionario de la lengua española. <https://dle.rae.es/aprendizaje>

SEIDOR. (20 febrero 2023). ¿Qué es Firebase? ¿Qué ventajas ofrece en 2023 a nuestras apps? SEIDOR. <https://www.seidor.com/blog/firebase-que-es>

Tynker from BYJU'S (2023). Coding for kids, kids online coding classes & games. Tynker from BYJU'S. <https://www.tynker.com/#/>