ISSN 1688-2784

UNIVERSIDAD DE LA REPÚBLICA
FACULTAD DE INGENIERÍA

Extensión de Funcionalidades y Validación de MateFun Infantil - Plataforma Web

PROYECTO DE GRADO DE INGENIERÍA EN COMPUTACIØN
INSTITUTO DE COMPUTACIÓN, FACULTAD DE INGENIRÍA
UNIVERSIDAD DE LA REPÚBLICA

Federico Luongo
Stéfano Pesamosca
Emiliano San Román

TUTORES

Gonzalo Tejera . . . . . Universidad de la República
Marcos Viera . . . . . Universidad de la República

Montevideo
jueves 7 abril, 2022

Agradecimientos

A familiares y amigos, presentes a lo largo de toda la carrera con su apoyo incondicional, inclusive para este proyecto.
A los tutores Gonzalo y Marcos, por el tiempo dedicado, transmitiéndonos su conocimiento y experiencia en todo momento. Fue un placer haber trabajando en conjunto.
También agradecer a las maestras Bettina, Leticia y Marcela que nos brindaron su tiempo y conocimiento.
A la experta en interfaz de usuario Ewelina, que gracias a ella pudimos resolver inconvenientes de índole gráfico.
Por último dar las gracias a los niños Felipe y Julieta que nos dedicaron su tiempo para la validación, siempre predispuestos a ayudar.

Resumen

MateFun Infantil es un lenguaje de programación visual desarrollado en la Facultad de Ingeniería de la Universidad de la República como Proyecto de Grado, con objetivos didácticos para niños de segundo ciclo escolar. Tiene una interfaz amigable ya que está implementada mediante programación orientada a bloques, con la salvedad que reemplaza los clásicos bloques de encastre por elementos que se llaman tuberías. Al ejecutar un conjunto de estas, retornan resultados según los datos que reciben de entrada, permitiendo a los niños introducirse en el concepto de función matemática en edades tempranas.

Este Proyecto de Grado tiene dos objetivos clave. El primero, la implementación de nuevas funcionalidades para mejorar la interacción con el usuario y la usabilidad de la aplicación. El segundo, realizar validaciones con los usuarios finales.

En las validaciones se consultó tanto con maestros como con niños. En una primera instancia se utilizaron prototipos para mostrar las nuevas funcionalidades que se pretendían implementar. Además se presentó lo ya implementado en el Proyecto del año anterior, con la intención de analizar posibles cambios en el lenguaje utilizado.

Por último, luego de implementados los cambios, teniendo presente los comentarios recibidos en primer lugar, se volvió a validar con las mismas personas. De este modo se buscaba confirmar si el resultado cumplía con lo esperado según los prototipos mostrados y las sugerencias recibidas, e identificar posibles correcciones a realizar en una última instancia.

Tabla de contenidos

Capítulo 1

Introducción

1.1. Motivación y contexto

La aplicación web MateFun, en su versión infantil, es un proyecto desarrollado por la Facultad de Ingeniería de la UdelaR, que tiene como objetivo brindar a los niños un primer acercamiento a la programación y al concepto de función.
La misma está comprendida dentro de los lenguajes de programación visual con bloques. Esto implica que los programas se implementan sin la necesidad de escribir código, lo que facilita su utilización en los niños. Los programas se manipulan arrastrando bloques prediseñados que encastran entre si según su definición, desde una barra de herramientas, hacia un área donde posteriormente serán ejecutados. La diferencia de esta aplicación con la mayoría de estos lenguajes que se encuentran en la actualidad, es que este maneja el concepto de tubería, y no estrictamente de bloque.
A su vez fue pensada como un lenguaje de programación funcional, lo que significa que un programa se puede interpretar como una expresión que arroja un único resultado. La expresión se estructura utilizando funciones, las que ante una misma entrada retornan siempre un mismo resultado.
Asimismo la plataforma web de MateFun Infantil, es una extensión de la versión original de MateFun, la cual fue diseñada para introducir a los liceales a la programación y fortalecer la apropiación del concepto de función matemática. Son varios los proyectos que han extendido la aplicación original de MateFun [30], tanto en su versión web como en su aplicación para Android. Esta última, la cual fue desarrollada en su versión infantil, se implementó mediante la utilización de Blockly[1], ya que permite crear lenguajes visuales utilizando bloques.
La motivación de este proyecto apunta a mejorar la usabilidad de la plataforma web de MateFun Infantil. Por tanto se pretende diseñar e implementar un conjunto de características disponibilizándolas para su uso, que permitan un uso más sencillo y amigable para los niños. Para ello durante el proceso se involucrará a docentes y niños. Siguiendo la misma linea, se procura realizar validaciones sobre los términos que se utilizan en la aplicación de modo que estos sean adecuados para los niños.

1.2. Estado inicial de MateFun Infantil - Plataforma web

El proyecto desarrollado, tiene como base el proyecto de grado del año 2020, tutorizado por los mismos docentes, en donde se creó una primera versión de la plataforma web de MateFun Infantil (a la cual nombraremos como MateFun Infantil o simplemente MateFun). La misma fue ideada

[1] Cliente de bibliotecas para el lenguaje de programación Javascript creado por Google

para niños que se encuentran en los últimos tres años de educación primaria, y posee las siguientes funcionalidades y características implementadas:

- Capacidad de manejar números decimales, números enteros, colores y figuras.
- Capacidad de manipular mediante diferentes operaciones, figuras de dos dimensiones.
- Posibilidad de definir funciones, siempre tomando el conjunto de tuberías de todo el tablero para la definición de la misma.
- Operaciones aritméticas con diferentes niveles de dificultad
- Manipulación de listas de elementos.

La aplicación tiene la posibilidad de seleccionar niveles de dificultad, en donde la mayoría de las características antes mencionadas se encuentran en el nivel básico. Las que poseen una dificultad algo mayor, como por ejemplo la función seno, se encuentran en el nivel intermedio, llamado “avanzado”. Por último se encuentra el nivel “completo” que tiene habilitadas todas las características anteriores y añade la manipulación de listas.
Con respecto a su interfaz gráfica, en la Figura 1.1 se puede observar la pantalla principal de la aplicación, la cual aparece luego de haber iniciado sesión. En el centro de esta se encuentra el tablero o grilla, donde se colocan las tuberías que serán ejecutadas. En la parte superior derecha, se ubica el panel de resultado, el cual al inicio se encuentra colapsado. En la parte superior izquierda se encuentran los diferentes menús que ofrecen distintos tipos de tuberías, por ejemplo constantes como en la misma figura antes mencionada. Por último se encuentran los botones de limpiar, guardar y probar, en la esquina superior derecha del panel central y la papelera en la parte inferior izquierda a la cual se pueden arrastrar tuberías que desean ser borradas.

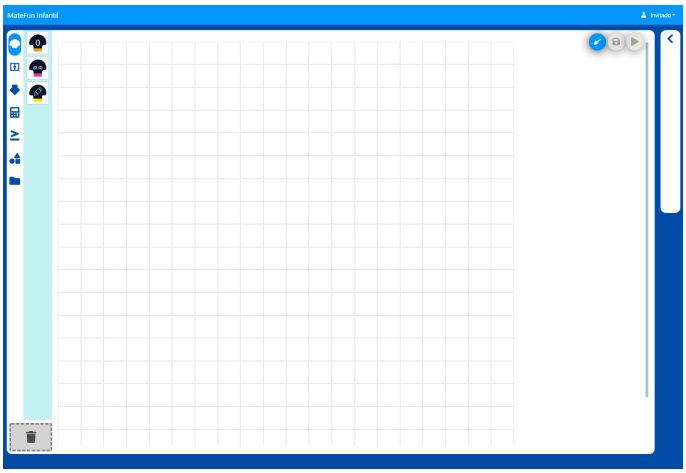


Figura 1.1: Pantalla inicial MateFun

Al formar un conjunto de tuberías en el tablero, y ejecutarlas mediante el botón probar (símbolo de reproducir), se obtiene el resultado, que en caso de ser numérico como en la Figura 1.2 se puede apreciar tanto en la tubería de tipo resultado, como en el panel que se destina para esto.

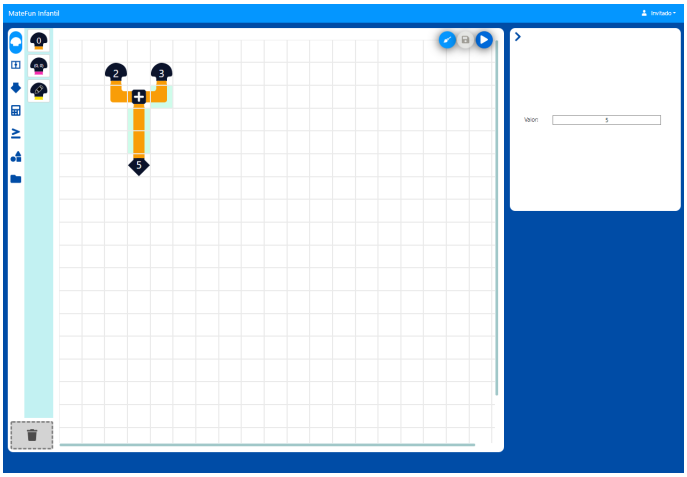


Figura 1.2: Suma en MateFun

Otro ejemplo de uso clásico de la aplicación es la creación de figuras y el coloreado, el cual se puede apreciar en la Figura 1.3. En este caso, la única forma de observar el fruto de la ejecución del programa, es a través del panel de resultado. En caso que el lector desee conocer en mayor profundidad el funcionamiento de MateFun Infantil en el estado previo al inicio de este proyecto, puede consultar el manual de usuario [31].

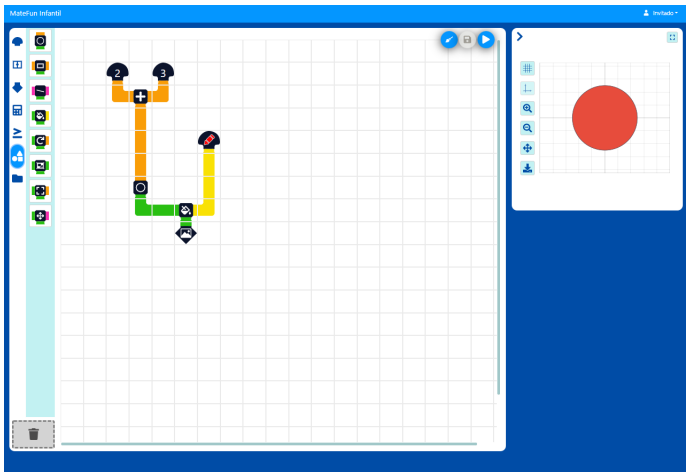


Figura 1.3: Figura en MateFun

Si bien lo implementado en la aplicación está muy bien logrado, la misma sufre algunas carencias. En primer lugar, posee un área de trabajo estática. Esto implica que al agregar muchas tuberías, se llena el tablero donde se ubican estas y no permite añadir más. Otra característica con un funcionamiento deficiente, es la manipulación de tuberías en el tablero. Para borrar o mover se deben seleccionar una a una, lo cual es bastante tedioso si el conjunto de tuberías creado es grande. Otro defecto que se encuentra es la interacción con el usuario. Dado que los usuarios finales son niños, se debe mejorar la presentación de los mensajes que muestra MateFun, como por ejemplo los errores.

Acerca de esto, también se deben corregir los textos ajustándolos al contexto. Por último también se halla la falta de algún tipo de modalidad que permita resolver ejercicios ya creados, evitando que la única forma de utilizar la herramienta por parte del usuario sea construyendo desde cero el conjunto de tuberías.

1.3. Objetivos

Este proyecto tiene dos objetivos principales:

1. Realizar el diseño e implementación de una serie de funcionalidades para añadir a la aplicación web de MateFun Infantil, de forma que mejore su usabilidad.
2. Realizar validaciones con expertos en el dominio y niños, sobre ciertas características ya implementadas y sobre las nuevas funcionalidades.

Acerca del primero de estos puntos, se parte del siguiente conjunto potencial de funcionalidades a implementar que fueron provistas por los supervisores, las cuales a lo largo del proyecto se irán refinando mediante las validaciones y la retroalimentación recibida:

- Implementar la grilla de forma que se vea como potencialmente infinita creciendo en todas direcciones. Identificada en el resto del proyecto como “grilla infinita”
- Permitir la selección de múltiples tuberías a la vez, para poder manipularlas en conjunto. Identificado en el resto del proyecto como “selección múltiple”
- Generalizar el concepto de par, como forma de unir dos tuberías en una de dos colores. De esta forma habría tuberías que llevan dentro más de una tubería. Identificado en el resto del proyecto como “Generalización del concepto de par” o “multitubería”.
- Extender el tipo de datos numérico, de forma que permita manipular enteros y racionales en vez de números. Identificado en el resto del proyecto como “extensión del tipo de datos numérico”.
- Permitir la definición de funciones recursivas resolviéndolo de forma visual. Identificado en el resto del proyecto como “funciones recursivas”.
- Permitir importar funciones en código MateFun sin la metainformación de sus tuberías. Identificado en el resto del proyecto como “importar código MateFun”.

Sobre el segundo objetivo, se pretende revisar la terminología usada en el lenguaje para adaptarla al contexto, validándolo con docentes de primaria. Además de esto, se realizará un diseño participativo [16] de las nuevas funcionalidades, teniendo una validación previa y posterior a la implementación, y como consecuencia se ratificará la usabilidad de toda la aplicación.

1.4. Organización del documento

El documento está compuesto por seis capítulos con sus correspondientes secciones y subsecciones, los cuales están ordenados siguiendo una lógica que le permita al lector un entendimiento de como se implementó y desarrolló el proceso de este proyecto.
En el Capítulo 2 se realiza un análisis del estado del arte, haciendo foco en las características nuevas

que se implementarán en la aplicación.
Continúa el Capítulo 3 en donde se detalla la validación inicial que se hizo con docentes y niños mediante prototipos de algunas de las funcionalidades a implementar.
Seguido se encuentra el Capítulo 4 en el cual se explica como se implementaron las nuevas funcionalidades.
Posteriormente el Capítulo 5 detalla las validaciones realizadas sobre estas funcionalidades implementadas y sobre otras características de MateFun. Finalmente en el Capítulo 6 se explica a qué conclusiones se llegó en el proyecto, junto con algunas posibles implementaciones y tareas a desarrollar en el futuro.
Por último se encuentran las secciones que contienen casos de uso, pruebas y un anexo técnico para un trabajo futuro, además de los índices de tablas, índices de figuras, las referencias y el glosario con los principales conceptos.

Capítulo 2

Estado del arte

En este Capítulo se presenta un resumen del estado del arte del proyecto, el cual fue confeccionado al inicio del proyecto haciendo énfasis en las características potenciales a implementar durante el mismo.
En principio se presenta la evaluación de alguno de los lenguajes de bloques tenidos en cuenta en el análisis del proyecto donde se creó MateFun Infantil (en el documento del Estado del Arte [17] se encuentra el análisis de varios lenguajes más) y se agregan aquellos que tienen relevancia en cuanto a la implementación de funcionalidades similares.
Posteriormente se desarrolla un análisis de las bibliotecas actuales con las cuales se puede llegar a implementar algunas de las características.

2.1. Características de estudio

A continuación se realiza una breve descripción de las características que se pretenden incorporar o modificar en la versión actual de la aplicación. Sobre las siguientes características se realiza el estudio del estado del arte:

Funciones recursivas: Si bien en el estado en que se encuentra MateFun, es posible definir funciones recursivas, no esta resuelto de forma intuitiva y por lo tanto es necesario establecer una implementación visual que brinde la posibilidad de definir este tipo de funciones de forma más simple.

Concepto de par: Se pretende realizar una generalización del concepto de par como forma de unir dos o más tuberías en una de dos o más colores. De esta forma sería posible contar con tuberías que llevan juntas en su salida a más de una tubería en paralelo. Para esto es necesario analizar formas de resolver visualmente esta funcionalidad. Una tubería de salida identificaría mediante colores a las tuberías que contiene y por lo tanto es necesario implementar visualmente la selección de una de estas tuberías.

Selección múltiple: Se considera necesario incorporar la selección de un conjunto de bloques por área para poder manipularlos. En el estado que se encuentra MateFun, se deben mover bloques uno a uno lo cual no resulta práctico a medida que se incrementa la cantidad de tuberías en el tablero.

Grilla infinita: Se aspira a que la grilla o tablero de trabajo no sea de tamaño fijo, es decir que la grilla se vea como potencialmente infinita y pueda crecer en todos los sentidos. En el estado actual del proyecto, el área de trabajo es acotada y esto limita el tamaño de la soluciones que pueden implementarse.

Terminología utilizada: Se realiza una revisión de la terminología usada en el lenguaje, con el objetivo de evaluar posibles cambios para mejorar la usabilidad. Para trabajar sobre este punto se toma el conjunto de palabras claves utilizadas en el sistema, comparándolo con otros lenguajes con el objetivo de encontrar posibles cambios. Las palabras claves están ordenadas por categoría con una breve descripción para dar contexto durante la revisión

Detalle de palabras claves:

- **Tipos de Tubería:** En esta categoría están comprendidos todos los tipos de tuberías que se pueden utilizar en MateFun Infantil. Comprende todas las constantes, variables, funciones y figuras disponibles en la modalidad básica de la aplicación.
  - Constantes
    - Número, par de números, color
  - Variables
    - Genérico, número, par de números, color, figura
  - Resultado
  - Matemática
    - Suma, resta, multiplicación, división
  - Condiciones y comparadores
    - Si, y, igual, distinto, mayor, mayor o igual, menor, menor o igual
  - Figuras
    - Círculo, rectángulo, linea, pintar, rotar, agrupar, escalar, mover.
- **Botones de acción:** Comprenden los botones de ejecución disponibles al usuario mientras este utiliza con la aplicación.
  - Limpiar
  - Guardar
  - Probar
- **Otros botones:** Contiene el botón que permite consultar las funciones guardadas previamente para su uso posterior.
  - Mis funciones guardadas

Importar código MateFun: Se estudia la posibilidad de importar funciones en código MateFun "puro", es decir sin la meta información de sus tuberías. Para este punto se plantea la opción de ver este código como "caja negra" o de analizarlo para llegar a su representación visual.

Extensión el tipo de datos numérico: Surge la necesidad de extender el tipo de datos numérico y validar si es posible contar con enteros, decimales y fracciones en lugar de enteros y decimales como es soportado actualmente.

2.2. Lenguajes afines

En la presente Sección se realiza una revisión de lenguajes similares a MateFun Infantil con el objetivo de comparar y analizar como se resuelven en dichos lenguajes las funcionalidades que el proyecto apunta a incorporar al sistema. Para seleccionar los lenguajes revisados se incluyeron los lenguajes que fueron tendidos en cuenta en la etapa anterior del proyecto, quitando de la selección las aplicaciones TortugArte y TortuBots. Esto debido a que en acuerdo con el equipo y docentes no tenían relevancia relacionada a la nuevas funcionalidades. Además de lo mencionado anteriormente fue propuesto incorporar al análisis las aplicaciones EtcH, CODE.org y Lightbot.

2.2.1. Scratch

Scratch es el primer lenguaje que fue analizado en la etapa anterior del proyecto, es un lenguaje de alto nivel basado en bloques recomendado para niños de entre 8 a 16 años. Es de licencia libre y está implementado sobre SqueaK [27], lenguaje de programación de código abierto que promueve el desarrollo de aplicaciones gráficas. En relación al concepto de recursión en Scratch es posible crear un bloque personalizado que puede ser reutilizado, se pueden agregar tipos de datos de entrada que permiten implementar la noción recursiva [6]. En la Figura 2.1 se muestra la definición de una función “Countdown from” y la misma es invocada a si misma dentro de su bloque de definición.

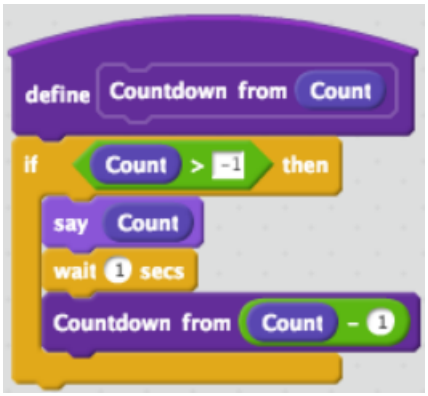


Figura 2.1: Ejemplo de recursión en Scratch

En cuanto al manejo del entorno de trabajo, el tamaño de la grilla es incrementada automáticamente a medida que se agregan componentes y se llena el espacio en el eje horizontal o vertical. Como se observa en la Figura 2.2, se hace uso de los controles de zoom in, zoom out y restablecer.

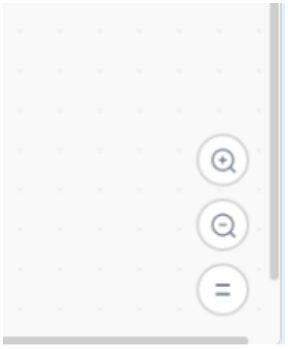


Figura 2.2: Uso de la grilla en Scratch

Sratch no posee ninguna herramienta para manejar pares de elementos, como tampoco permite la manipulación de números decimales o fracciones. Otra característica la cual carece, es la importación de código de programación puro.
La selección múltiple de objetos es algo que no se encuentra disponible tanto en la versión de prueba online como en la documentación. Dado que las funciones en este lenguaje se definen con cierta jerarquía, basta con arrastrar las funciones presionando a nivel de la definición de la función para moverla entera en la grilla. De lo contrario, si se arrastra desde partes inferiores, la función se desacopla según la sección del bloque clickeado. En MateFun no es posible transmitir esa noción de jerarquía constructiva por lo tanto es necesario tener una herramienta de selección explícita, dado que la única opción es mover elemento por elemento hacia "abajo" pero no es una solución que sea eficiente para el usuario.

En la Tabla 2.1 se muestra la comparación entre la terminología utilizada en MateFun Infantil y Scratch.

MateFun	**Scratch**
Constantes	Por tipo de constante: movimiento, apariencia, sonido
Variables	Variables
Resultado	No aplica
Matemática	Operadores
Condiciones y comparadores	Control
Figuras	No aplica
Limpiar	No disponible
Guardar	Archivo/Guardar
Probar	Ir (bandera verde)

Tabla 2.1: Mapeo de terminología Scratch

2.2.2. PilasBloques

Este lenguaje es otro de los ya conocidos en el contexto del proyecto, el mismo está orientado a resolver problemas o desafíos previamente definidos por lo que no es posible enfocar su uso en

proyectos de final abierto. Fue desarrollado para abarcar dos grupos de público objetivo, en primer lugar niños entre 5 y 8 años incluidos dentro del primer ciclo escolar en Argentina. En segundo lugar niños entre 9 y 12 años incluidos dentro del segundo ciclo. Cabe destacar que es un proyecto educativo de licencia libre y fue implementado sobre la biblioteca Blockly. En cuanto a sus características si bien son más restringidas, son muy similares a las ya vistas en Scratch. Analizando las características tales como la recursión, el manejo de la grilla y el concepto de jerarquía al momento de definir los bloques, son resueltas de la misma forma que en el lenguaje anterior. Del mismo modo, no permite importar código puro, ni posee herramientas para manipular pares de elementos. Finalmente el lenguaje no maneja operaciones, únicamente tiene disponible el tipo numérico para implementación de repeticiones. [18]

En la tabla 2.2 se muestra la comparación entre la terminología utilizada en MateFun Infantil y PilasBloques.

MateFun	**PilasBloques**
Constantes	Valores
Variables	No disponible
Resultado	No aplica
Matemática	Operadores
Condiciones y comparadores	No aplica
Figuras	No aplica
Limpiar	Ícono de papelera
Guardar	Guardar solución
Probar	Ejecutar

Tabla 2.2: Mapeo de terminología PilasBloques

2.2.3. Viskell

Viskell [29] es un ambiente experimental de programación visual open source, implementado en Java 8/JavaFX [10], para un lenguaje de programación funcional similar a Haskell [14]. También conocido en el contexto del proyecto, siendo el único analizado anteriormente que no utiliza bloques encastrados. En su lugar, los bloques tienen entradas y salidas definidas y se utilizan "cables" para unir salidas con entradas. En los bloques es posible iniciar variables o definir funciones. Al igual que en MateFun Infantil es posible crear bloques a partir de otros. En el caso de Viskell los bloques son agrupados en regiones y estas pueden ser nombradas y colapsadas. Los bloques se pueden mover de forma individual o en conjunto moviendo las regiones creadas y el borrado funciona de igual manera. No permite el manejo de pares de elementos, como tampoco manipular fracciones, ni importar código de programación en estado puro.
Viskell cuenta con una interfaz gráfica multitáctil permitiendo distintas acciones según los dedos que se utilizan y el movimiento que realizan. Además cada bloque muestra los tipos y ante cualquier error de tipo se visualiza localmente. El flujo de ejecución va desde arriba hacia abajo donde la grilla es infinita teniendo botones para aumentar y disminuir el zoom utilizado.
La recursión en Viskell es soportada conectando la salida con una entrada superior tal y como se muestra en la Figura 2.3. En este caso, se utiliza la función "register" que almacenan enteros y retornan el valor demorado un ciclo de reloj. Ambos "register" luego del primer ciclo del reloj, retornan el valor entero 1. El valor que retorna el "register" inferior es el que se concatena en la salida

generada, en el siguiente paso el “register” superior recibe el valor de la suma del paso anterior, mientras que el “register” inferior recibe el valor almacenado previamente en el “register” superior. Generando de esta forma la secesión de Fibonacci de forma recursiva (Fn = Fn1 + Fn2).

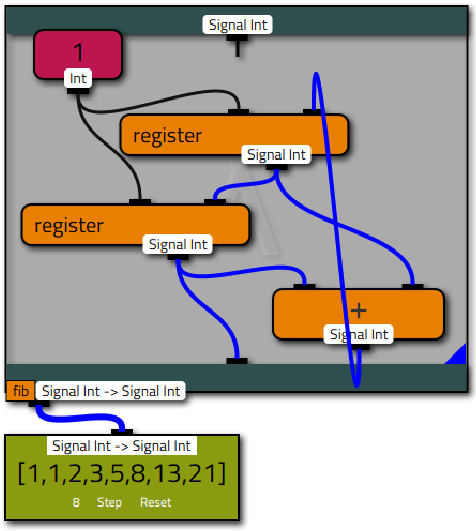


Figura 2.3: Ejemplo de recursión

Para evitar el ciclo de edición - compilación - depuración Viskell proporciona una retroalimentación inmediata ante cada cambio realizado.

En la Tabla 2.3 se evalúa la terminología utilizada en Viskell comparándola con MateFun:

MateFun	**Viskell**
Constantes	Un bloque puede ser una constate con su tipo
Variables	Un bloque puede ser una variable con su tipo
Resultado	No aplica
Matemática	Operadores
Condiciones y comparadores	Control
Figuras	No aplica
Limpiar	No disponible
Guardar	Archivo/Guardar
Probar	No aplica

Tabla 2.3: Mapeo terminología Viskell

2.2.4. CODE.org

CODE.org [2] es una organización sin fines de lucro que desarrolló un proyecto en Estados Unidos que incluye una aplicación web de código abierto implementado sobre Blockly. Uno de sus principales objetivos es ampliar el acceso a las ciencias de la computación en las escuelas.
Posee un proyecto bien formado con variedad de cursos para adolescentes y niños a partir de 4 años.

Cada uno de los cursos está compuesto por lecciones que parten de los conceptos más básicos para llegar a construcciones más complejas.
En lo que refiere a la aplicación web, la misma está basada en la programación por bloques. En esta, existe un panel donde se construye el conjunto de bloques, que al ejecutarlos, desencadenan una serie de acciones del personaje principal de la lección. La aplicación se utiliza tanto dentro de los cursos que se brindan, como de forma libre si se quieren crear proyectos desde cero.
En relación a sus funcionalidades, acerca de la implementación de funciones recursivas, en CODE es posible realizarlas. Una de las posibilidades que ofrece es la creación de funciones (además permite crear acciones, bucles, entre otros), en donde a la hora de crear una, la misma se va auto-guardando, de forma que permite en su construcción, invocarse a sí misma. En la Figura 2.4 se muestra un ejemplo de su implementación

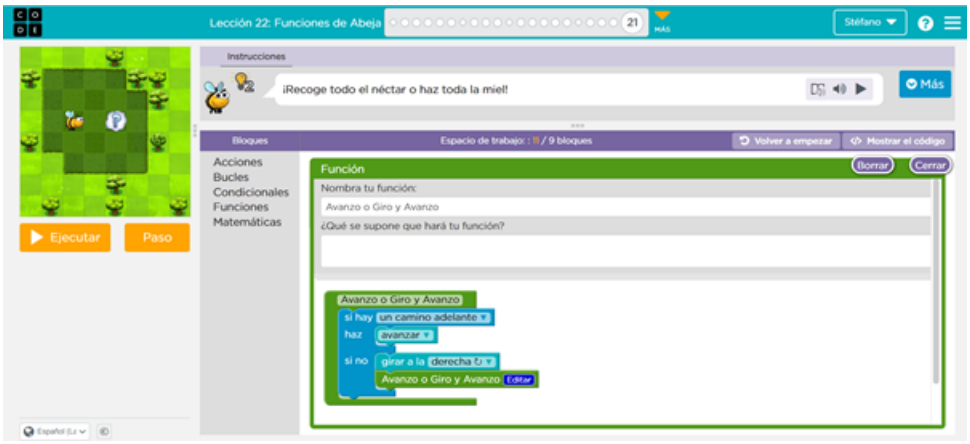


Figura 2.4: Ejemplo de recursión en CODE.org

En cuanto a la funcionalidad de manipular pares de elementos, no tiene esta posibilidad.
Por otra parte, con respecto a la implementación del panel en donde se colocan los bloques, en primera instancia es de dimensiones fijas. A medida que se van ubicando los bloques permite extenderlo únicamente hacia abajo de forma infinita. Para las lecciones que plantea la aplicación y la mayoría de los usos que le podrían dar los usuarios, esta característica es suficiente, ya que por el modo en que se van ubicando los bloques no es necesaria la extensión en otras direcciones.

Con respecto a la terminología que utiliza CODE, en la Tabla 2.4 se plantea la comparación con la terminología actual de MateFun.

MateFun	CODE
Constantes	No aplica. Las une a otros bloques constructivos
Variables	Variables
Resultado	No aplica
Matemática	Subconjunto dentro de variables
Condiciones y comparadores	Lógica
Figuras	Utiliza el término “Sprite”. No son figuras geométricas
Limpiar	No disponible
Guardar	No disponible
Probar	Ejecutar

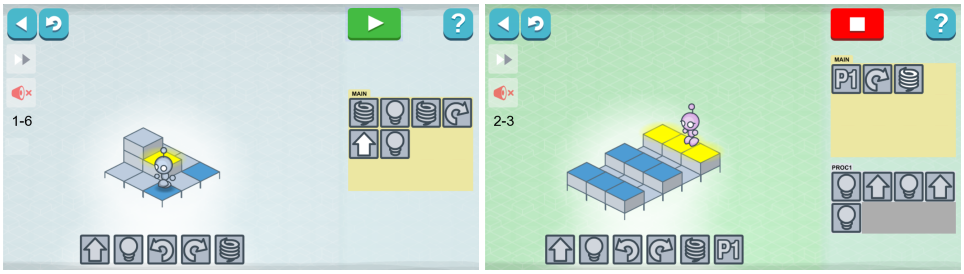
Tabla 2.4: Mapeo terminología CODE.org

En la tabla comparativa se observa que hay algunas coincidencias en la terminología de los conceptos en ambos lenguajes. En gran parte las diferencias se hallan debido a que CODE no es un lenguaje que esté focalizado en lo aritmético y geométrico, sino más que nada en movimientos y acciones.

Otra funcionalidad que se estudia en esta aplicación, es la selección múltiple. Como fue mencionado, CODE es un lenguaje de programación con bloques. A partir de esto, está implementado que al arrastrar el bloque exterior, los que están anidados en su interior, se mueven junto a él. Sin embargo, no es posible seleccionar y mover de forma múltiple, bloques que están a un mismo nivel. Con respecto a la importación de código puro, CODE no permite hacerlo, sin embargo a modo ilustrativo, muestra un ejemplo del código JavaScript que se ejecuta por detrás de algunas de las acciones que realiza el usuario.
En relación al tipo de datos numérico que soporta la aplicación, la misma permite la manipulación únicamente de números enteros y decimales.
Analizando globalmente la aplicación, la misma es bastante completa. Su uso es intuitivo, y la posibilidad de introducirse a la misma mediante cursos, con lecciones basadas en juegos, y adaptadas para varias edades, hacen el punto fuerte de esta aplicación. Otro detalle que es interesante es que si bien no permite la manipulación de código puro, da algunos ejemplos de la ejecución de código JavaScript que hay por detrás de las diferentes funcionalidades que el usuario utiliza, para al menos dar un acercamiento al mismo.

2.2.5. Lightbot

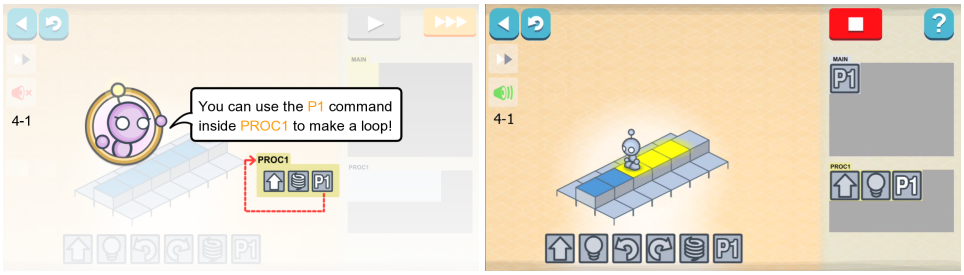
Por último se realiza el análisis de la aplicación Lightbot [13]. Es necesario mencionar que no es un lenguaje de programación en si mismo, sino que es un juego de rompecabezas de programación, lo que significa que su mecánica de juego requiere el uso de lógica de programación para resolver los diferentes niveles. Está disponible de forma gratuita tanto para Windows, como MAC y dispositivos móviles, en varios idiomas.
A su vez aplica la programación con bloques, además de otras características que hacen que sea interesante para su estudio y que mencionaremos a continuación. El juego consiste en guiar a un robot para que ilumine los mosaicos resolviendo los desafíos. Los jugadores realizan una comprensión real de los procedimientos, loops y condicionales.



(a) Definición del main en Lighbot (b) Uso de procedimiento en Lightbot

Figura 2.5: Inicio Lightbot

Como se observa en la figuras 2.5a y 2.5b, se deben arrastrar los bloques de la barra inferior hacia los paneles de la derecha, ya sea al main o definiendo un nuevo proceso (que tiene la posibilidad de ser invocado dentro del main). Por tanto tiene un uso bastante básico e intuitivo.
Una de las principales virtudes que se halló en este juego, es que resalta el uso del concepto de recursión. Tanto en el video demostrativo de su pantalla inicial, como en diferentes secciones de la página web del juego, se encuentran las imágenes 2.6a y 2.6b que lo ejemplifican.



(a) Explicación de recursión en Lighbot (b) Uso de recursión en Lightbot

Figura 2.6: Recursión en Lightbot

La aplicación no permite la utilización de pares de bloques ni de otros elementos
En referencia a los paneles donde se colocan los bloques, estos son fijos y no se extienden por más que se completen todas las posiciones de los bloques. El objetivo de esto, es que se use de forma obligatoria la definición e invocación de procedimientos.
La terminología que utiliza refiere a movimientos y otros símbolos que aluden exclusivamente al juego, por tanto los términos no son comparables a MateFun. Del mismo modo sucede con la importación de código, al ser exclusivamente un juego, no implementa este tipo de funcionalidades.
En lo que respecta a la selección múltiple de bloques, Lightbot no permite realizarla.

2.3. Tecnologías

La aplicación web MateFun Infantil está implementada con React. Por tanto, la finalidad del siguiente estudio tecnológico es encontrar opciones compatibles con lo ya implementado, para realizar

las nuevas características. Se opta por llevar a cabo el análisis sobre tres características en particular, ya que el equipo de trabajo entiende que son las que ameritan una mayor investigación tecnológica, y no solo modificaciones en el código actual.

2.3.1. Grilla infinita

Como se mencionó al inicio, se pretende que la grilla donde se colocan las tuberías se extienda de forma potencialmente infinita. Para ello se estudia alguna posible biblioteca o modificación en el código para soportar esta funcionalidad.

Barra de desplazamiento infinita

Una forma de implementar esta característica es mediante una barra de desplazamiento lateral, que a medida que el usuario gire el scroll se vaya extendiendo infinitamente.
Para realizar esta implementación, se puede utilizar alguna de las siguientes bibliotecas:

- react-infinite-scroller [9] [8]
- react-infinite-scroll-component [7] [3]

Esta es una alternativa posible, sin embargo, se encuentra el defecto que debido a que el usuario puede extender la grilla a su antojo, pueden quedar elementos dispersos en diferentes partes de la misma, dificultando el armado de un conjunto de tuberías que sea ejecutable.

Extensión por inserción de elementos

Otra forma de implementar esta característica es, a medida que se vayan agregando tuberías a una distancia del borde menor a cierto umbral definido, extender la grilla hacia esa dirección en una cantidad de casillas también previamente definida.
Este método es el que utilizan la mayoría de los lenguajes que se estudiaron anteriormente y permiten esta característica.
Para su implementación en el proyecto actual, se debería llevar un control de lo mencionado anteriormente en la implementación del manejo del tablero, de forma que se modifique el tamaño de la grilla cuando corresponda. Esta implementación está desarrollada dentro del componente Board API que está detallado en la sección 4.2.
Esta alternativa se entiende adecuada, ya que el tamaño de la grilla se modifica en caso estrictamente necesario, evitando los problemas mencionados en la primer alternativa.

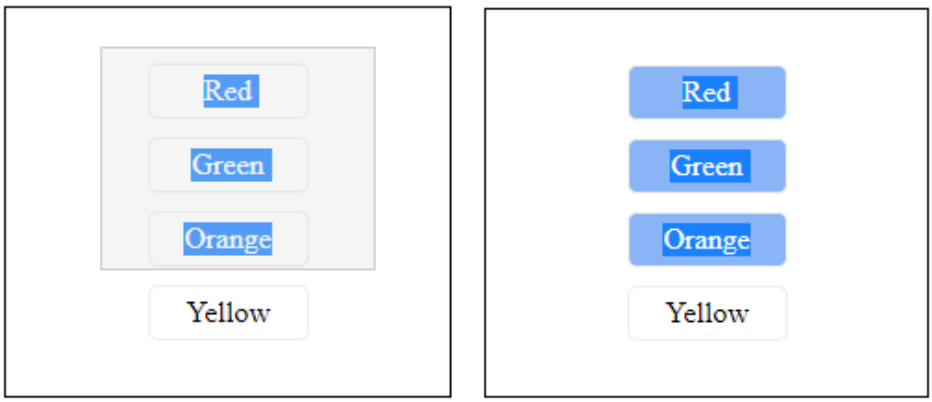
2.3.2. Selección múltiple

Luego de realizado el análisis, solo en Viskell se identifica esta característica, pero por la forma en que está implementado el tablero, en donde los bloques se agrupan en regiones para este tipo de movimientos, no es posible reutilizar el código, ni el modo de implementación. Por lo tanto se procede a investigar posibles implementaciones de esta funcionalidad. A continuación se listan las soluciones existentes relevadas para esta característica:

- react-selectable-simple [25]
- react-selectable [23]

react-selectable-simple

Este componente permite incorporar el rectángulo de selección de items en el tablero, de forma de seleccionar los items clickeando en el tablero y formando el área de selección, como se ejemplifica en las figuras 2.7a y 2.7b. Se revisa la documentación de la biblioteca con el objetivo de encontrar los detalles de instalación, propiedades de uso básico y características de la implementación. Finalmente se evalúa la demo de uso presentada en la referencia [24] para validar el funcionamiento de la biblioteca.



(a) Ejecución de selección de items con react-selectable-simple

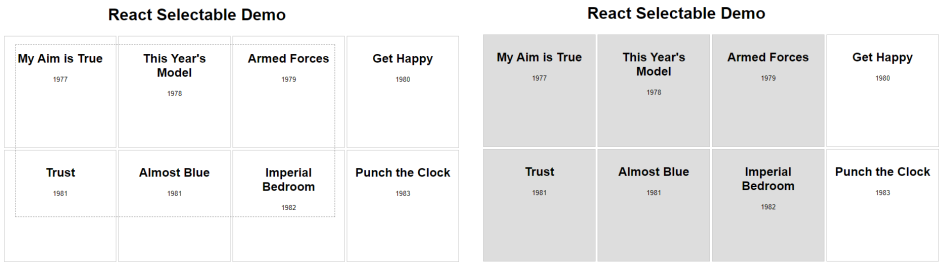
(b) Resultado de selección de items con react-selectable-simple

Figura 2.7: Ejemplo de uso de biblioteca react-selectable-simple

react-selectable

Al igual que la biblioteca anterior, esta implementación permite la selección múltiple o individual de items haciendo uso del rectángulo de selección. Como resultado del análisis de la documentación de la biblioteca se encuentran más detalles de configuración y del uso de la misma, además de tener un mayor nivel de adaptabilidad al tipo de aplicación al que se la incorpora. Este último punto genera la mayor diferencia respecto a la primera biblioteca analizada puesto que su uso está enfocado y acotado a un escenario compuesto por listas mientras que react-selectable [23] permite ser adaptado de forma más sencilla a la realidad del proyecto. Por lo tanto, el análisis de configuración se realiza teniendo en cuenta los componentes a ser modificados en el proyecto MateFun en la etapa de implementación.

A continuación, en las imágenes 2.8a y 2.8b se muestra el comportamiento de la biblioteca en la demo de referencia [4]:



(a) Ejecución de selección de items con react-selectable (b) Resultado de selección de items con react-selectable-simple

Figura 2.8: Ejemplo de uso de biblioteca react-selectable

Si bien ambas bibliotecas aplican a lo que se necesita implementar, utilizar react-selectable [23] sería la mejor opción debido a las diferencias de adaptabilidad, documentación y mayor uso de la biblioteca en otros proyectos.

2.4. Análisis comparativo

En la Tabla 2.5 se presenta un resumen comparativo que muestra si las aplicaciones estudiadas satisfacen o no los requerimientos que se pretenden implementar en este proyecto.

	Scratch	PilasBloques	Viskell	CODE.org	Lightbot
Definición de funciones recursivas	Sí	No	Sí	Sí	Sí
Concepto de par asociado a tubería	No	No	No	No	No
Grilla infinita	Sí	Sí	Sí	Sí	No
Selección múltiple	Solo bloques anidados	Solo bloques anidados	Sí	Solo bloques anidados	No
Importar código puro	No	No	No	No permite importar pero muestra código puro de forma ilustrativa	No
Extensión del tipo de datos numérico a fracción	No	No	No	No	No

Tabla 2.5: Resumen comparativo de aplicaciones

En base a las funcionalidades estudiadas en las diferentes aplicaciones, resumidas en el cuadro anterior, y a las interfaces gráficas de estos, se observa que el lenguaje más similar a MateFun es Viskell. Este cuenta con algunas de las funcionalidades que se pretenden implementar, sin embargo su uso es más complejo ya que su dominio de aplicación es muy amplio y no está orientado al público objetivo de MateFun Infantil.

2.5. Elección de funcionalidades a implementar

A partir de los lenguajes estudiados, se llegó a ciertas conclusiones respecto a algunas de las funcionalidades que se pensaban implementar y a otras que fueron sustituidas.
En lo que respecta a la funcionalidad que permite extender la grilla, además que se entiende que lo más simple para el usuario es extenderla a medida que se añaden elementos sobre los bordes, todos los lenguajes que la implementan lo hacen de esta manera (con algunas variaciones como botones de zoom in y zoom out). Por tanto se va a implementar la extensión de esta forma.
También se estudió la posibilidad de realizar la selección de múltiples elementos, en donde Viskell es el único lenguaje que permite esta característica. Dado que la implementación del tablero de este último es muy distinta a la de MateFun, se encontraron bibliotecas que ofrecen una solución a la característica planteada. En el siguiente capítulo se mostrará la integración de estas bibliotecas al proyecto para así definir con cual de las relevadas se llevará a cabo la implementación.

En relación a la terminología que aplican, es muy variada, y no hay disponible documentación que justifique su uso, por tanto no es posible basarse en uno de los lenguajes en particular, más aún cuando está implementado en otro idioma. Sobre este aspecto se entiende que lo mejor es realizar una validación completa con docentes.
Con respecto al concepto de par asociado a tubería, al no utilizar tuberías ninguno de los lenguajes, no es posible realizar una comparación. Viskell es el lenguaje que más se asemeja gráficamente, pero no permite este tipo de funcionalidad.
Otro aspecto estudiado es el uso de la recursión, que la mayoría de los lenguajes implementa pero no explica del todo claro o no hacen foco en su funcionamiento. Acerca de esto, se distingue el juego Lighbot que con algunas imágenes explica su uso, y el autoguardado que implementan otros lenguajes para poder definir funciones recursivas e invocarlas al mismo tiempo. Sobre esta funcionalidad, se decidió junto a los tutores postergar su implementación, ya que se entendió que hay otras funcionalidades que en principio aportan mayor valor a la usabilidad del producto. En este caso, se decidió sustituirla por la implementación de un asistente que se encargará de guiar al usuario dándole mensajes de error, advertencia o acierto, entre otros, que se detallarán posteriormente. Esta característica se observa en todas las aplicaciones, ya que siempre hay algún tipo de guía para que los niños puedan utilizar de mejor forma el lenguaje, como por ejemplo en Lightbot 2.6a o en CODE.org 2.4.
Del mismo modo que con la funcionalidad anterior, se decidió dejar de lado la funcionalidad de importar el código MateFun puro, ya que los lenguajes que se analizaron no lo implementan. Únicamente CODE.org muestra código de forma ilustrativa. Además de esto, se encontró que todas las aplicaciones permiten realizar ejercicios que fueron creados previamente, por tanto se decidió sustituir esta funcionalidad por un gestor de desafíos. De las aplicaciones analizadas se decidió seguir al formato de presentación de los desafíos tal como está implementado en la aplicación Pilas Bloques. En la aplicación de referencia los desafíos se encuentran categorizados por edad escolar y luego ordenados por capítulos con dificultad ascendente. En el Capítulo 3.13 se definen las categorías definidas para presentar los desafíos de la aplicación.

Por último, en los objetivos iniciales se planteaba extender el tipo de datos numérico a enteros y racionales, luego de analizar la aplicación, se observa que soporta enteros y decimales, lo cual se acordó con los tutores que es suficiente, por tanto se decidió cancelar esta implementación. Dado esto, se resolvió dedicar este esfuerzo a la implementación de una funcionalidad que permita crear funciones a partir de la selección múltiple. Lo cual añadiría una mejora en la usabilidad a la hora de realizar estas creaciones.

Capítulo 3

Validación inicial

Luego del análisis del estado del arte, se trabajó en el proceso de definición del conjunto final de funcionalidades a implementar, destacando que se partió desde un conjunto potencial de funcionalidades que tuvo modificaciones en acuerdo con el equipo de trabajo y tutores. A continuación se detalla el conjunto final de funcionalidades:

- Selección múltiple.
- Creación de funciones a partir de la selección múltiple del punto anterior.
- Generalizar el concepto de par.
- Creación de un asistente que oriente al usuario.
- Grilla infinita.
- Implementación de un gestor de desafíos donde haya ejercicios precargados.

A partir de esto, y dado que uno de los objetivos del proyecto es realizar validaciones con expertos del dominio, se decidió realizar un diseño participativo con los mismos. Para ello, se contactó a tres docentes de primaria egresadas del Instituto Normal de Montevideo, y a un niño de 8 y una niña 11 años.
En esta etapa se evaluó la utilización de prototipos con el objetivo de mostrar la función de cada uno de los nuevos requerimientos. Tal como se presenta en el Capítulo 2 del texto Sommerville - Ingeniería de Software [26], los prototipos no tienen que ser ejecutables para ser útiles, los modelos de la interfaz de usuario del sistema pueden ser efectivos para ayudar a los usuarios a refinar un diseño de interfaz y trabajar a través de escenarios de uso. Uno de los motivos ya mencionados, hacer partícipes a los interesados en el diseño y adaptación de cada una de las funcionalidades, sumado a que el desarrollo de los prototipos es muy económico y suelen construirse en pocos días, fueron las principales razones por los cuales se procedió de esta manera. Esto permitió que los usuarios interactúen con esta interfaz, realizando sugerencias y a su vez planteando solicitudes que pasaron a ser evaluadas nuevamente en la fase previa al desarrollo.
Se optó por diseñar junto a los expertos las funcionalidades en donde existía una mayor cantidad de alternativas gráficas de implementación. Las mismas son: multitubería, asistente para ayuda de la aplicación y gestor de desafíos, para el resto de las funcionalidades el camino a seguir a nivel de implementación ya estaba bien definido.
Con respecto a la multitubería, se realizó el prototipo mediante la implementación en código para

la creación de la misma, y mediante imágenes para el desarmado de esta. Para validar las otras dos funcionalidades, se crearon prototipos únicamente con imágenes, utilizando editores gráficos.
Se decidió validar la primera de estas características con los docentes y con los niños, ya que existían varias dudas sobre su usabilidad. En relación a las otras dos, la validación se realizó únicamente con los docentes, ya que las dudas esencialmente eran conceptuales o de aporte de valor para los usuarios.
Previo a comenzar las validaciones, se introdujo tanto a los docentes como a los niños a la aplicación. En esta instancia, los docentes explicaron que las funcionalidades de nivel avanzado y completo son demasiado complejas para los escolares, por tanto, se continuó trabajando en el proyecto únicamente con el nivel básico.
A continuación se detalla el proceso de validación inicial de cada una de estas tres funcionalidades.

3.0.1. Multitubería

Para validar el uso de la multitubería, se creó un prototipo que se mostró a los docentes y niños, con una tubería de tipo función llamada "CM" (crear multitubería), con una entrada a la izquierda y una entrada a la derecha. La salida de esta función arroja como resultado la concatenación de los dos valores de entrada, y la tubería de salida gráficamente también muestra la concatenación de los dos colores que representan a los dos tipos de entrada, como se observa en la Figura 3.1.

Figura 3.1: Creación de multitubería con una entrada de tipo numérico y una de tipo color

Esta función también permite realizar nuevas concatenaciones, con la salvedad que cuando una de las entradas de la función, ya contiene más de un color, en la salida será representada con el color violeta (color que siempre representará al tipo multitubería). Esta decisión se tomó por una limitante gráfica de espacio, y por aspectos del código que dificultaban concatenar varias lineas más de colores. Esto último es uno de los aspectos que más interesaba validar.
En la Figura 3.2 se observa lo antes mencionado, donde se concatenan un número, un color, y nuevamente otro número.

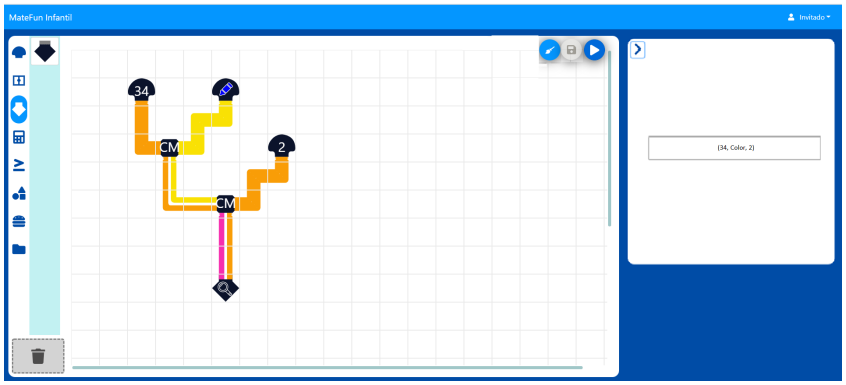


Figura 3.2: Creación de multitubería con tres valores concatenados

Acerca de esta modalidad de armado de la multitubería, los docentes consideraron que es una forma adecuada de construcción que los niños son capaces de comprender. Al validarlo con los niños, una vez explicado el significado del color rosado en la multitubería, no hubo dificultades para entender su armado.
Otra característica que se pretendía validar, es la función de desarmado de la multitubería. Para ello había dos opciones posibles. La primera, una tubería de tipo función llamada “desarmar multitubería”, cuya entrada izquierda era una multitubería, y en su entrada derecha un número. El resultado de esta función, como muestra la siguiente Figura 3.3, era el elemento de la multitubería, ubicado en la posición que indicaba este último número.

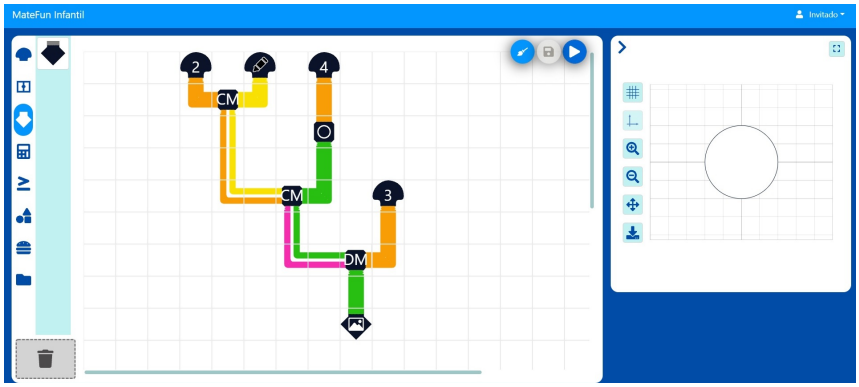


Figura 3.3: Desarmado con índice

La segunda opción, eran dos funciones, cada una con una única entrada de tipo multitubería, una llamada “desarmar por izquierda” y que la otra “desarmar por derecha”. La primera, retornaba como resultado la multitubería que ingresó por izquierda a la multitubería que se conecta con dicha función de desarmado. De forma análoga, el desarmado por derecha, lo hacía con el elemento de la derecha. Esto se mostró a los docentes y niños, con algunos casos a modo ilustrativo como el de la siguiente Figura 4.33:

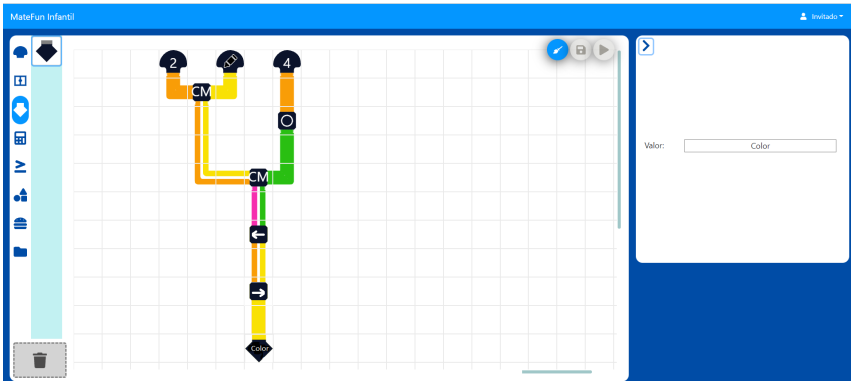


Figura 3.4: Desarmado con izquierda y derecha

Al presentar a los niños las dos opciones, frente a algunos casos particulares del desarmado con índice numérico, mostraban dificultad para determinar cual era el elemento resultado. Un ejemplo de esto es la Figura 3.5 en donde los niños pensaban que el elemento 3 de la multitubería era el color, ya que contaban de izquierda a derecha según el orden de aparición de las constantes en el tablero.

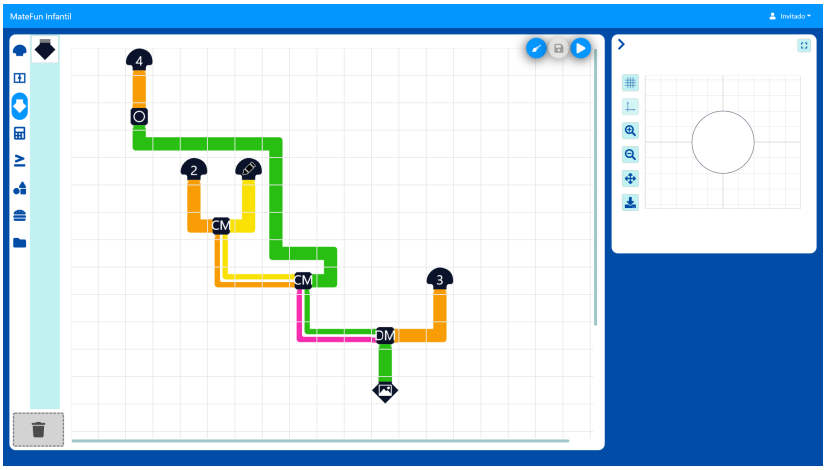


Figura 3.5: Desarmado con índice con dificultad

A su vez los docentes plantearon que el desarmado por izquierda y derecha, implica que el niño aplique un mayor razonamiento lógico al realizar la regresión. Por tanto el equipo de trabajo decidió optar por la opción de desarmado de multitubería por la izquierda y derecha.

Otros aspectos de la multitubería que se buscaban validar, eran detalles gráficos, que al ser la primera vez que una tubería poseía colores no simétricos, implicaba una toma de decisiones a la hora de realizar ciertos cruces de las mismas. Para ello el equipo de trabajo, junto con los tutores, se reunió con Ewelina Bakala, especialista en interfaces de usuario.
En primera instancia, se tenía una posible opción para los cruces de tuberías, como muestra la Figura 3.6. Aquí se observa como en el centro de la imagen, en donde hay una tubería en forma de “+”, el brazo izquierdo continúa con el color amarillo en la izquierda de la multitubería, a diferencia de los

demás.

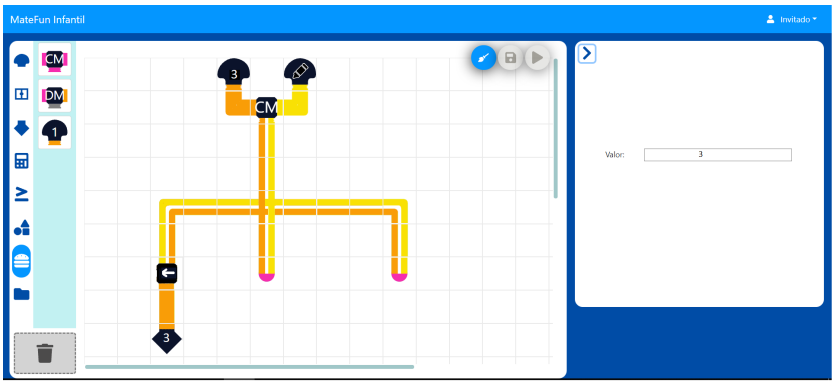


Figura 3.6: Cruces de multituberías inicial

Luego de la reunión con la especialista, surgió una segunda alternativa 3.7. En esta opción, todas las salidas del cruce en forma de "+" coinciden en la posición de los colores.

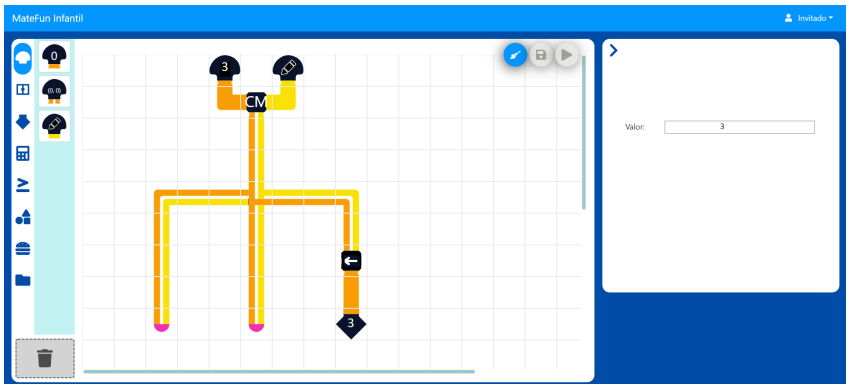


Figura 3.7: Cruces de multituberías alternativo

La especialista recomendó validar con los niños la mejor opción, y luego de realizarlo, se decidió implementar la alternativa de la Figura 3.7, ya que en la opción de la Figura 3.6, al invertirse en el cruce los colores de la multitubería, los niños tendían a equivocarse en el resultado del desarmado.

3.0.2. Asistente

Con la creación de un asistente se busca incorporar a la aplicación, mediante sus gestos, una interfaz intuitiva que permita comunicar las diferentes situaciones que se dan durante la ejecución de un conjunto de tuberías. Junto a estos iría acompañado el texto con el detalle del mensaje.
El objetivo de esta validación, era corroborar con los docentes la idea de que un asistente aporta valor a la aplicación mejorando su usabilidad y las principales características que deberían tener los textos que muestre el mismo. Para esto se diseñó gráficamente un prototipo con varios mensajes, a continuación se presentarán algunos de ellos a modo de ejemplo:

- La Figura 3.8 muestra el mensaje de bienvenida del asistente:

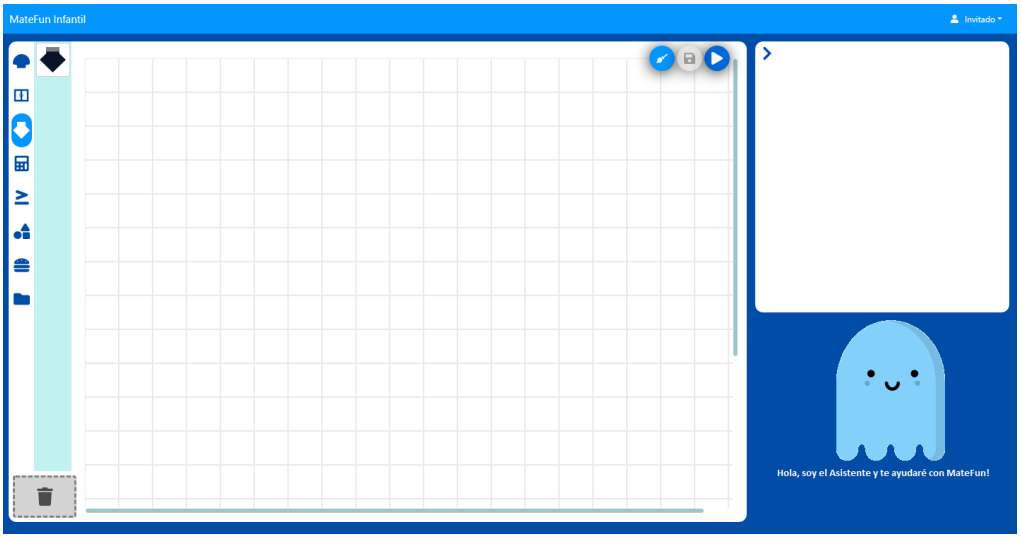


Figura 3.8: Mensaje de bienvenida del asistente

- La Figura 3.9 ejemplifica un mensaje de error cuando no coinciden dos tipos de tubería que se desean unir.

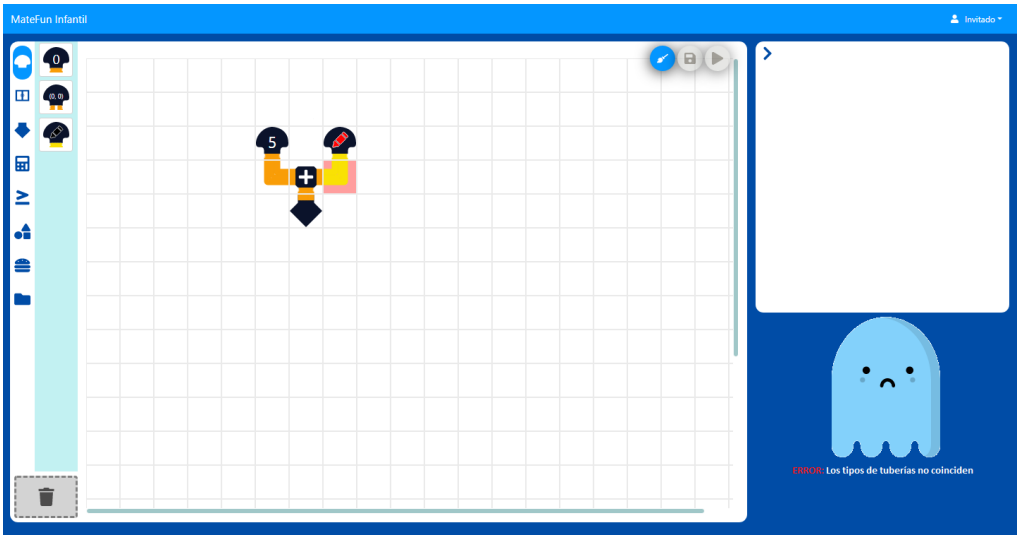


Figura 3.9: Mensaje de error del asistente

- La Figura 3.10 muestra un caso de mensaje de advertencia en donde aún no se conectaron todas las tuberías del tablero:

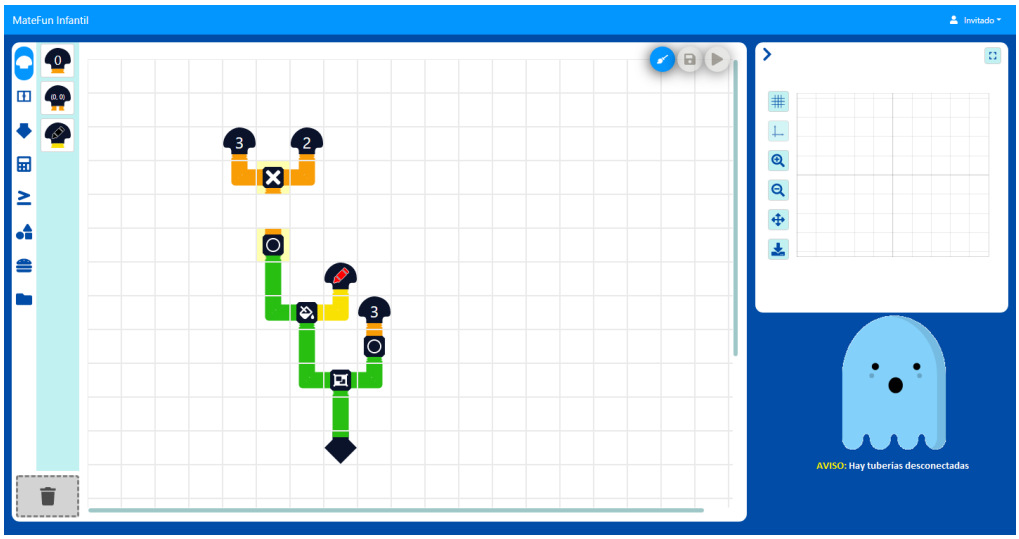


Figura 3.10: Mensaje de advertencia del asistente

- Por último, en la Figura 3.11 se muestra un mensaje de éxito al ejecutar un conjunto de tuberías.

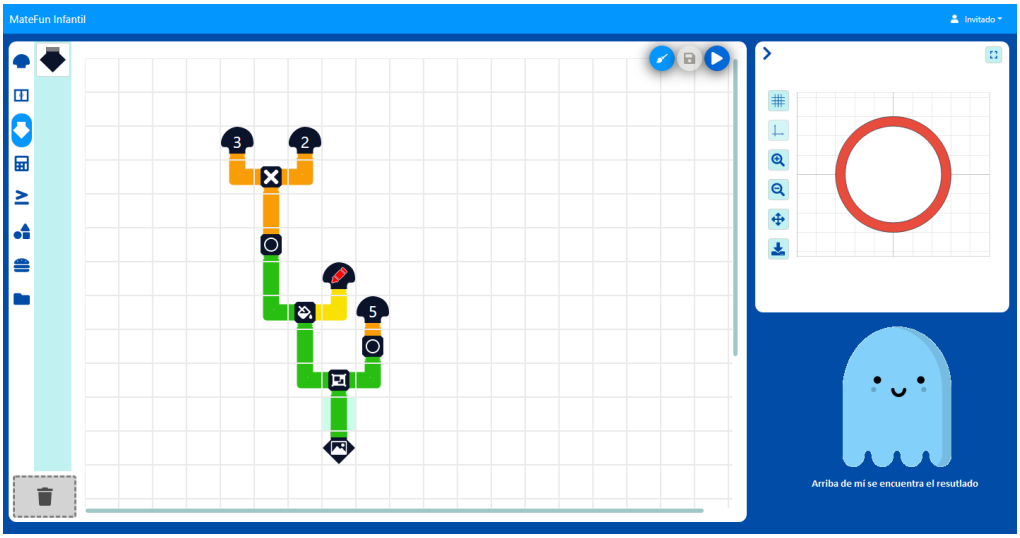


Figura 3.11: Mensaje de éxito del asistente

Luego de presentar a los docentes el prototipo, se obtuvieron algunas recomendaciones a tener en cuenta. En primer lugar, el asistente les parece muy útil, ya que facilita a los niños el uso de la herramienta sin tanta o ninguna ayuda de ellos.
Con respecto a aspectos puntuales, se destaca que los textos deben ser cortos, concretos e imperativos. En caso que haya varios mensajes que deba mostrar el asistente, como por ejemplo, dos alertas, mostrar únicamente un mensaje, ya sea genérico, o una de las alertas. El tamaño de la letra debe ser

más grande, y el tipo de la misma podría ser caricaturesco. La letra debería ser en minúscula ya que para el público objetivo (niños de últimos años de primaria), el texto en letra mayúscula satura la vista.
Los mensajes de alerta serían con las palabras *REVISA*, en color rojo, buscando indicar que hay un error y se debe corregir. Se recomendó no poner la palabra *ERROR* con la intención de no desmotivar a los niños. El mensaje de advertencia se representaría en amarillo con la palabra *CUIDADO*, y en verde el mensaje de éxito con la palabra *CORRECTO*.
Una recomendación que dieron, fue que cuando se muestre el resultado, el asistente mire hacia arriba con la leyenda "CORRECTO: Arriba se encuentra el resultado".
Otra idea que aportaron los docentes es que a cada elemento de MateFun se le podría hacer click derecho y el asistente dar ideas de como utilizarlo.
Posterior a la validación con docentes, junto a los tutores, surgió la idea que el panel de resultado podría ser un globo de texto completo en donde se muestre tanto la mensajería como los resultados.
A partir de las recomendaciones recibidas el asistente se implementó como en la Figura 3.12.

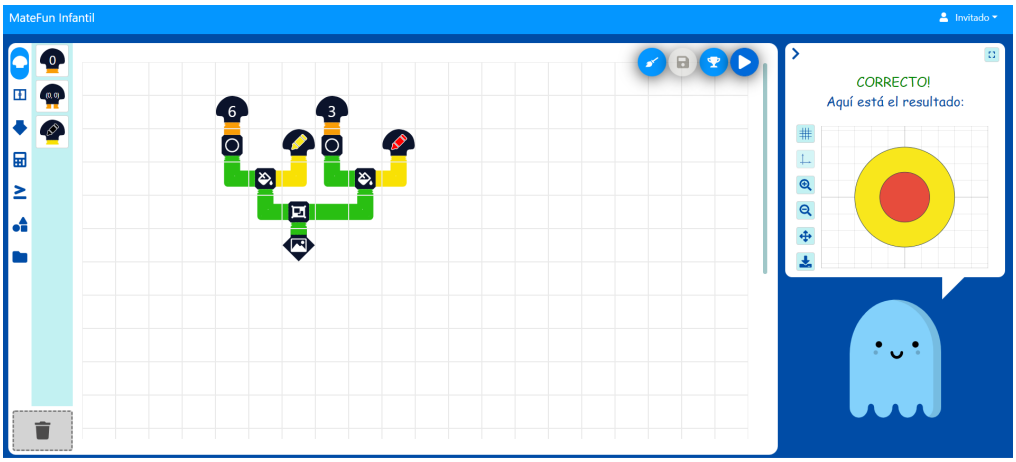
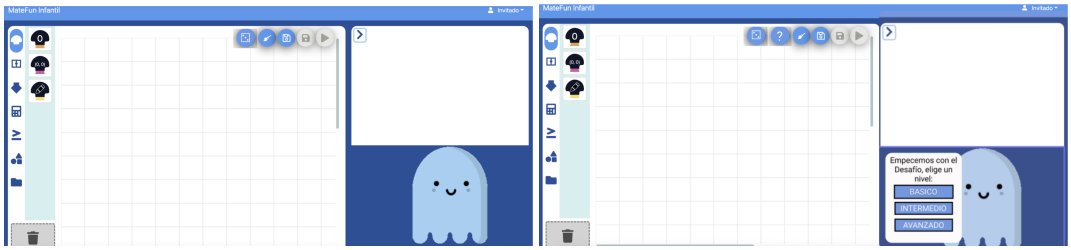


Figura 3.12: Resultado figura 2D

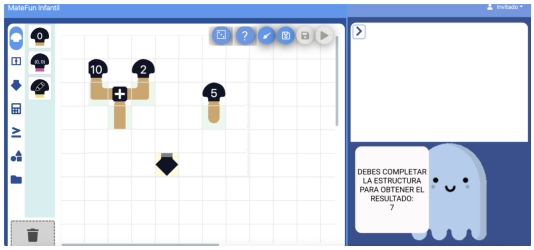
3.0.3. Gestor de desafíos

La otra funcionalidad que se buscaba validar previo a su implementación, era la posibilidad de tener un gestor de desafíos en donde existieran ejercicios precargados para que el niño los resuelva. Para ello se mostraron las figuras 3.13a y 3.13b a modo de prototipo, en donde se observa un nuevo botón con un símbolo de dado, que al seleccionarlo permite elegir la dificultad del desafío.



(a) Pantalla principal con botón de desafíos

(b) Selección de la dificultad del desafío.



(c) Desafío desplegado en el tablero para resolver.

Figura 3.13: Prototipo de acceso a desafíos

Una vez seleccionada la dificultad del desafío, se desplegaría en el tablero el mismo, y el asistente mostraría la consigna del desafío en cuestión, como muestra la Figura 3.13c. Además de esto, también se mostraría un botón de ayuda (en este caso representado con el signo de interrogación), en donde al seleccionarlo, el asistente daría una pista para resolver el desafío.

Sobre esta funcionalidad, en términos generales los docentes nuevamente destacaron su importancia, ya que permite ejercitar al niño sin que el maestro primero tenga que plantear una situación problema.

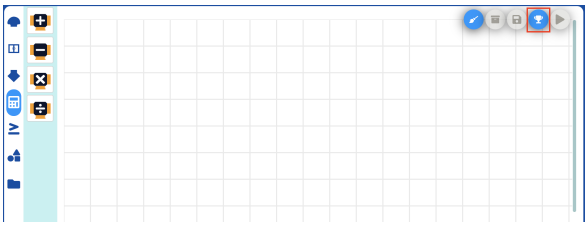
Con respecto a aspectos puntuales de esta funcionalidad, recomendaron que el ícono para ingresar a los desafíos sea un trofeo, ya que el dado representa lo lúdico, lo cual está vinculado con toda la herramienta. Mencionaron que el ícono de ayuda sería mejor que estuviese representado por una lamparita encendida. Nuevamente se hizo alusión a que los textos de los desafíos deben ser cortos, concretos, imperativos y además no deben presentar ambigüedades.

También plantearon algunos tipos de desafíos que aportarían valor para que los resuelvan los niños:

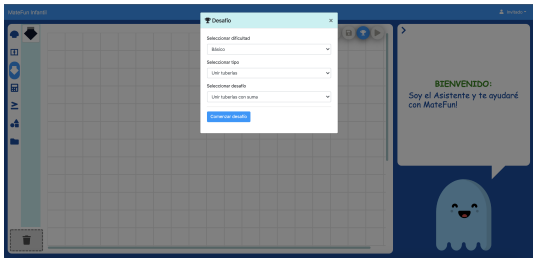
- Conjuntos de tuberías presentadas en el tablero en donde falten signos de operación y el niño deba sacarlos de la paleta para que el resultado sea el solicitado.
- En el tablero se encuentran todas las tuberías de operaciones y constantes y el niño debe ordenarlas para que el resultado sea el solicitado.
- Se brinda un conjunto de tuberías ya armada, con alguna variable y el niño debe modificarla para que el resultado sea el solicitado.

Por último, también fue mencionado que sería deseable que los propios docentes puedan cargar los desafíos para que los niños posteriormente los resuelvan.

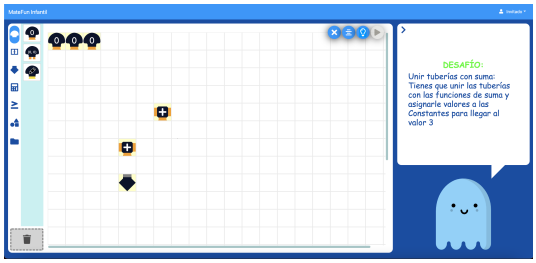
Posterior a la implementación, el resultado de esta funcionalidad se puede observar en las Figuras 3.14:



(a) Acceder a desafíos



(b) Modal para la selección de desafío



(c) Ejemplo de desafío del tipo Unir tuberías

Figura 3.14: Implementación de gestor de desafíos

Capítulo 4

Implementación de funcionalidades

En este capítulo, teniendo en cuenta lo validado en el Capítulo 3, se detallan los principales aspectos de la implementación de las funcionalidades, tales como:

- Identificación de componentes de código involucrados.
- Bibliotecas con las que se experimentó.
- Problemas que surgieron durante la implementación.
- Decisiones que se tomaron en base a lo anterior.
- Algoritmos relevantes en la implementación.

En primer lugar se muestran los componentes de la estructura en capas definida originalmente, que han sido involucrados en el desarrollo de las nuevas funcionalidades. Para esto se cita el diagrama de componentes de cada una de las capas, se describe el uso y se detallan los cambios realizados de acuerdo a las nuevas funcionalidades. Es importante notar que aquellos componentes que están mencionados sin incluir una descripción en detalle no se vieron implicados en modificaciones por lo tanto su definición y propósito se mantienen. En caso de necesitar profundizar sobre alguno de estos componentes es recomendable revisar la documentación original de la primera etapa del proyecto [31].

4.1. Tecnología utilizada

Como se mencionó anteriormente, Matefun Infantil Web está implementado en **React** [19], una biblioteca de JavaScript [11] de código abierto desarrollada por Meta [15], que permite la creación de interfaces de usuario de manera sencilla. Está basado en componentes que manejan su propio estado y tienen la lógica escrita en JavaScript. Una de las principales ventajas es que utiliza DOM [5] virtuales, siendo DOM una interfaz de programación de aplicaciones para XML y HTML que define la estructura lógica de los documentos y como acceder o modificar los mismos. El DOM virtual permite actualizar únicamente las secciones modificadas en el caso de que existan diferencias entre el HTML anterior y el actual, otorgando una mejor performance.
Además Matefun Infantil Web cuenta con la biblioteca **React Redux** [22] que permite mantener el estado de la aplicación en memoria y que los componentes puedan leerlo y actualizarlo mediante

acciones. Y la biblioteca **ReactDND** [20] utilizada para crear interfaces donde se puedan arrastrar y soltar componentes.
Para los gráficos se utiliza el lenguaje **SVG** [28] (Scalable Vector Graphics) en el cual se pueden escribir gráficos de 2 dimensiones en formato XML, pudiendo ser formas vectoriales gráficas, imágenes y/o textos. Además los gráficos definidos pueden ser interactivos y dinámicos, incorporando animaciones.

4.2. Estructura MateFun Infantil

En esta sección se realiza una breve introducción a la estructura de la aplicación previo a la ejecución del presente proyecto, tomando como referencia la descripción de la arquitectura del proyecto anterior descripta en la Sección 5 del informe citado [31]. La estructura consta de capas que interactúan entre ellas como se puede visualizar en la Figura 4.1. A continuación se listan los elementos de la estructura de capas:

- Interfaz de Usuario:

  Las clases que forman la interfaz con el usuario y la implementación de la biblioteca Redux [22] que sirve para manejar su estado.

- Modelo de Datos:

  Implementación del modelo de la aplicación.

- Lógica:

  Capa donde se implementan funciones y APIs.

- Conexión:

  Conexión con el servidor de Java.

- Biblioteca de Gráficos 2D:

  Biblioteca externa que centraliza la lógica de los gráficos en 2 dimensiones.

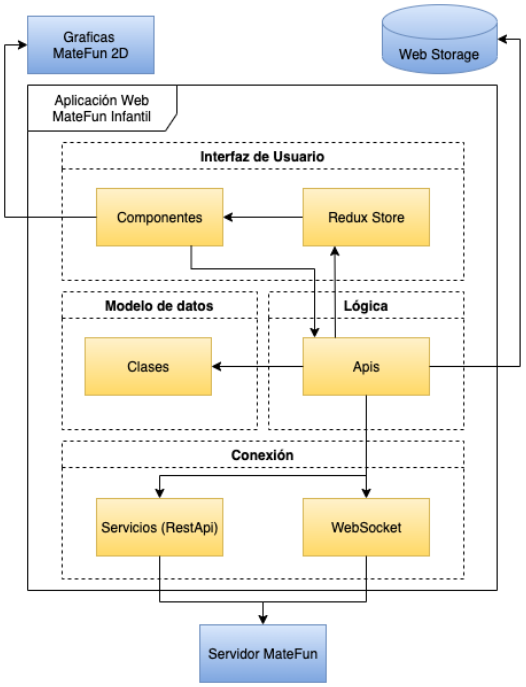


Figura 4.1: Arquitectura MateFun Infantil

4.2.1. Descripción de capas

En la Figura 4.2 se muestra con mayor detalle las capas Redux Store, APIs y Modelos de datos, y en la Figura 4.3 la capa Componentes.

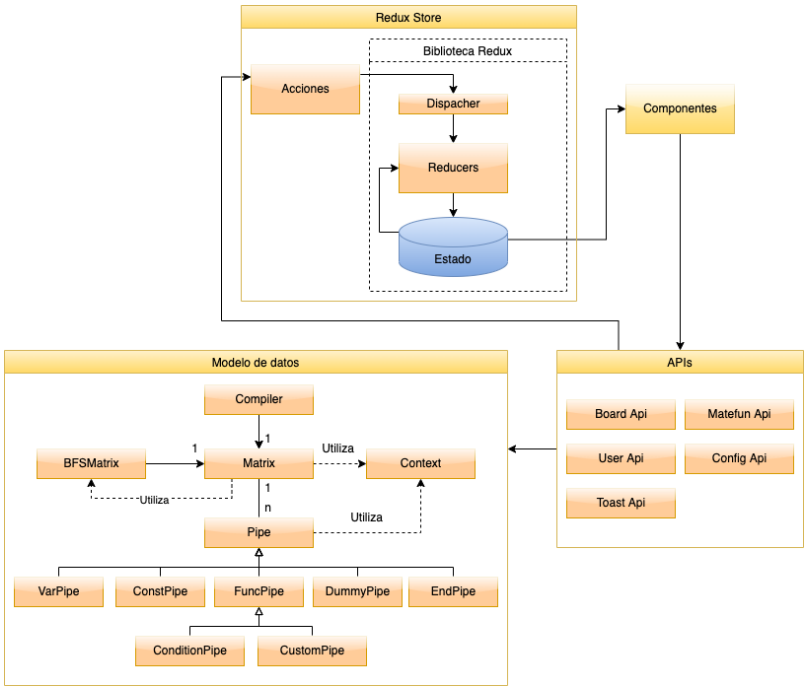


Figura 4.2: Capas MateFun Infantil

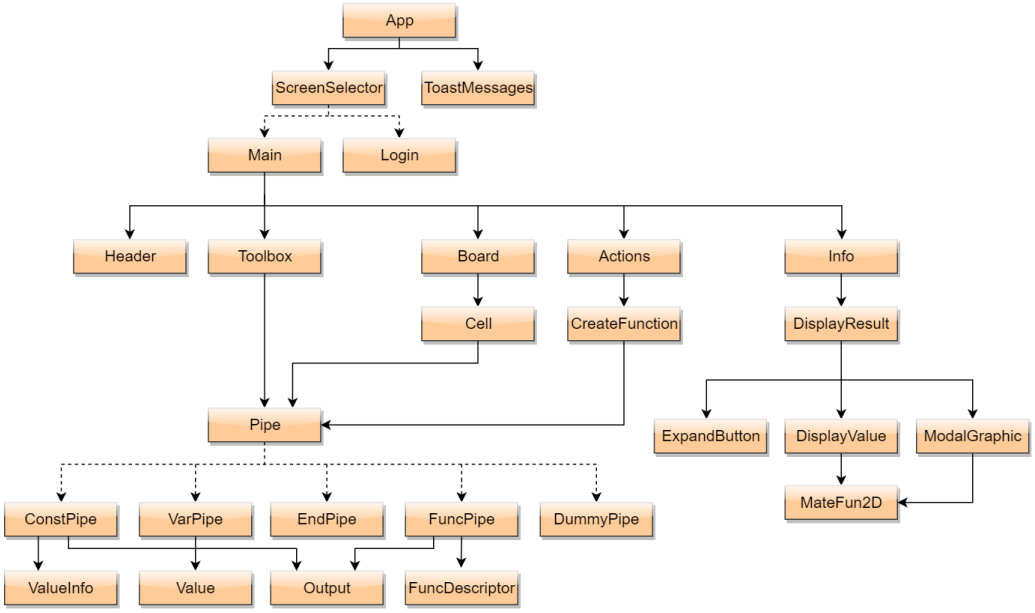


Figura 4.3: Diagrama de componentes Interfaz

Componentes

Son los componentes de React que implementan la interfaz de usuario, donde cada componente se ocupa de mostrar cierto elemento en la pantalla y se organizan de forma jerárquica. Es decir, los componentes de más arriba en la Figura 4.3 engloban los componentes que están debajo. La pantalla que se muestra en la aplicación es determinada por el componente **ScreenSelector**, que puede ser la pantalla de **Login** o **Main**, siendo esta última la pantalla principal de MateFun. **ToastMessages** es el componente a cargo de mostrar mensajes en pantalla (errores, advertencia, éxito o información).
La pantalla principal (Main) está compuesta por los componentes **Header** encargado de generar el cabezal, **Toolbox** que retorna las herramientas, **Info** que es responsable de la visualización de resultados, **Board** quien es el encargado del tablero que contiene celdas (**Cell**) que a su vez puede contener tuberías (**Pipe**), y por último el componente **Actions** es el encargado de desplegar los botones que permiten distintas acciones como Limpiar el tablero, Guardar una función y Probar (Ejecutar). El componente **CreateFunction** dentro de **Actions** es el encargado del modal para el guardado de funciones.
Dentro del componente **Info** se encuentra el componente **DisplayResult** que es el encargado de mostrar los resultados al ejecutar. **ExpandButton** se encarga de dejar visible o no la sección de resultados, y **DisplayValue** y **ModelGraphic** se encargan de mostrar el componenete MateFun2D en pantalla normal y pantalla completa respectivamente.
ConstPipe, **VarPipe**, **EndPipe**, **FuncPipe**, **DummyPipe** son los distintos tipos que puede tener el componente **Pipe**. Las constantes y variables pueden ser colores o número y un ejemplo de función es Suma, que recibe dos números y devuelve como resultado la suma, o Pintar que recibe una figura y un color, y devuelve una figura pintada. En el manual de usuario de la primera versión del sistema se encuentra la lista completa de tuberías [31].

Redux Store

Como fue mencionado anteriormente para almacenar y manejar el estado de la aplicación se utiliza la biblioteca React-Redux [22]. Es necesario emitir una **Acción** llamando la función **Dispatch** para modificar el estado de la aplicación. Los **Reducer** indican como debe ser modificado el estado. Los componentes se actualizan en cada actualización de estado.
Hasta ahora en el estado se guarda la configuración de la aplicación (nivel de dificultad), entorno (archivos necesarios en memoria), variables que indican el estado de la matriz, siguiente mensaje a mostrar y los datos del usuario logueado.

Modelo de Datos

Se utiliza una matriz (**Matrix**) contenida por la clase **Compiler** para mantener el estado de la aplicación donde las celdas contienen tuberías **Pipe**. **Compiler** además de la matriz contiene conceptos propios de MateFun. La clase **BFSMatrix** es quien se encarga de recorrer la matriz y **Context** guarda los nodos visitados.

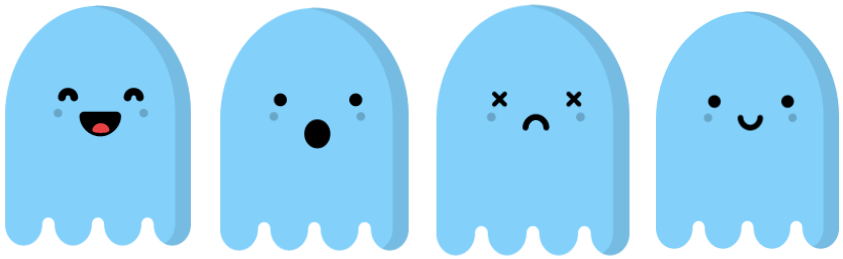
APIs

Comprende el conjunto de APIs que implementan la lógica del sistema y brindan la operaciones a los demás componentes para realizar la interacción dentro del modelo planteado originalmente.
Dicho modelo esta compuesto por las siguientes APIs:

- **Board API:** Realiza el manejo del estado de la matriz y el procesamiento de todas las interacciones realizadas por el usuario durante sobre el tablero. A su vez las respuestas gráficas que se obtienen sobre el mismo son comunicadas a la interfaz gráfica.
- **User API:** Resuelve la gestión del usuario, desde el inicio hasta el cierre de la sesión de trabajo. Así como también tiene la tarea de mantener el token mientras la sesión de trabajo se mantiene activa.
- **MateFun API:** Se encarga de proveer la lógica que permite al usuario guardar, editar y borrar funciones MateFun. También incluye la implementación necesaria para realizar importación y exportación de funciones a través del modal.
- **Toast API:** Genera mensajes de notificaciones de tipo pop-ups ante cualquier evento que sea necesario informar al usuario.
- **Config API:** En esta API se lleva a cabo la configuración del nivel de complejidad del sistema que el usuario puede seleccionar en la sección de configuración.

4.3. Asistente

Para la implementación del asistente se utilizó la biblioteca react-kawaii [21], la cual otorga ilustraciones fáciles de modificar y brinda la posibilidad de realizar las animaciones que se validaron en el prototipo, generando un impacto visual acorde al estado de éxito, alerta o error de la aplicación, como se muestra en el conjunto de figuras 4.4. Se realizó una búsqueda de bibliotecas que permitan representar estados de ánimo y que el personaje no fuese humano, buscando un estilo más caricaturesco. También se estudió la biblioteca de React cartoon-avatars [1] pero rápidamente se desechó al encontrar react-kawaii, ya que esta última cumplía los requisitos antes mencionados y además resultaba más sencilla su implementación.



(a) Asistente Bienvenida (b) Asistente Alerta (c) Asistente Error (d) Asistente Éxito

Figura 4.4: Gestos asistente

En la Figura 4.5 se muestra la ubicación del asistente en la aplicación, utilizando el espacio vacío que se encontraba bajo el panel de resultado. En esta imagen se presenta el mensaje de bienvenida a la aplicación. El sector original de resultados, tal como se validó, se convirtió en un globo de texto que parte del asistente, de forma tal de mantener toda la comunicación de resultados y mensajes efectuados por la aplicación a partir de éste.
Si bien se entiende que el asistente es de gran ayuda para el usuario, por lo validado en la etapa anterior, se decidió permitir la posibilidad de ocultarlo junto al panel de resultado, con el mismo

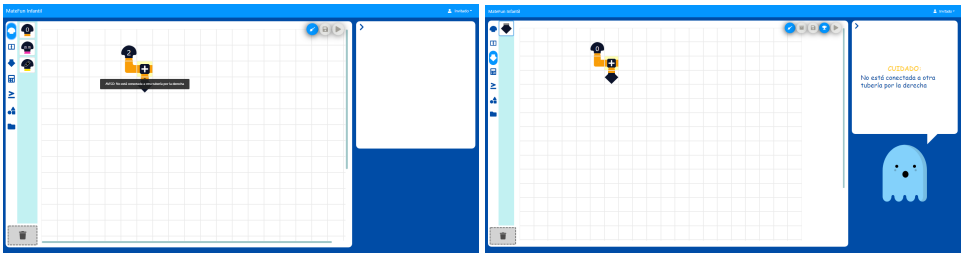
botón que se esconde este último. Este se encuentra en la zona superior izquierda del globo de texto del asistente.

Figura 4.5: Asistente MateFun bienvenida

Previo a la implementación del asistente, la aplicación contaba con tres vías diferentes de comunicación hacia el usuario que se describen a continuación:

- **Resultados:** Tanto valores numérico como gráficos eran mostrados en la sección lateral derecha de la pantalla (recuadro blanco).
- **Mensajes en el tablero:** Se mostraban mensajes de alerta o error asociados a celdas específicas del tablero. Para leer el contenido de los mensajes era necesario colocar el cursor sobre la celda en cuestión para visualizar el tooltip asociado.
- **Mensajes flotantes:** También llamados “toast messages”, eran desplegados en la parte superior central de la aplicación, generalmente estos mensajes indicaban errores o aciertos relativos al manejo de funciones.

En la Figura 4.6a se puede observar como se mostraba en pantalla previo a la implementación una alerta en el tablero, mientras que en la Figura 4.6b se presenta la nueva forma de indicar una alerta.

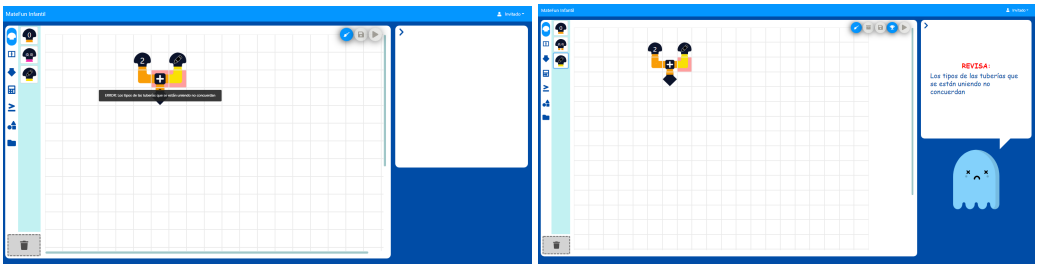


(a) Asistente con solución previa de alertas

(b) Asistente con solución actual de alertas

Figura 4.6: Alertas previo y posterior a la implementación

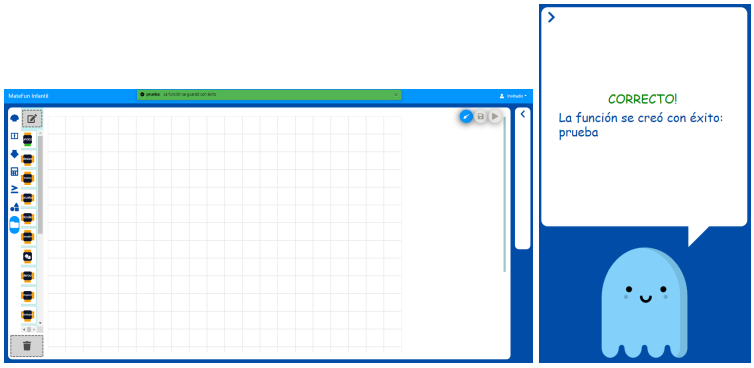
De la misma forma en las figuras 4.7a y 4.7b se observa el cambio en la forma de mostrar mensajes de error.



(a) Asistente solución previa de errores (b) Asistente solución actual de errores

Figura 4.7: Errores previo y posterior a la implementación

En la Figura 4.8a se muestra el mensaje de tipo “toast”, notificando el correcto guardado de una función. Tal como se indicó al principio de la sección, estos mensajes dejaron de ser desplegados de esta forma y pasaron a ser incluidos dentro del cuadro de texto del asistente tal como se observa en la Figura 4.8b.



(a) Mensaje Toast guardar función (b) Mensaje guardar función

Figura 4.8: Errores previo y posterior a la implementación

Por último, en la Figura 4.9 se observa el resultado de una ejecución que tiene como salida la composición de figuras. En este caso se muestra un texto indicado por el asistente informando la validez de la ejecución junto con el resultado obtenido.

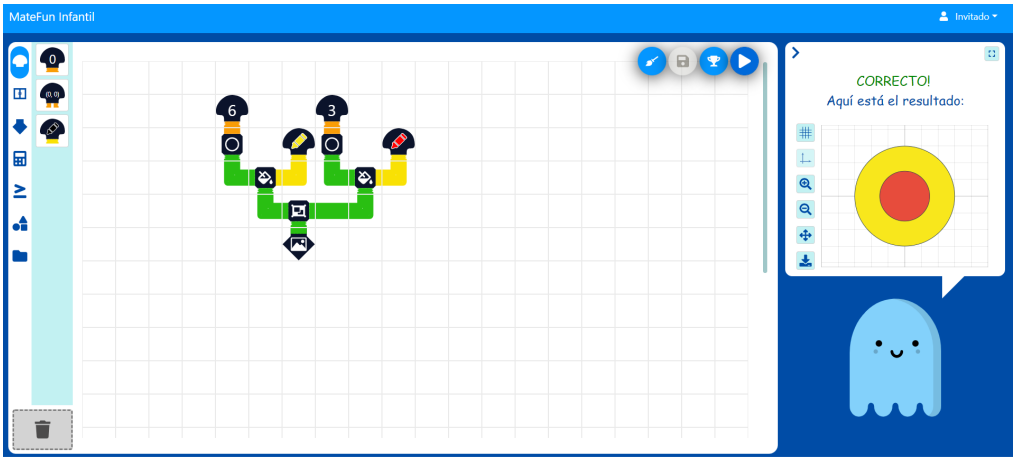


Figura 4.9: Resultado figura 2D

Acerca de los cambios realizados sobre los distintos componentes para implementar esta funcionalidad, en primer lugar se destaca el componente **DisplayResult**, donde se implementó la consolidación de toda la mensajería desplegada por la aplicación. Para esto el componente muestra en pantalla todos los mensajes de tipo alerta, éxito o error con un color identificativo en conjunto con los resultados obtenidos. Al mismo, se incorporó la posibilidad de agregar nuevos mensajes, asignándoles un orden de prioridad a la hora de ser mostrados. Respecto a este orden, ante la existencia de mensajes de tipo error y de tipo alertas, se muestra únicamente el mensaje de alerta. Cuando solo hay mensajes de alerta o de error, pero son dos o más mensajes, se muestra únicamente un mensaje genérico. Esto fue validado, y es para evitar que el niño ante tanto texto ignore el mensaje. A su vez, en caso de obtener un resultado y existir alertas en otras celdas que no son relevantes al mismo, se despliega solo el resultado. El desarrollo de este componente está altamente ligado al desarrollo del gestor de desafíos, por lo tanto todos los mensajes asociados a los desafíos que puedan ser ejecutados en la plataforma, son procesados y desplegados en el componente mencionado. Tal como se observa en el diagrama 4.3 el componente está compuesto por **DisplayValue**, **ModalGraphic** y **MateFun2D**. Estos componentes se mantienen sin cambios.
En el componente **Main** se incorpora el asistente de la aplicación, se define la configuración de la biblioteca encargada de la representación gráfica y las funciones que permiten obtener el estado de la aplicación para asignar en cada momento las reacciones al asistente. En este se realiza la importación de la biblioteca ya mencionada. Se destaca la función *getMood* 4.10 que se encarga de evaluar el estado actual de la aplicación teniendo en cuenta el tablero, posible ejecución, desarrollo de desafío, y el módulo de procesamiento de funciones. La función retorna el estado de ánimo actual del asistente que representa el estado de la aplicación en cada momento.

```
//board: tablero
//canProcess: se puede procesar para obtener resultado
//isChallengeMode: Está en modo desafío
//showChallengeText: Mostrar texto del desafío
//resultValue: resultado obtenido luego de ejecución
getMood(board, canProcess, isChallengeMode, showChallengeText, resultValue) {

    Si está en modo desafío y el resultado actual es null:
        //Se esta ejecutando un desafio pero aun no hay resultado
        var challengeMood = true;
    Si está en modo desafío:
        //Se esta ejecutando un desafio y ya se tiene un resultado, se asigna en challengeMood el resultado
        var challengeMood = board.getChallengeResult()

    //Obtiene los eerores del tablero si existen
    getBoardText()
    Si está en modo desafío y, challengeMood es falso o el tablero tiene errores:
        //Ante existencia de errores o resultado incorrecto del desafio
        return "ko"
    Sino, si no se debe mostrar el texto del desafío, existen alertas en el tablero y no se puede procesar para obtener resultado:
        //Ante existencia de warnings, que no sea necesario mostrar la letra del desafío y que no se pueda ejecutar el programa
        return "shocked"
    Sino, si está en modo desafío y, se debe que mostrar el texto del desafío o challengeMood = true
        //Si se esta ejecutando un desafio y ademas se necesita mostrar la letra o el resultado es el esperado
        return "happy"
    Sino:
        return "blissful"
}
```

Figura 4.10: Pseudocódigo de función getMood Asistente

4.4. Extensión de la grilla

En primer lugar, se tomó la decisión de extender la grilla únicamente cuando se colocan tuberías en las últimas tres filas o columnas de la misma. En estos casos la grilla se agranda tres filas o columnas, según corresponda. Esta extensión se realiza únicamente hacia abajo y la derecha, ya que el componente del tablero lo permite únicamente de esta forma.
Evaluando el código mediante el cual se implementó el tablero en el ya mencionado componente **Board API**, se observó que existían dos constantes que manejaban las dimensiones del mismo, *BOARD_ROWS* y *BOARD_COLS*, ambas con valor 20. Esto determinaba que el tablero utilizado sea de tamaño fijo sin posibilidad de extensión durante la ejecución.
Dentro del componente **Board API**, se modificaron las funciones *movePipe* y *updateMatrix* para actualizar el tamaño del tablero en caso que corresponda, tal como se muestra en las siguientes figuras:

```
export function movePipe(pos, pipe, Matrix) {
    //Mueve el Pipe "pipe" a la Posicion "pos"
    Matrix.movePipeInMatrix(pos.x, pos.y, pipe.pos);
    {addRows,addColumns} = extendMatrix(pos);
    //Las variables addRows y addColumns determinan si se debe extender la matriz por filas y/o columnas
    updateMatrix(Matrix,addRows,addColumns);
}
```

(a) Función movePipe

```
//Matrix: matriz en la que se está trabajando
//addRows, addColumns: indican si hay que extender filas y columnas respectivamente
//isEditMode, isChallengeMode: globales que indican si es modo de edición o de desafío respectivamente
function updateMatrix(Matrix, addRows, addColumns) {
    //Se actualiza la matriz y si es necesario se extiende por filas y/o columnas
    var cantRows = Matrix.getRows;
    var cantCol = Matrix.getCols

    if(addColumns){
        cantCol = cantCol + 10;
    };

    if(addRows){
        cantRows = cantRows + 10;
    };

    //No es modo de edición ni desafío
    if(!isEditMode && !isChallengeMode) {
        if (addColumns || addRows) {
            //Se edita la matrz
            editMatrix(addRows, addColumns);
        };
    }
    //Se actualiza el tablero
    updateBoard();
}
```

(b) Función updateMatrix

Figura 4.11: Código de extensión de grilla

La lógica de esta implementación se basa en que al colocar una tubería en el tablero, se invoca la función *movePipe* 4.11a, la cual llama a la función *extendMatrix* encargada de determinar si la grilla debe o no extenderse. Una vez resuelto esto, la función *extendMatrix* 4.11b realiza la modificación del tablero.

Al realizar la implementación, se observó que a medida que el tamaño del tablero crecía se experimentaba lentitud en la interfaz a la hora de mostrar cuando una tubería es “soltada” en una celda, posterior al arrastre de la misma. Este problema se debe a una limitante en la biblioteca mencionada anteriormente **React DnD** [20], utilizada para manejar el arrastrar y soltar tuberías y se encuentra reportado en el repositorio oficial[1].

Dado que no fue posible la utilización de otra biblioteca ya que esto implicaba una refactorización del código legado demasiado grande, se optó por la alternativa de extender la grilla de a tres filas o columnas, hasta el límite de 700 celdas (aproximadamente cinco extensiones), en donde se aprecia una lentitud menor que en lo antes mencionado.

En las figuras 4.12a y 4.12b se aprecia como se extiende la grilla al colocar una tubería en la última columna.

[1] https://github.com/react-dnd/react-dnd/issues/421

(a) Tubería colocada para extender la grilla

(b) Grilla extendida luego de colocar tubería

Figura 4.12: Extensión de la grilla

Esta extensión limitada, está acompañada de mensajes de alerta brindados por el asistente al llegar al límite de la misma. En la Figura 4.13a se observa como el asistente da aviso que esa fue la última vez que se pudo extender la grilla. En la Figura 4.13b, luego de la última extensión permitida, cuando se coloca nuevamente una tubería en una de las últimas tres columnas, el asistente indica que el tablero ya no se puede extender más. Este mensaje lo muestra una única vez, ya que sino resultaría bastante molesto que lo recordara constantemente.

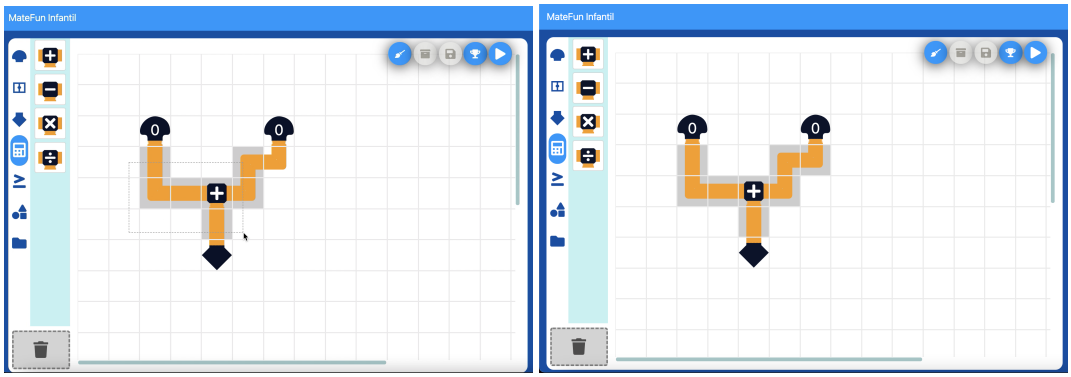
(a) Mensaje de advertencia de última extensión.

(b) Mensaje de advertencia que indica que ya no se puede extender más.

Figura 4.13: Límite de extensión de la grilla

4.5. Selección múltiple

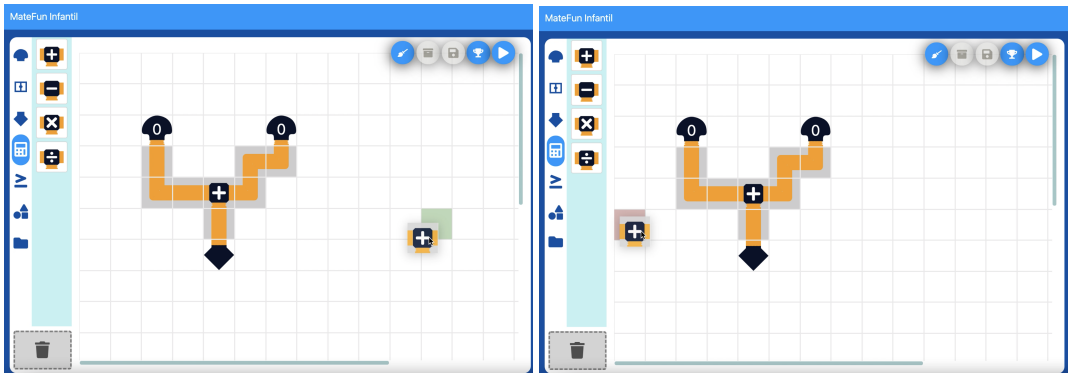
Otra funcionalidad que se decidió implementar para mejorar la usabilidad de la aplicación, fue la selección de varias tuberías mediante el arrastre del cursor, manteniendo el click izquierdo del mouse presionado. A través de esta acción se crea un recuadro punteado sobre el tablero, como el de la Figura 4.14a, que al soltar el click, deja las tuberías marcadas (Figura 4.14b), y permite moverlas de posición o eliminarlas. Previo a esta implementación, para mover o eliminar tuberías había que hacerlo una por una.



(a) Selección de tuberías (b) Tuberías seleccionadas

Figura 4.14: Proceso de selección de tuberías

Para llevar a cabo esta implementación se realizaron modificaciones en el código, sobre el armado del tablero incluyendo la biblioteca *react-selectable* [4] analizada en el estado del arte en la Sección 2. Se realizaron mínimos cambios a la hora de integrarla al proyecto con respecto a la detección de las acciones del mouse. Se modificó que al hacer click en una celda vacía y realizar un arrastre, en vez de arrastrar esa celda, se comienza a trazar el recuadro punteado para la selección.
En el componente *Board* de la interfaz, se incorporaron las funciones *selectableGroup* y *createSelectable*, siendo la primera utilizada para la creación del tablero, volviéndolo seleccionable, y la segunda para la creación de las celdas. El componente de las celdas del tablero no fue modificado, ya que se pasa como parámetro a la función *createSelectable* obteniendo así las seleccionables. Todas las celdas son seleccionables pero se incluyó una verificación para que aquellas que contienen tuberías sean las que efectivamente queden seleccionadas y así evitar celdas vacías señalizadas con el fondo gris, tal como se observa en la Figura 4.14b.
Con respecto al movimiento de múltiples tuberías, debido a que la biblioteca *react-dnd* [20], utilizada en el proyecto para la implementación del arrastre y soltado de las tuberías en el tablero, permite únicamente arrastrar de a un elemento, fue necesario agregar lógica que calcule la posición final de todas las tuberías seleccionadas que se pretenden mover y se muevan al soltar la tubería arrastrada. Para este cálculo se tienen en cuenta la posición de origen y destino de la tubería arrastrada.
Por otro lado, esta última biblioteca permite al momento de arrastrar una tubería, indicar si la celda donde se desea soltar es una posición válida o no. En caso que la posición sea válida se muestra en verde 4.15a y en caso contrario en color rojo 4.15b. Al soltar una tubería en una celda inválida, la tubería se mantiene en su posición de origen. Si hubiese más de una tubería seleccionada se tienen en cuenta todas las posiciones finales para indicar si es válido el movimiento o no.



(a) Posición válida para soltar tuberías (b) Posición inválida para soltar tuberías

Figura 4.15: Movimiento de múltiples tuberías

Por último, si se arrastra una de las tuberías seleccionadas hacia la papelera, se eliminan todas estas, lo que facilita la usabilidad al momento de querer borrar grandes conjuntos de tuberías del tablero.

4.6. Creación de función a partir de selección múltiple

En MateFun Infantil ya existía la posibilidad de guardar funciones utilizando todas las tuberías presentes en el tablero, para ello el tablero debía contener al menos una variable pero no más de tres, una salida y tuberías sin errores. Inspirado en Viskell, que permite crear bloques a partir de otros, y basándose en la funcionalidad descrita anteriormente, dado que la extensión del tablero se vio restringida hasta cierto límite, se decidió implementar la funcionalidad que permite crear una función a partir de la selección de múltiples tuberías. Una vez creada la función, es insertada automáticamente dentro del recuadro seleccionado, sustituyendo y optimizando la cantidad de tuberías de su interior. Con esta característica es posible simplificar el conjunto de tuberías del tablero, además de poder a futuro reutilizar la función creada en otras partes de este. Asimismo, crear funciones para tener el código modularizado, es una buena práctica de programación que pueden ir adquiriendo los niños.
Esta implementación implicó que se debiera corroborar y actualizar el estado de la matriz posterior a la creación de la función, de forma que esta última quede incluida en el tablero y conectada a las tuberías no seleccionadas. Otro aspecto relevante en la implementación fue la decisión que se tomó a la hora de crear la nueva función e integrarla al tablero. Al unirla al resto de las tuberías, solo se pueden modificar tuberías de la parte seleccionada del mismo. Esta decisión se tomó pensando en que el usuario podría querer mantener intacta la estructura que se encuentra por fuera de este.
Dicha creación se puede dar siempre y cuando las tuberías seleccionadas cumplan con el formato necesario para pertenecer a una función. Luego de seleccionadas las tuberías, se crea una nueva matriz con las mismas almacenándola en el componente **Compiler** presentado en el diagrama de las capas de la Figura 4.2, por lo que la relación **Compiler** - **Matrix** pasa ser de uno a dos en lugar de uno a uno.
La funcionalidad consta de cuatro partes:

- **Validación**: Se verifica que la matriz seleccionada sea una matriz válida para ser guardada como función.

- **Armado**: Se completa la matriz con variables y resultado, uniendo con variables las tuberías que representan entradas y a la salida con la tubería de resultado.
- **Guardado**: Se guarda la función utilizando lo implementado para el guardado de funciones de todo el tablero.
- **Inserción**: Se inserta la función creada en el tablero y se unen sus entradas y salida según corresponda.

4.6.1. Validación

En la Figura 4.16 se muestra el algoritmo que comprueba si la matriz seleccionada es válida. En caso de serlo, retorna una variable booleana en verdadero, un arreglo con las tuberías que van ser conectadas a variables y la tubería que va a ser conectada a una salida, de lo contrario retorna la variable booleana en falso.

```
isValid(Matrix){
    //cuenta la cantidad de pipe seleccionados que son distintos a dummy
    countNoDummy = 0

    //Arreglo con las entradas
    entries = []

    //Variable para guardar la salida
    out = null

    //Antes de la recorrida verifica tamaño mayor a 3
    if (Matrix.rows.size < 3 || Matrix.columns.size < 3){
        //Abortar ejecución y no lo seleccionado no se puede guardar como función
        return false
    }

    //Recorro las columnas y las filas de la matriz
    for i = 0 to Matrix.length {
        for j=0 to Matrix[i].length {
            if (Matrix[i][j] tiene pipe y no es dummy) {
                countNoDummy = countNoDummy + 1
            }
            if (Matrix[i][j] está en el limite de lo seleccionado, tiene una tubería no conectada hacia el exterior){
                if (Matrix[i][j] es una entrada){
                    entries.add(Matrix[i][j])
                    //Verifica que la función tenga hasta 3 entradas, requisisto para ser función válida
                    if (entries.length > 3) {
                        //Terminar ejecución y no lo seleccionado no se puede guardar como función
                        return false
                    }
                    if (out != null){
                        //Verifica que la entrada no este por debajo de la salida
                        if (Matrix[i][j].pos.row >  out.pos.row){
                            //Terminar ejecución y no lo seleccionado no se puede guardar como función
                            return false
                        }
                    }
                } else {
                    //Es salida
                    out = Matrix[i][j]
                }

            }
        }
    }
    //Luego de la recorrida se verifica los no Dummy Pipe
    if (countNoDummy < 2) {
        return false
    }
    return true, entries, out
}
```

Figura 4.16: Algoritmo que define si la matriz seleccionada es válida.

Se comprueba que dicha matriz cumpla con las características necesarias para ser función, validando que contenga al menos una tubería que represente una entrada que no esté conectada a otra seleccionada, y otra tubería que represente una salida. A su vez, son tres la cantidad máxima de entradas permitidas al igual que el guardado de función ya implementado. Las tuberías identificadas como entradas son conectadas a variables y la salida a una tubería resultado.
Además de las restricciones mencionadas anteriormente alineadas con las existentes para creación

de funciones con todo el tablero, se agregaron las siguientes que guían a un uso adecuado de la funcionalidad:

- No pueden existir tuberías de entrada por debajo de la salida. Esta restricción se puede observar en la Figura 4.17.

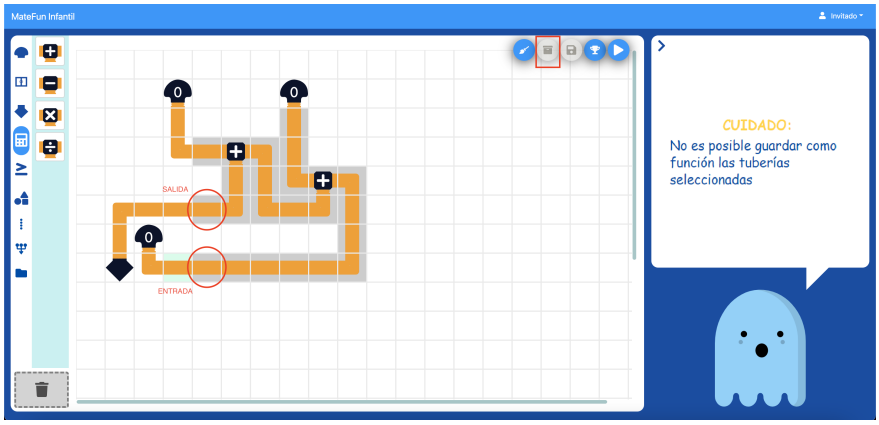


Figura 4.17: Selección inválida para ser guardada como función por existir una entrada por debajo de una salida.

El motivo por el cual no se permiten tuberías de entrada por debajo de la salida, es que en algún caso particular, debido al algoritmo elegido a partir de la decisión de modificar únicamente tuberías dentro del área seleccionada, podría generar un bloqueo entre las mismas al momento de la inserción de la función. Un ejemplo de esto es cuando se crea la función a partir de un conjunto de tuberías seleccionadas como en la Figura 4.18, en donde al insertar la función, se generaría un bloqueo entre tuberías, como se observa en la Figura 4.19

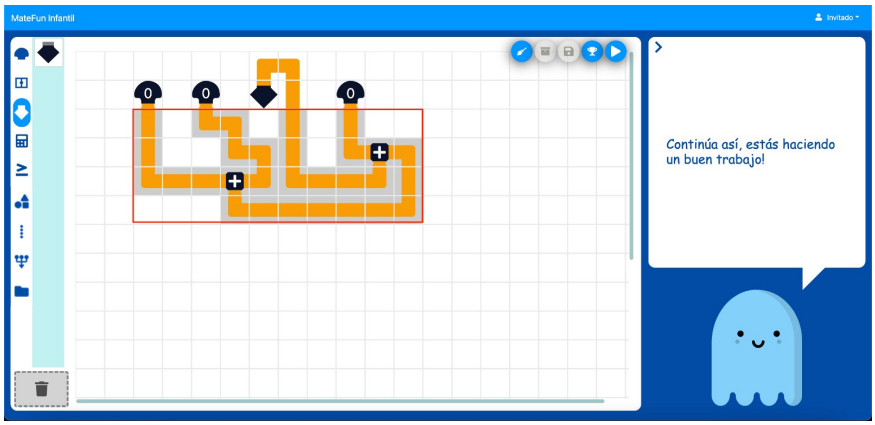


Figura 4.18: Tuberías seleccionadas inválidas para ser guardadas como función por tener una entrada por debajo de una salida.

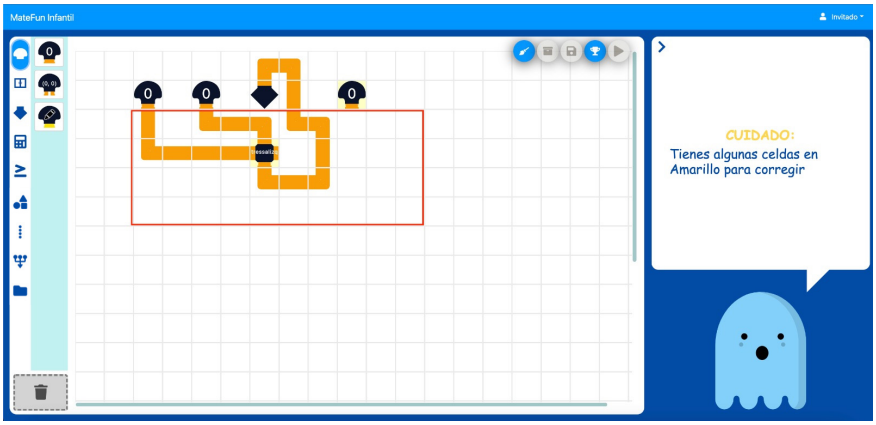


Figura 4.19: Bloqueo que se generaría al insertar la función.

- Las tuberías seleccionadas deben contener al menos una tubería de tipo función y otra tubería que no sea de tipo dummy. Esto se observa en la Figura 4.20 y refiere a la idea que no tiene sentido crear una función con una única operación, constante o variable, ya que se estaría creando algo que ya está definido en MateFun, y además no disminuiría la cantidad de tuberías en el tablero.

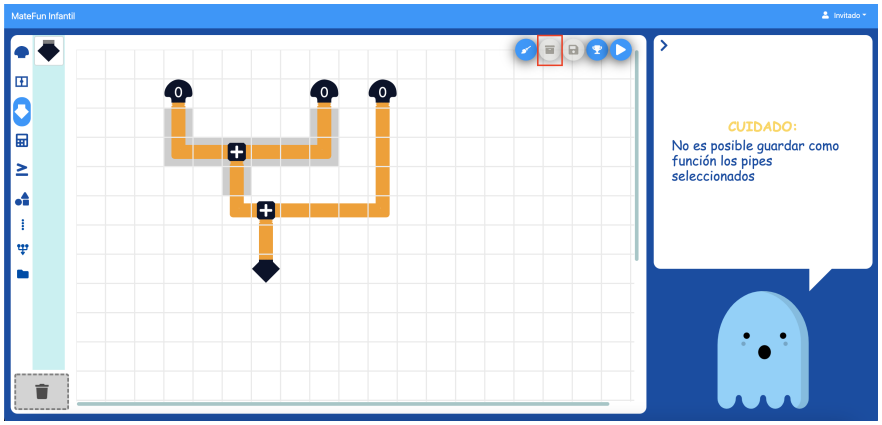


Figura 4.20: Tuberías seleccionadas inválidas para ser guardadas como función por tener una única distinta a una tubería de tipo dummy.

- El recuadro seleccionado para crear la función, debe ser de tamaño mínimo de tres celdas de largo y tres celdas de ancho. Esta decisión se tomó ya que uno de los objetivos, como se mencionó en el punto anterior, es disminuir la cantidad de tuberías en el tablero. Por tanto, crear una función a partir de una selección menor a esa, no tendría sentido.

En caso que no sea posible crear la función, se muestra un mensaje notificando que lo seleccionado no puede ser guardado.

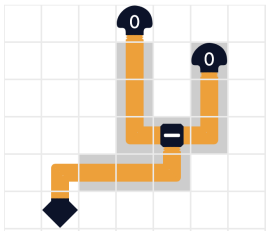
4.6.2. Armado

Luego, si la nueva matriz con las tuberías seleccionadas es válida, se conectan las tuberías desconectadas a variables o a la salida según el pseudocódigo de la Figura 4.21.

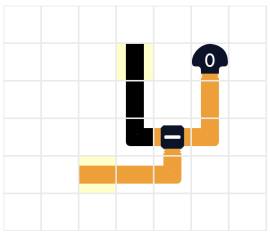
```
// Matrix: matriz con las tuberías seleccionadas
//initRow, finalRow, initColumn, finalColumn: filas y columnas que marcan el incio y el fin del area seleccionada
addEntriesAndOut (Matrix m, initRow, finalRow, initColumn, finalColumn) {
    //Se comienza a evaluar desde lateral izquierdo abajo hacia el derecho por lado superior
    for (var i = finalRow; i <= initRow; i--) {
        evaluatePipe(Matrix[i][initColumn])
    }
    //Se evalua el lado superior del recuadro seleccionado
    for (var i = initColumn; i <= finalColumn; i++) {
        evaluatePipe(Matrix[initRow][i])
    }
    //Se evalua el lado derecho del recuadro seleccionado
    for (var i = initRow; i <= finalRow; i++) {
        evaluatePipe(Matrix[i][finalColumn])
    }
}

evaluatePipe(cell) {
    Si cell contiene tubería y está desconectada:
        Si la tubería corresponde a una entreda:
            Se inserta una varaible generica y se une a la tubería desconectada
        Si la tubería corresponde a una salida:
            Se inserta una tubería resultado y se une a la tubería desconectada
}
```

Figura 4.21: Pseudocódigo de algoritmo de armado de la matriz a guardar

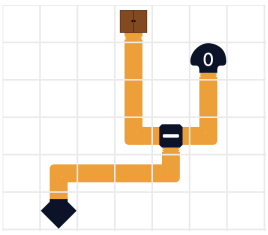
En la Figura 4.22 se puede ver como se genera una matriz con las tuberías seleccionadas y luego se conecta la variable y la salida.



(a) Tubería seleccionadas para convertir en función



(b) Nueva matriz que se guardará como función



(c) Nueva matriz guardada como función con la variable y salida correspondiente

Figura 4.22: Posibles casos con entrada y salida en la misma esquina inferior

4.6.3. Guardado

Después de haber sido validada y conectada con la salida y variables, se guarda la matriz utilizando la función del componente **MateFun API**, ya creada para el guardado de funciones.

4.6.4. Inserción

Por último, se inserta la función creada en el tablero conectando sus entradas y salida a las tuberías correspondientes.

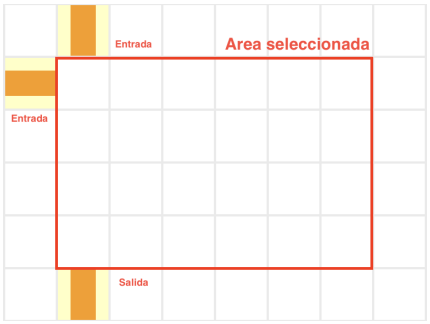
Para encontrar la posición óptima donde insertar la función se podría haber implementando un algoritmo de backtraking donde se probaran todas las celdas dentro del recuadro seleccionado hasta encontrar una donde se pueda conectar la función a todas las tuberías. Debido a que dicho algoritmo es muy costoso y el problema se puede acotar según casos bordes, se decidió implementar el algoritmo de la Figura 4.23 que retorna la posición donde debe ser insertada la función y si es posible unir todas las tuberías (entradas y salidas).

```
// leftEntries, topEntries y  rightEntries: son las celdas dentro del area seleccionada que se van
// a conectar a una tubería exterior y a las entradas de la función.
// out: celda que será conectada a la salida y de la función y limita con una tubería exterior
// funcPipe: tubería de la función creada
// initPos: posición inicial del area seleccionada (la tubería seleccionada más a la izquierda y más arriba)
// endPos: posición final del area seleccionada (la tubería seleccionada más a la derecha y más abajo)
positionInsertFunction (leftEntries, topEntries, rightEntries, out, funcPipe, initPos, endPos) {
    middleX =  initPos.x + ((endPos.x - initPos.x) / 2)
    middleY =  initPos.y + ((endPos.y - initPos.y) / 2)
    nextCells = []
    countEntries = funcPipe.entries.length
    Si existe leftEntries con posisición X igual a initPos.x y topEntries con posisición Y igual a initPos.y
        Si countEntries igual a 3 leftEntries.length igual 1:
            return initPos.x, initPos.y, true
        Sino:
            return middleX, middleY, false
    Si existe rightEntries con posisición X igual a initPos.x y topEntries con posisición Y igual a endPos.y:
        si countEntries igual a 3 y rightEntries.length igual 1:
            return initPos.x, endPos.y, true
        Sino:
            return middleX, middleY, false
    Si existe leftEntries con posisición X igual a endPos.x:
        Si out tiene posisición Y igual a initPos.y:
            Si countEntries mayor a 1:
                return endPos.x, initPos.y, true
            Sino:
                return middleX, middleY, false
        Sino:
            return endPos.x, out.y, true
    Si existe leftEntries con posisición X igual a endPos.x:
        Si out tiene posisición Y igual a endPos.y:
            Si countEntries mayor a 1:
                return endPos.x, endPos.y, true
            Sino:
                return middleX, middleY, false
        Sino:
            return endPos.x, out.y, true
    Si  leftEntries.length mayor igual a 2:
        x = pos x de la última tubería de leftEntries
        return x, middleY, true
    Si  rightEntries.length mayor igual a 2:
        x = pos x de la última tubería de rightEntries
        return x, middleY, true
    return middleX, middleY, true
}
```

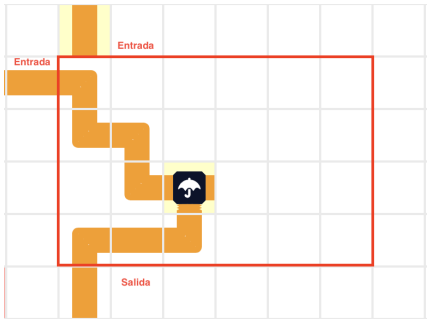
Figura 4.23: Pseudocódigo de algoritmo que encuentra la posición donde debe ser insertada la función.

En la figura anterior se distinguen los siguientes casos:

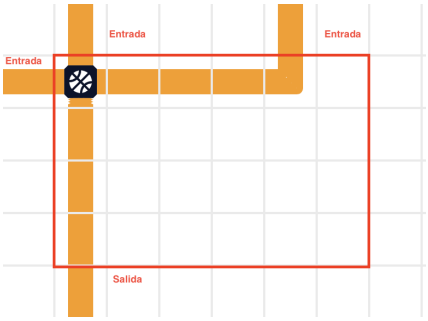
1. Las condiciones 1 y 2 corresponden a cuando la sección seleccionada limita con dos entradas en una esquina superior. Ingresa en la primera o segunda condición del código dependiendo si la esquina es la derecha o la izquierda. Solo es posible unir las tuberías luego de insertar la función en caso que exista una tercer entrada. El ejemplo de la Figura 4.24a cumple la primera condición con dos entradas, sin importar el lugar de inserción no es posible unir ambas entradas y se visualiza en la Figura 4.24b.



(a) Área seccionada con dos tuberías de entrada en una esquina superior.



(b) Inserción de función con dos entradas en área seleccionada con dos tuberías de entrada en una esquina superior.

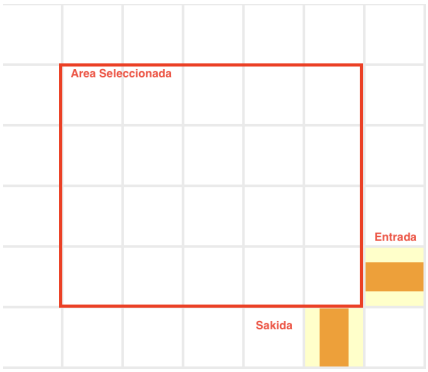


(c) Inserción de función con tres entradas en área seleccionada con dos tuberías de entrada en una esquina superior

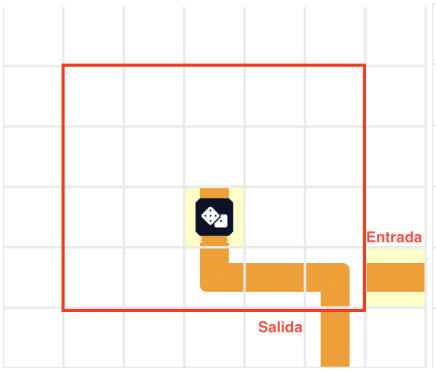
Figura 4.24: Caso con dos entradas en esquina superior

Sin embargo, como se muestra en la Figura 4.24c. al existir una tercer entrada como en la figura, sí se puede insertar y unir como lo indica el algoritmo.

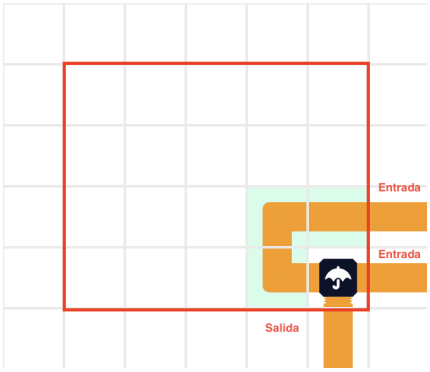
2. Si el área seleccionada limita con al menos una tubería de entrada en su última fila inferior, se cumple la condición 3 o 4 dependiendo el lado.
   Si además coincide que la tubería a unir con la salida se encuentra en la primera o última columna, solo es posible unir las entradas si la función tiene más de una como en la Figura 4.25. En otro caso la función será insertada en la misma columna que la salida.



(a) Área seleccionada



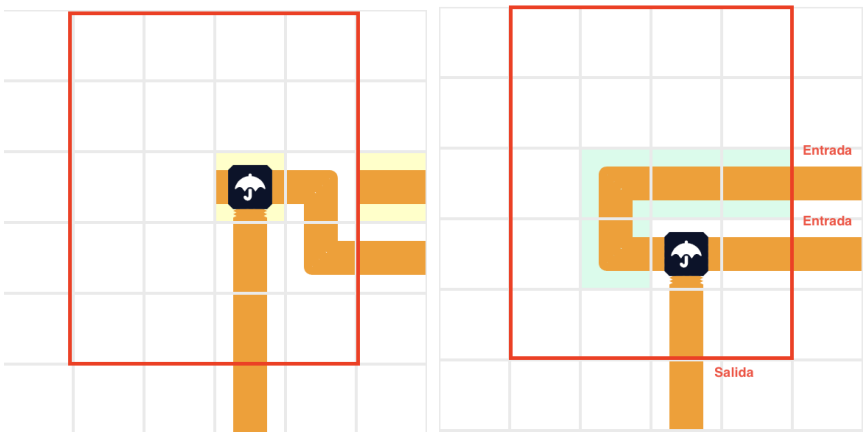
(b) Inserción una entrada



(c) Inserción dos entradas

Figura 4.25: Posibles casos con entrada y salida en la misma esquina inferior

3. Si en el recuadro seleccionado, en uno de sus laterales tiene dos o tres entradas y no cumple con ninguna de las condiciones anteriores, entra en la condición 5 o 6. La función se crea insertándose en la misma fila que la entrada inferior de dicho lateral y en la columna que se encuentre en la mitad de lo seleccionado. Si esto no sucediera, se podrían generar bloqueos en las uniones de las tuberías. En la Figura 4.26 se muestran ambos casos.



(a) Función insertada en el medio del área seleccionada

(b) Función insertada según el algoritmo

Figura 4.26: Más de una entrada por el lateral

4. En caso que no cumpla ninguna de las condiciones anteriores, se selecciona la celda que se encuentra en el centro del recuadro seleccionado, allí se inserta la función y se une con las entradas del mismo.

4.6.5. Ejemplo

En caso que las tuberías seleccionadas puedan ser reducidas a una función, se habilita un botón en la esquina superior derecha del tablero, como en la Figura 4.27:

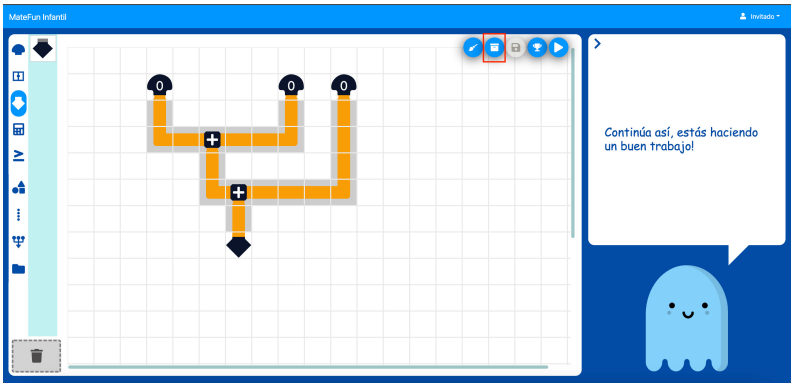


Figura 4.27: Tuberías seleccionadas válidas para ser guardadas como función.

Al presionarlo despliega un modal idéntico al utilizado en el guardado de función ya implementado en MateFun, como en la siguiente Figura 4.28.

Figura 4.28: Modal para el guardado de función.

Este permite elegir un nombre y un ícono, y al guardar, integra la función dentro del tablero, como en la Figura 4.29.

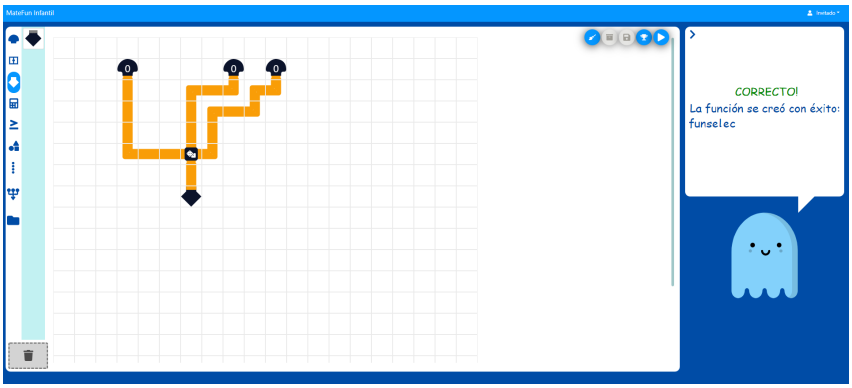


Figura 4.29: Función creada a partir de selección múltiple.

4.7. Multitubería

La implementación de esta funcionalidad en primer lugar comprende la definición de tres nuevas funciones, una de ellas para crear una tubería múltiple, es decir concatenar dos tuberías en una y las funciones de desarmado de tubería múltiple por derecha e izquierda. Para esto se incluyeron las nuevas funciones en el componente **Pipe** más específicamente dentro el conjunto agrupado bajo **FuncPipe**. Además, fue necesario crear un nuevo tipo de tubería denominado tubería múltiple que resuelve lógicamente el concepto de concatenar más de una tubería de igual o distinto tipo.

Detalle de nuevas funciones implementadas:

- Crear tubería múltiple:

  Recibe dos entradas de tipo genérico y retorna una tubería múltiple.

- Desarmar tubería múltiple por derecha:

Recibe una entrada de tipo tubería múltiple y retorna una tubería múltiple o una tubería de tipo genérico.

- Desarmar tubería múltiple por izquierda:

  Recibe una entrada de tipo tubería múltiple y retorna una tubería múltiple o una tubería de tipo genérico.

La motivación de esta funcionalidad inicialmente parte de la necesidad de generar un par de números sin tener que hacerlo mediante la tubería "Par de números", para que de esta forma estos puedan ser obtenidos como resultados de operaciones. Posteriormente, junto a los tutores se decidió extender esta noción, generalizando el concepto de par, y así poder anidar cualquier tipo de tubería dentro de una de tipo multitubería. Junto a ellos se decidió que el resultado que se obtiene al crear una multitubería siempre debe ser un par de elementos. En donde cualquiera de estos dos elementos puede ser un valor único u otro par de elementos. En la Figura 4.30 se presenta el caso que motivó la definición e inclusión de esta funcionalidad en el sistema. Esta funcionalidad se incluyó dentro de los niveles de dificultad "avanzado" y "completo" de la aplicación.

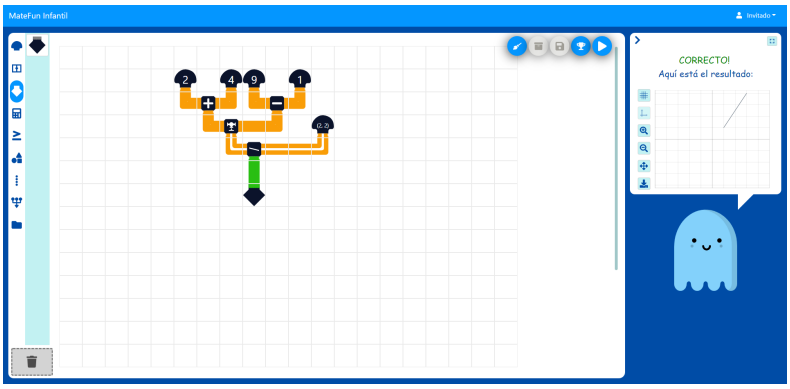


Figura 4.30: Multitubería motivación.

El desarrollo comprendió dos etapas principales, la primera de ellas fue resolver gráficamente la noción de tubería múltiple. Dado que inicialmente esta extensión no estaba prevista en los componentes originales de la solución, se requirieron ajustes sobre todos los componentes gráficos que comprenden cada tipo de tubería existente en el tablero. Como se describe en la Figura 4.31, una multitubería es capaz de representar gráficamente dos tipos de datos indicando su color identificativo y en caso de contener tres o más tuberías, el lado del par que está compuesto por dos o más tuberías se lo identifica con el color violeta. El resultado esperado según la definición antes mencionada de construcción de la multitubería en dicho ejemplo es: [[[[1,Color],Figura],(0,0)]]. Esto se debe a que en primer lugar se creó la multitubería con la constante numérica 1 en la izquierda y la constante de tipo color en la derecha. Esto genera el primer par de elementos [1,Color]. Posteriormente se crea una nueva multitubería donde este par de elementos ingresa por izquierda y el círculo de radio 2 por la derecha, obteniendo el resultado [[1,Color],Figura]. Por último se crea la multitubería con el par de números (0,0) por la derecha y el par de elementos creado anteriormente por la izquierda, llegando así al resultado final [[[[1,Color],Figura],(0,0)]].

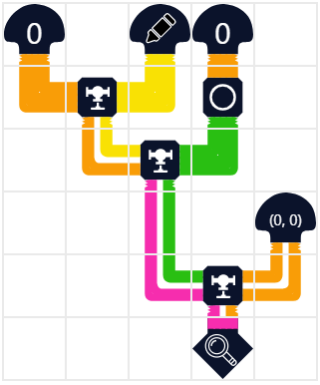
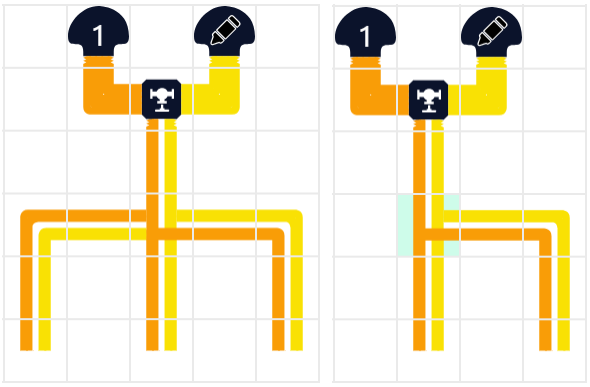


Figura 4.31: Ejemplo de Multitubería.

Como se detalló en la sección de validación de esta funcionalidad, fue importante decidir que forma resolver casos como cruces de tuberías en formas de “T” o “+”. En las figuras 4.32a y 4.32b se presenta la solución gráfica a este tipo de cruzamiento de tuberías.



(a) Multitubería +. (b) Multitubería T.

Figura 4.32: Cruces de multituberías

La segunda etapa consistió en desarrollar una estructura y funciones lógicas que permitan crear, mantener y seleccionar la multitubería durante la ejecución. Para esto se agregó tanto a los componentes **funcPipe** como **dummyPipe** un atributo de tipo arreglo de nombre *multipipeConstruction* que mantiene la información asociada a la tubería creada cuando esta es de tipo tubería múltiple. Este nuevo atributo es utilizado en la implementación de las funciones de desarmado y a su vez determina la representación gráfica de la tubería múltiple en cuanto a composición colores y ubicación.

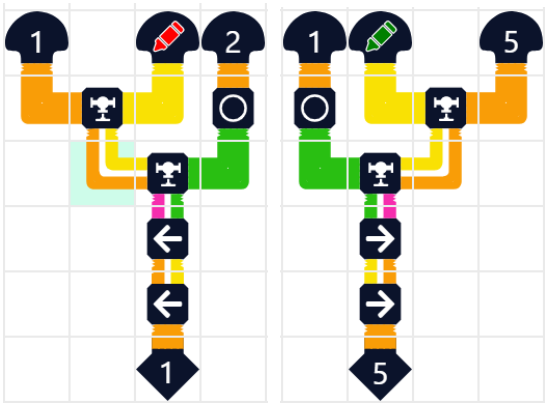
A continuación se detallan las funciones más destacadas que fueron implementadas para resolver la creación y desarmado de tuberías múltiples:

- **createMultipipeInputLeft** La función retorna el contenido asociado a la entrada izquierda de la tubería, es utilizado para construir o actualizar el contenido del la tubería en el caso de ser múltiple.

- **createMultipipeInputRight** Análogamente la función retorna el contenido asociado a la entrada derecha de la tubería, es utilizado para construir o actualizar el contenido de la tubería en el caso de ser múltiple.
- **createMultipipeOutput** La función determina la salida de la tubería dependiendo de la función, es decir que en el caso de la función "Crear Tubería Múltiple" la salida a partir de las entradas derecha e izquierda de la misma. También contempla las funciones de selección por derecha e izquierda, para estos casos configura la salida de la función con el tipo correspondiente con el "lado" de la tubería múltiple seleccionado.
- **createMultipipeDisarmInput** La función actualiza los tipos de datos contenidos en la tubería de tipo múltiple que ingresa como entrada a las funciones de desarmado derecha e izquierda.

Para esta implementación se crearon estructuras para mantener y actualizar dinámicamente dicho atributo para todas las tuberías pertenecientes a la multitubería creada. Esto implicó varias modificaciones debido a que las recorridas recursivas para chequeo y actualización previamente implementadas en la solución original del sistema no preveían mantener más de un tipo de datos por tubería, ni realizar cambios dinámicos, como por ejemplo al modificar un tipo en una entrada en la creación de la tubería múltiple.

A continuación, en las Figuras 4.33a y 4.33b, se muestra un ejemplo de uso de las funciones de desarmado por izquierda y derecha respectivamente.



(a) Desarmado por izquierda. (b) Desarmado por derecha.

Figura 4.33: Multituberías con desarmado por izquierda y derecha

4.8. Gestor de desafíos

En primer lugar, para acceder al gestor de desafíos se incluyó un nuevo botón en la parte superior de la pantalla con un trofeo dentro como se muestra en la siguiente Figura 4.34, tal cual fue validado con los docentes.

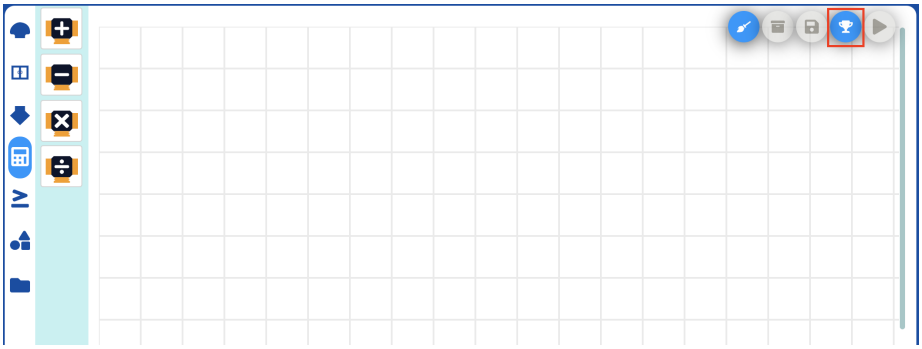


Figura 4.34: Acceder a desafíos

Al presionar este botón se despliega un modal que permite seleccionar en primera instancia el nivel del desafío, luego el tipo de desafío y por último el nombre del desafío que desea, como muestra la Figura 4.35.

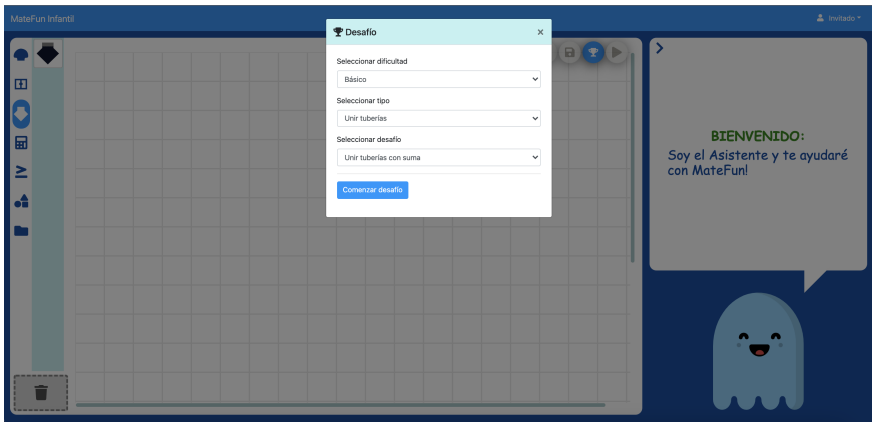


Figura 4.35: Modal para la selección de desafío

El seleccionable con los nombres de desafíos es visible solamente en caso que existan desafíos cargados previamente para el nivel y el tipo seleccionados, tal como se observa a continuación en la Figura 4.36.

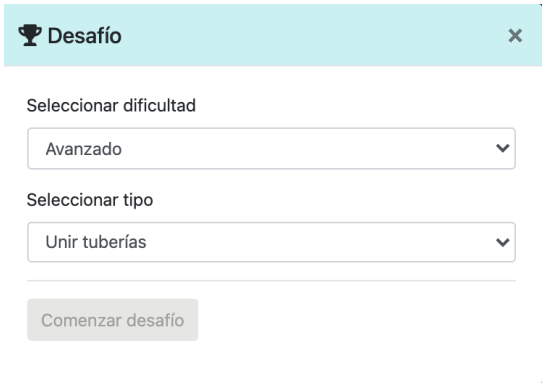


Figura 4.36: No existen desafíos para el nivel y tipo seleccionados

El nivel del desafío puede ser principiante, intermedio, avanzado y para cada uno de ellos, los tipos definidos son:

- **Unir tuberías**. Consiste en unir las operaciones y constantes pre cargadas en el tablero, para obtener el resultado solicitado por el desafío.
- **Completar con funciones**. En el tablero se encuentran pre cargadas algunas constantes y funciones y es necesario completar con otras funciones para llegar al resultado esperado.
- **Editar valor de constantes**. Editar los valores precargados para obtener el resultado deseado.

Los desafíos se cargan previamente en archivos JSON dentro del directorio /constant/challenges divididos por niveles principiante, intermedio y avanzado y tienen el formato mostrado en la Figura 4.37. En este caso el desafío cuenta con tres variables de tipo entero que deben ser asignadas y conectadas a las funciones de suma provistas en el tablero. Se debe obtener el resultado entero “3” para completar el desafío correctamente.

```
export const challenges = {
    name: "Unir tuberías con suma",
    pipes: [
        {type:"VARIABLE",dir:{bottom:"NUMBER"},name:"x0",pos:{x:0,y:0}},
        {type:"VARIABLE",dir:{bottom:"NUMBER"},name:"x1",pos:{x:0,y:1}},
        {type:"VARIABLE",dir:{bottom:"NUMBER"},name:"x2",pos:{x:0,y:2}},
        {type:"FUNCTION",dir:{bottom:"NUMBER",right:"NUMBER",left:"NUMBER"},name:"ADD",pos:{x:4,y:6}},
        {type:"FUNCTION",dir:{bottom:"NUMBER",right:"NUMBER",left:"NUMBER"},name:"ADD",pos:{x:6,y:4}},
        {type:"END",dir:{top:"GENERIC"},pos:{x:8,y:4}}
    ],
    size:{x:20,y:20},
    result: "3",
    description: "Une las tuberías con sumas y asígnale un valor a las variables para llegar al resultado 3",
    validations:[
        {
            count: 2,
            type: 'FUNCTION',
            name: 'ADD',
            error: 'Deberías usar 2 operaciones suma'
        }
    ],
    tip: "Si le asignas un valor adecuado a cada variable no es necesario que se agreguen nuevas sumas.",
    type: 0
}
```

Figura 4.37: Formato de los desafíos cargados

Allí se encuentran los siguientes atributos:

- **type**: Se determina el tipo de desafío de los definidos teniendo en cuenta el siguiente mapeo:

Unir tuberías: Type = 0

Completar con funciones: Type = 1

Editar Valor de constantes: Type = 2

- **name**: Nombre del desafío.
- **pipes**: Arreglo donde se definen las tuberías precargadas que tendrá el tablero al comenzar el desafío. Para cada tubería se deben especificar los siguientes campos:

  type: Tipo de tubería, (CONST, VARIABLE, FUNCTION, END)

  value: Se asigna el valor de la tubería en caso de ser de tipo CONST o VARIABLE.

  dir: Se definen los tipos de entrada y salida de la tubería para cada dirección. Se utiliza el siguiente formato: dir: { bottom:“FIGURE”,right:“NUMBER”,left:“NUMBER” }.

  name: Nombre asingado, el caso de las funciones se debe ingresar la funcion deseada, para el resto de los tipos no es necesario.

  pos: Se determina la posición en el tablero. Ejemplo: pos: {x:6,y:4}.
- **size**: Tamaño del tablero del desafío.
- **result**: Resultado esperado del desafío.
- **description**: Consigna del desafío, lo que se solicita realizar.
- **tip**: Pista que se puede solicitar durante la realización de un desafío. Por el momento se estableció tener una única pista por desafío. Esto puede ser extendido modificando el formato de los archivos JSON para soportar una lista de pistas, además de modificar la implementación del mostrado de pistas a nivel del componente **Actions**.
- **validations**: Son validaciones que se pueden realizar sobre el tablero para evaluar si la solución creada para resolver el desafío ha sido la más óptima según quién creó el desafío. La forma de cuantificarlo es definiendo la cantidad de tuberías de cierto tipo que sean necesarias para resolver el desafío.

  count: Cantidad óptima de tuberías para el desafío.

  type: Tipo de tubería a cuantificar, generalmente de tipo “FUNCTION”.

  name: Nombre de la tubería a cuantificar, en el caso de ser de tipo “FUNCTION”, por ejemplo: “ADD”, “SUB”, “MUL”, “DIV”.

  error: Mensaje de error que se muestra cuando la validación no fue superada para la resolución planteada.

La implementación del gestor de desafíos ha sido incluida dentro el componente **Actions**, la misma compone la interacción con los botones descriptos anteriormente, el modal de selección de desafío así como también el nuevo estado del sistema; *isChallengeMode* que indica que hay un desafío en ejecución, estado que se actualiza dentro de **Redux Store**. Si bien el inicio y elección de un desafío se resuelve en los componentes mencionados anteriormente, se destaca que tanto la mensajería asociada al desafío, como letra, ayuda, alertas y resultados se despliega mediante el asistente en la sección de comunicación y resultados. Por lo tanto el desarrollo también contempló modificaciones sobre los componentes **Main** y **DisplayResult**.

Con un desafío seleccionado, presionando el botón “Comenzar desafío”, este comienza cargando el tablero precargado en el campo “pipe”mencionado anteriormente. Como se puede ver en la Figura 4.38, la consigna del desafío es presentada por el asistente y en la esquina superior derecha se muestran cuatro botones. En orden de izquierda a derecha, el primer botón permite cancelar el desafío, volviendo al tablero en el estado que se encontraba antes de empezar el mismo. Los dos siguientes son para volver a mostrar la letra del desafío, y el botón para solicitar la pista. Por último, se encuentra el botón para ejecutar el conjunto de tuberías que se completó en el desafío.

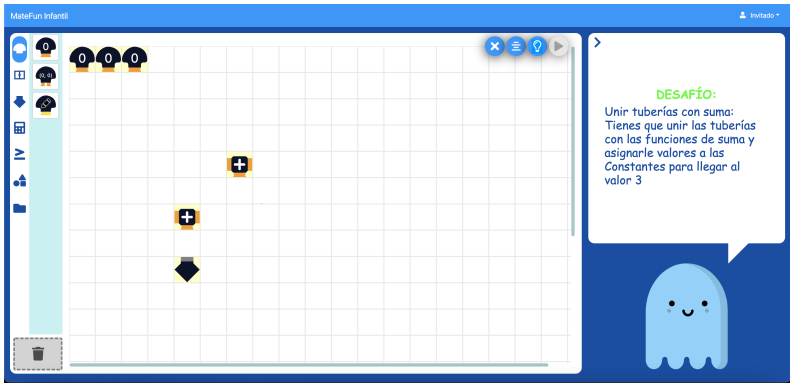
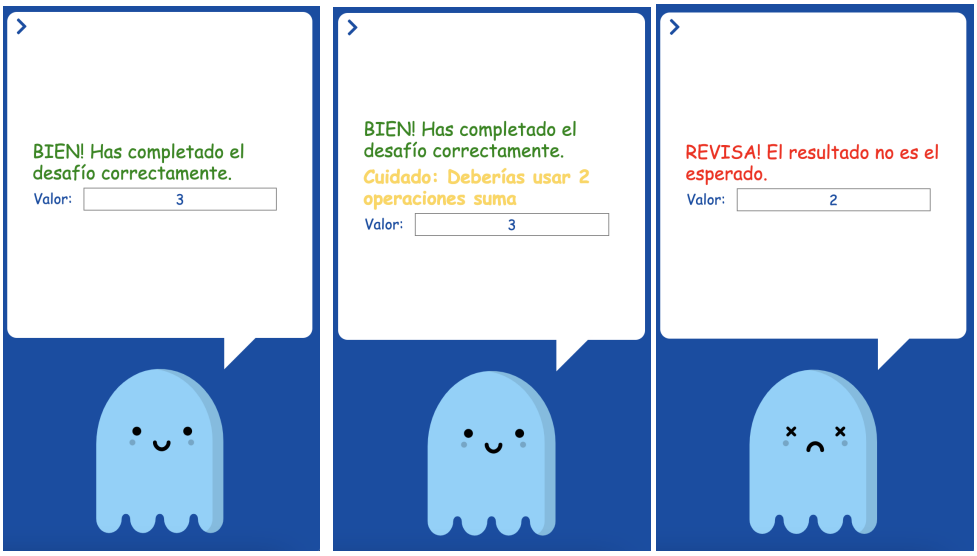


Figura 4.38: Ejemplo de desafío del tipo Unir tuberías

Al presionar el botón de ejecutar, se compara el resultado obtenido con el esperado y el asistente despliega un mensaje en consecuencia. En caso que se haya obtenido el mismo resultado cumpliendo con las validaciones definidas, se muestra un mensaje de éxito como en la Figura 4.39a. Si el resultado obtenido es el correcto pero no cumple con los atributos de calidad, se muestra un mensaje de éxito advirtiendo que la solución podría mejorar, tal como se observa en la Figura 4.39b. Por ultimo en caso de error en el resultado, se indica que el desafío no ha sido completado correctamente, como en la Figura 4.39c.



(a) Mensaje resultado correcto

(b) Mensaje resultado correcto con advertencia

(c) Mensaje resultado obtenido erróneo

Figura 4.39: Mensajes luego de ejecución del desafío

Capítulo 5

Validación final

Posterior a la implementación, de mutuo acuerdo con los docentes, se ejecutó una ronda de pruebas de sistema de acuerdo a los casos de uso definidos en el Anexo A 7. La misma permitió determinar ajustes y mejoras en las nuevas funcionalidades. Posteriormente se realizó una nueva instancia de validación tanto con docentes como con niños. En esta oportunidad, el objetivo era corroborar si las funcionalidades implementadas que se validaron previamente cumplían con lo esperado, y revisar si aquellas que no fueron validadas inicialmente -ya que su forma de implementación estaba bien definida- poseían una usabilidad adecuada. Además de esto, se validó con los docentes toda la terminología.

5.1. Pruebas de sistema

Una vez definidos los casos de uso asociados a las funcionalidades que fueron incluidas en la etapa de implementación se realizaron las pruebas de sistema. En el Anexo B 8 se detallan las pruebas junto con los resultados obtenidos

Durante esta etapa del proyecto se siguió el enfoque de testing exploratorio [12] basado en los quince casos de pruebas definidos. Las sesiones de trabajo permitieron identificar diferentes flujos que fueron incorporados en cada uno de los casos de prueba.

5.2. Validación con docentes

Para realizar esta tarea se tomó al mismo grupo de tres docentes y se realizó una presentación por separado con cada uno, abarcando todas las funcionalidades implementadas.
En primer lugar se mostró el asistente con todos los mensajes que arroja en las diferentes situaciones. Esta funcionalidad fue la más valorada por los docentes, ya que según ellos, es un elemento que motiva al niño para el uso de la aplicación, así como también lo auxilia y le brinda herramientas a la hora de realizar las propuestas.
Luego se presentaron las características de selección múltiple y extensión del tablero. Acerca de esto, no hubo grandes comentarios, más allá que opinaron que colabora al uso de la aplicación en general y que los niños van a entender sin problemas como utilizarlo.

Posteriormente se mostró la creación de funciones a partir de la selección múltiple, sobre esto indicaron que está clara la forma de utilizarla, pero es una funcionalidad que implica un mayor nivel de complejidad y abstracción para un niño. Podrían utilizarlo niños de edades avanzadas, en primaria probablemente de sexto grado.
Se continuó validando el gestor de desafíos, en donde se detalló la forma de elegir el nivel, tipo y desafío, ya que esto no existía en los prototipos iniciales. Lo más destacado por los docentes es la forma en que el asistente da aviso cuando el niño acierta o falla en el resultado. Del mismo modo elegir diferentes tipos de desafíos es un buen aporte. Una posible mejora que plantearon, es que los docentes puedan cargar los desafíos para así abordar las temáticas del curso que consideren necesario.
Por último se validó la multitubería. La misma fue implementada tal cual se mostró en la primera instancia de validación con prototipos, optando por la opción de desarmado por izquierda y derecha. Sobre este aspecto entendieron su funcionamiento y consideran que los niños van a comprender como se utiliza, siempre y cuando se les explique anteriormente el rol que tiene el color violeta (color de tipo multitubería). Si bien les cuesta un poco encontrarle algún tipo de utilidad a esta herramienta, opinan que es positiva ya que los niños al construirla y desarmarla en varias etapas de izquierda y derecha, aplican razonamiento lógico. Después de validar las funcionalidades, se pasó a una segunda etapa donde se realizó un cubrimiento de toda la terminología de MateFun para corregir los conceptos o frases que los docentes consideraran necesario.
A continuación se presenta la Tabla 5.1 con las correcciones que plantearon:

Termino Actual	Definición	Sugerencia	Justificación
Constantes	Menú donde se incluyen el conjunto de tuberías de tipo constante (Número, Par de número, Color)	Valores	El término constante es demasiado técnico para un niño
Matemática	Menú donde se incluyen el conjunto de tuberías para realizar operaciones aritméticas (suma, resta, multiplicación y división)	Operaciones	El término matemática es mucho más abarcativo
Condiciones y Comparadores	Menú donde se encuentran tuberías para comparar números, figuras, etc. También contiene tuberías para aplicar condiciones y según su veracidad retornar uno u otro valor	Condicionantes y comparadores	Corrección sintáctica
Unir tuberías	Tubería que toma como entrada dos valores que podrían ser simples o de tipo multitubería y los une en una nueva mutltitubería que contiene a ambos	Crear multitubería	Termino acorde al menú en el que se encuentra.
Guardar	Botón que permite guardar un conjunto de tuberías como una función creada por el usuario	Guardar función	Indica con mayor especificidad lo que se está guardando
Agregar funciones	Tipo de desafío que tiene los valores numéricos ingresados y faltan las tuberías de operación	Agregar operaciones	Especifica correctamente qué es lo que se debe agregar.
Completar con constantes	Tipo de desafío que tiene todo el sistema de tuberías armado y solo resta editar las variables	Completar variables	Indica adecuadamente la acción que debe realizar el niño.
Cancelar	Botón para salir del desafío	Salir del desafío	Termino que se adecúa a la acción. El ícono podría ser una puerta indicando salida.
Letra	Botón para mostrar la letra del desafío	Consigna	Término que se utiliza en la escuela para explicar una situación problema.
Unir tuberías con suma: Tienes que unir las tuberías con las funciones de suma y asignarle valores a las variables para llegar al valor 3	Título y texto que muestra el asistente en uno de los desafíos cargados	Une las tuberías con sumas y asígnale un valor a las variables para llegar al resultado 3	Mejoras en la sintaxis y quitar el título. Dejar solo la consigna.

Tabla 5.1: Validación final de terminología con docentes

Las correcciones de la tabla anterior, se tuvieron en cuenta y se modificaron en MateFun.

Con el resto de los términos se confirmó su correctitud, aunque hubo algunos que tuvieron opiniones encontradas:

- Variable: Este término una docente planteó modificarlo por interrogante o incógnita, pero las demás plantearon que por la edad para la cual apunta la aplicación, se puede mantener este término.
- Multitubería: Una docente planteó nombrarla como Tubería múltiple, pero las demás prefirieron el término actual, ya que apunta más a lo lúdico y es más atractivo.
- Par de números: Plantearon que una opción podía ser el término coordenada, aunque terminaron inclinándose por el término actual ya que es más abarcativo.

Otro aspecto que se decidió validar con los docentes, fueron los textos que despliega el asistente ante las diferentes situaciones.
A continuación se muestra la Tabla 5.2 con las correcciones que plantearon los docentes:

Termino Actual	Situación	Sugerencia	Justificación
No está conectada a ninguna otra tubería	Cuando falta conectar una tubería de constante o variable	Falta conectar una tubería	Error sintáctico. Además es mejor evitar poner “No”.
No está conectada a otra tubería por la izquierda	Cuando una tubería de función no tiene conectado tubería en la entrada izquierda	Falta conectar una tubería por la izquierda	Error sintáctico. Además es mejor evitar poner “No”.
Los tipos de las tuberías que se están uniendo no concuerdan	Cuando se unen dos tuberías que los tipos no coinciden	Los tipos de tuberías no concuerden	Texto más corto y claro.
Hay más de una tubería de resultado	En el tablero se colocaron dos o más tubería de tipo resultado	Hay más de una tubería resultado	Corrección sintáctica. A su vez la palabra “resultado”podría estar en negrita, mayúscula o cursiva para destacarla.
Tienes algunas celdas en rojo para corregir	Cuando hay más de un error en el tablero, muestra este mensaje	Tienes celdas en rojo para revisar	Es un texto más breve y revisar es un término más sutil

Tabla 5.2: Validación de textos del asistente

Los docentes consideraron que el resto de los textos que muestra el asistente son adecuados, por tanto se corrigieron únicamente los antes mencionados.

5.3. Validación con grupo de trabajo interdisciplinario

Sobre el final del proyecto, los tutores acercaron la posibilidad de realizar una presentación a un equipo de trabajo interdisciplinario del cual participan. Este grupo ya tenía conocimiento previo sobre MateFun, lo cual era de utilidad para validar las nuevas funcionalidades implementadas.

Luego de una breve introducción a MateFun, a modo de ayuda memoria, mostrando la ejecución de algunas operaciones y figuras, se presentaron las funcionalidades implementadas en el siguiente orden:

- Asistente; mostrando los diferentes casos de interacción.
- Selección múltiple.
- Guardado de función a partir de la selección anterior.
- Gestor de desafíos.
- Multitubería, con su armado y desarmado parcial (esto último no estaba aún implementado por completo).
- Extensión de la grilla.

El grupo de trabajo mostró conformidad acerca de las nuevas características que se implementaron. Los comentarios fueron positivos, en donde se destacó la interacción que brinda el asistente, y el uso del gestor de desafíos para la resolución de ejercicios creados previamente. Surgieron algunas dudas acerca de la utilidad de la multitubería, pero una vez explicado, comprendieron los motivos de su implementación.

Durante el transcurso de la presentación, y principalmente sobre el final de la misma, surgieron varios comentarios, a modo de futuras implementaciones, los cuales se mencionan a continuación:

- Permitir que el uso de la creación de función a partir de selección múltiple esté habilitado únicamente en el nivel avanzado o completo, ya que es más complejo el razonamiento que hay que aplicar para entender su uso.
- Del mismo modo, permitir el uso de la multitubería en nivel avanzado o completo. Se siguió la recomendación de este punto y el anterior, modificando la implementación.
- Sobre el gestor de desafíos, se podría implementar un plan de ejercicios donde el niño vaya superando diferentes etapas, de menor a mayor complejidad.
- El asistente podría tener sonidos para las diferentes situaciones.
- Evaluar si no conviene dejar un poco de lado la idea que todo en MateFun se haga a partir de la ejecución de una función, y permitir que gráficamente el niño pueda realizar algunas creaciones, como por ejemplo dibujar un segmento o un punto (par de números).
- Permitir cambiar el tema de fondo, con la intención de ser más inclusivos y también dejar que el niño elija el color de su agrado.
- Crear tutoriales en video con todos los usos que se le puede dar a MateFun.
- Una vez subido el proyecto con las nuevas funcionalidades, abrirlo a la comunidad para que lo pruebe, busque errores y realice recomendaciones.

5.4. Validación con niños

Para esta validación final, nuevamente se tomó a los dos mismos niños de 8 y 11 años, dado que ya conocían la aplicación y de este modo se podía evaluar de mejor forma como interactuaban con las nuevas funcionalidades.
En primer lugar, para refrescar un poco su memoria, se les pidió que hicieran algunas operaciones aritméticas sencillas, y mientras esto sucedía, que observaran al asistente. A medida que el asistente desplegaba diferentes mensajes, se les consultaba a qué pensaban que refería este, obteniendo respuestas acertadas. Luego se los guió para que colocando tuberías llegaran tanto a mensajes de advertencia de desconexión de tuberías, como mensajes de error en unión de tipos, y mensajes de acierto en conjunto de tuberías bien armadas y ejecución de resultados. En todos los casos lograron interpretar lo que quería decir el asistente.
Posteriormente se les pidió que eliminaran tuberías del tablero, y al observar que lo hacían una a una, se les mostró la nueva funcionalidad de seleccionar varias tuberías mediante un recuadro tanto para eliminar como para mover a otra parte del tablero. Al principio tuvieron algunos fallos al intentar arrastrar, ya que seleccionaban en partes del recuadro donde no había tuberías. Al entender que debían arrastrar seleccionando alguna tubería de las marcadas en gris, no volvieron a repetir este error. Otro detalle que les resultó extraño al principio, fue que al arrastrar las tuberías seleccionadas, antes de soltarlas, solo se muestra la tubería en la que se hizo click al comenzar el arrastre, y no todo el recuadro moviéndose. Sin embargo al probar algunas veces, lo pudieron usar sin inconvenientes.
Seguido de esto, se les pidió que coloquen tuberías en las últimas tres columnas del tablero, reiteradas veces. Durante el transcurso observaron que la grilla se iba extendiendo hasta el momento en que el asistente indicó que ya no se podía extender más.
Luego se les presentó el gestor de desafíos. Primero se les solicitó que piensen cual era el botón que los llevaría a esto, y al ver un trofeo, fue la primera opción en ambos casos. Después se les mostró las opciones para elegir desafíos, y se les pidió que eligieran uno para resolver. La niña eligió el desafío de realizar una circunferencia de diámetro 8, y no tuvo problemas para hacerlo. El niño sin embargo, eligió un desafío aritmético que implicaba conectar tuberías de suma y modificar variables para llegar al resultado tres. En este caso le dificultó la realización del desafío, ya que no lograba visualizar que la salida de una tubería de suma, debía conectarse a la entrada de la otra. Esto muy probablemente se debió a que el niño era un año más pequeño que el público objetivo al que apunta MateFun Infantil. A ambos también se les pidió que colocaran un resultado erróneo en el desafío para que observaran la reacción del asistente. En este caso también entendieron perfectamente las indicaciones de este. Asimismo se les solicitó que identifiquen donde podrían ver la consigna del desafío nuevamente, o una pista en caso que la necesiten, y ambos botones fueron identificados sin problemas.
Después se validó el uso de la creación de funciones a partir de la selección múltiple. Para este caso se optó por hacerlo solo con la niña más grande, ya que implica un grado mayor de abstracción, que por las dificultades anteriores que tuvo el niño pequeño, no iba a poder lograr este ejercicio. A la niña se le mostró en primer lugar, como a partir de un conjunto de tuberías que formaban una operación aritmética, al realizar una selección múltiple que las contuviera, podía crear una nueva tubería que por dentro realizara todas esas operaciones. No tuvo ninguna duda acerca de la explicación. Para comprobar que entendió el funcionamiento, se le pidió que creara una nueva tubería que sumara dos números y al resultado lo multiplicara por un tercero. Lo realizó sin ningún inconveniente.
Por último, se validó la multitubería. Para ello se ejecutó el mismo conjunto de pruebas que en la validación inicial se habían presentado como prototipos gráficos. Como era de esperarse, ya que su implementación fue basada en la validación inicial que se tuvo con ellos, los niños comprendieron la mecánica tanto de la función de armado de la multitubería, como la de desarmado por izquierda y

por derecha.

Capítulo 6

Conclusiones y trabajo a futuro

6.1. Conclusiones

En primer lugar, a modo de conclusión general, se puede afirmar que se logró el resultado esperado, el cual era desarrollar un conjunto de funcionalidades que mejoren la usabilidad de la plataforma web de MateFun Infantil y realizar validaciones tanto con docentes como con niños. Tal es así, que las funcionalidades más destacadas del proyecto, fueron la creación de un asistente y un gestor de desafíos, las cuales inicialmente no estaba previsto desarrollar y fueron construidas mediante un diseño participativo con los docentes. Al involucrar su perspectiva se crearon mejoras en la aplicación y funcionalidades que posteriormente los niños pudieron utilizar con éxito. Este aporte fue clave, tanto desde el punto de vista funcional, como de los conceptos y términos utilizados.
Otro aspecto importante a resaltar es que tanto en las dos instancias de validación que se realizaron con docentes como en la validación con el grupo interdisciplinario, no hubo comentarios acerca de carencias en los contenidos que trabaja MateFun. Por el contrario, hay conceptos que exceden los conocimientos de un escolar (como por ejemplo, la función seno). Esto es importante ya que significa que a la hora de continuar ampliando la aplicación, se puede seguir focalizando en la usabilidad de la misma.
Con respecto a la validación realizada con niños, se observó que las mejoras implementadas que implicaron que la aplicación se torne más interactiva, tuvieron un mayor impacto. Más aún cuando involucraron características lúdicas, como lo es la resolución de ejercicios o las diferentes emociones que muestra el asistente. Al haber validado la aplicación con niños, se advirtió que si bien en primer ciclo de primaria podrían tener un primer acercamiento a esta, para poder aprovecharla en mayor dimensión sería necesario que el niño esté en un segundo ciclo escolar (cuarto, quinto y sexto año). De este modo, parecería correcta la idea inicial que se tenía acerca del público objetivo.
Como conclusión global acerca de la validación, a partir de la enriquecedora comunicación que se tuvo tanto con docentes como con niños, se considera que MateFun Infantil tiene potencial para ser una aplicación útil y atractiva. Sobre esta, los docentes pueden trabajar varios contenidos del programa de educación primaria y en el proceso el niño aprender algunas de las bases de la programación. Dado que en la actualidad el proyecto cuenta con una implementación avanzada validada con docentes y un pequeño grupo de niños, se podría pasar a una etapa de validación con muestras más grandes de estos últimos.

6.2. Trabajo a futuro

La característica más destacada sobre la cual se debería continuar trabajando es el gestor de desafíos. En primer lugar, del lado de los docentes se reiteraron las dudas acerca de si ellos tenían la posibilidad de crear los desafíos para que posteriormente los niños los resuelvan. Por lo tanto, sería conveniente generar una interfaz en donde el propio docente pueda crear los ejercicios que considere necesarios de acuerdo a lo que está trabajando en el aula y del lado del Servidor ofrecer un servicio que guarde, actualice y obtenga los desafíos. Luego de implementado el servicio se puede modificar Matefun Infantil Web para utilizarlo y obtener los desafíos, tal y como se explica en el Anexo C 9. Sobre la misma linea, al relevar la aplicación con el grupo interdisciplinario, surgió el planteamiento de crear un conjunto de ejercicios ordenados de menor a mayor dificultad, para que el niño los resuelva y de este modo aprenda el funcionamiento de la aplicación. Esta idea es acertada, ya que la mayoría de los lenguajes relevados en el estado del arte poseen esta característica. Esto se podría crear dándole un enfoque lúdico de forma que sea más atractivo para los niños. Otra particularidad del gestor de desafíos que se podría mejorar es las validación de las cantidades de tuberías que colocó el usuario para obtener el resultado. Se podría extender esta característica de forma que verifique si se colocaron N tuberías de tipo ADD o SUB por ejemplo.
Otro aspecto importante que mencionaron los docentes el cual hay que continuar desarrollando, es la independencia de los niños en el uso de la aplicación. Si bien con lo mencionado anteriormente ayudaría a que el niño comprenda gradualmente el uso de MateFun, sería útil poder consultar desde la aplicación y sobre cada tubería, una breve explicación de uso de la misma con un ejemplo. En este caso el asistente sería el gran candidato para realizar este tipo de tarea. Asimismo se podría continuar desarrollando la "inteligencia" de este, de modo que cuando detecte que el usuario utiliza la aplicación de una forma que se podría mejorar, se lo sugiera. Por ejemplo si borra una a una las tuberías, el asistente podría indicarle que con la selección múltiple puede realizarlo de forma más eficiente. Con este mismo objetivo también se podrían crear tutoriales en video mostrando los diferentes usos de la aplicación.
Acerca del asistente, con la intención que el mismo sea aún más interactivo, se podría evaluar la posibilidad de agregarle sonidos asociados a cada una de sus expresiones. También se podría permitir al niño asignarle un nombre y elegir el color de este. Si bien parece algo trivial, a los niños le resultaría atractivo. Del mismo modo que lo anterior, como recomendación del grupo interdisciplinario, sería bueno que el niño pueda elegir el tema general de toda la aplicación para que sea más inclusivo.
Con respecto a las funciones que permite utilizar MateFun en el nivel medio y avanzado, sería conveniente validar con docentes si alguna de ellas es necesario agregarlas al nivel básico, o si por el contrario, algunas de las que se encuentran en este último nivel deberían pasar a un nivel de mayor dificultad.
Otra característica que se recomendó estudiar, apunta a evaluar si en algunos casos valdría la pena dejar de lado el purismo que mantiene MateFun acerca que todo surge a partir de la ejecución de funciones. Se podría analizar si en algunos casos como en las representaciones gráficas, sería conveniente permitir que el niño realice trazados sobre el propio plano.
En lo que refiere al trabajo de validación, como se mencionó en las conclusiones 6.1, el mismo podría continuar tomando un grupo que esté a cargo de un docente con conocimiento de MateFun. De esta forma se tendría un conjunto de niños más heterogéneo, a su vez se le daría un uso más exhaustivo a la aplicación y así se obtendría más información sobre la cual seguir trabajando.

Capítulo 7

Anexo A. Casos de uso

En el presente capítulo se detallan los casos de uso implementados asociados con las funcionalidades incorporadas a la plataforma MateFun Infantil en esta etapa del Proyecto.

1. **Extensión de tablero simple horizontal**

Descripción	El usuario arrastra una tubería hacia una de las tres últimas columnas del borde derecho del tablero.
Precondiciones	1. El usuario tiene sesión iniciada. 2. El tablero no ha sido extendido a su tamaño máximo.
Flujo principal	1. Se expande el tamaño de la grilla agregando tres nuevas columnas. 2. Se actualiza el nuevo tamaño de matriz en el WebStorage.

Tabla 7.1: Extensión de tablero simple horizontal

2. **Extensión de tablero máxima horizontal**

Descripción	El usuario arrastra una tubería hacia una de las tres últimas columnas del borde derecho del tablero.
Precondiciones	1. El usuario tiene sesión iniciada. 2. El tablero ya ha sido extendido en cuatro ocasiones. 3. El tablero no ha sido extendido a su tamaño máximo.
Flujo principal	1. Se expande el tamaño de la grilla agregando tres nuevas columnas. 2. Se muestra mensaje indicando que fue la última extensión del tablero. 3. Se actualiza el nuevo tamaño de matriz en el WebStorage.

Tabla 7.2: Extensión de tablero máxima horizontal

3. **Extensión de tablero simple vertical**

Descripción	El usuario arrastra una tubería hacia una de las tres últimas filas del borde inferior del tablero.
Precondiciones	1. El usuario tiene sesión iniciada. 2. El tablero no ha sido extendido a su tamaño máximo.
Flujo principal	1. Se expande el tamaño de la grilla agregando tres nuevas filas. 2. Se actualiza el nuevo tamaño de matriz en el WebStorage.

Tabla 7.3: Extensión de tablero simple vertical

4. **Extensión de tablero máxima vertical**

Descripción	El usuario arrastra una tubería hacia una de las tres últimas filas del borde inferior del tablero.
Precondiciones	1. El usuario tiene sesión iniciada. 2. El tablero ya ha sido extendido en cuatro ocasiones. 3. El tablero no ha sido extendido a su tamaño máximo.
Flujo principal	1. Se expande el tamaño de la grilla agregando tres nuevas filas. 2. Se muestra mensaje indicando que fue la última extensión del tablero. 3. Se actualiza el nuevo tamaño de matriz en el WebStorage.

Tabla 7.4: Extensión de tablero máxima vertical

5. **Selección de tuberías para reubicación en el tablero**

Descripción	El usuario selecciona y arrastra un conjunto de tuberías desde la posición original al destino elegido.
Precondiciones	1. El usuario tiene sesión iniciada. 2. Debe existir por lo menos una tubería en el tablero.
Flujo principal	$\Rightarrow$ Si el destino de las tuberías seleccionadas es válido: 1. Antes de soltar las tuberías seleccionadas en el destino, se indica la celda destino en color verde. 2. Se trasladan las tuberías seleccionadas al nuevo destino. 3. Se actualiza la matriz en el WebStorage. $\Rightarrow$ Si el destino de las tuberías seleccionadas no es válido: 1. Antes de soltar las tuberías seleccionadas en el destino, se indica la celda destino en color rojo. 2. Si se suelta en dicho destino entonces no se realizará ninguna acción.

Tabla 7.5: Selección de tuberías para reubicación en el tablero

6. **Selección de tuberías para borrado**

Descripción	El usuario selecciona y arrastra un conjunto de tuberías hacia la papelera.
Precondiciones	1. El usuario tiene sesión iniciada. 2. Debe existir por lo menos una tubería en el tablero.
Flujo principal	1. Se borran del tablero las tuberías seleccionadas. 2. Se actualiza la matriz en el WebStorage.

Tabla 7.6: Selección de tuberías para borrado

7. **Creación de función a partir de selección**

Descripción	El usuario selecciona un conjunto de tuberías y las guarda como una función presionando el botón "Guardar seleccionado".
Precondiciones	1. El usuario tiene sesión iniciada. 2. El usuario debe seleccionar un conjuto de tuberías que permitan formar una función a ser guardada.
Flujo principal	1. El usuario asigna un nombre y/o un ícono para la función. 2. Se actualizan las tuberías conectando los inputs y outputs en el tablero. 3. Se edita el contenido del archivo Myfuncions y se agrega la nueva función. 4. Se carga el archivo en el intérprete.

Tabla 7.7: Creación de función a partir de selección

8. **Ocultar asistente y sección de resultados**

Descripción	El usuario presiona el botón de ocultar la sección de resultados.
Precondiciones	1. El usuario tiene sesión iniciada. 2. La sección de resultados y asistente deben estar desplegadas.
Flujo principal	1. Se oculta la sección de resultados y el asistente.

Tabla 7.8: Ocultar asistente y sección de resultados

9. **Seleccionar desafío**

Descripción	El usuario presiona el botón de desafíos y selecciona dificultad, tipo y desafío en la pantalla de selección.
Precondiciones	1. El usuario tiene sesión iniciada. 2. No se está ejecutando un desafío.
Flujo principal	1. Se editan los páramentros a nivel de sistema que establecen la ejecución del mismo junto con letra, pista, resultado esperado y atributo de correctitud a ser evaluado una vez ejecutada la solución planteada por el usuario.

Tabla 7.9: Seleccionar desafío

10. **Mostrar letra desafío**

Descripción	El usuario presiona el botón de mostrar letra del desafío.
Precondiciones	1. El usuario tiene sesión iniciada. 2. Se está ejecutando un desafío. 3. No se está mostrando la letra del desafío.
Flujo principal	1. Se edita el parámetro de sistema que indica que se debe mostrar la letra del desafío actual. 2. El asistente muestra la letra del desafío.

Tabla 7.10: Mostrar letra desafío

11. **Mostrar pista desafío**

Descripción	El usuario presiona el botón de mostrar pista del desafío.
Precondiciones	1. El usuario tiene sesión iniciada. 2. Se está ejecutando un desafío. 3. No se está mostrando la pista del desafío.
Flujo principal	1. Se edita el parámetro de sistema que indica que se debe mostrar la pista del desafío actual. 2. El asistente muestra la pista del desafío.

Tabla 7.11: Mostrar pista desafío

12. **Ejecutar solución de desafío**

Descripción	Una vez elaborado el sistema de tuberías con el objetivo de resolver el desafío, el usuario ejecuta la solución.
Precondiciones	1. El usuario tiene sesión iniciada. 2. Se está ejecutando un desafío. 3. El sistema de tuberías no tiene errores.
Flujo principal	$\Rightarrow$ Se obtiene el resultado esperado y cumple el atributo a ser chequeado: 1. Se actualiza el mensaje de éxito en el componente DisplayResult.jsx. $\Rightarrow$ Se obtiene el resultado esperado y no cumple el atributo a ser chequeado: 1. Se actualiza el mensaje de éxito pero indicando posible mejora en el componente DisplayResult.jsx. $\Rightarrow$ Si el sistema de tuberías no genera el resultado esperado: 1. Se actualiza el mensaje de error en el componente DisplayResult.jsx.

Tabla 7.12: Ejecutar solución de desafío

13. **Salir de un desafío**

Descripción	El usuario presiona el botón de salir del desafío.
Precondiciones	1. El usuario tiene sesión iniciada. 2. Se está ejecutando un desafío.
Flujo principal	1. Se actualiza la variable del sistema que retorna al modo edición fuera de desafíos, y se carga la matriz del estado guardado. 2. En pantalla se muestra el tablero en el estado anterior al comienzo del desafío.

Tabla 7.13: Salir de un desafío

14. **Prueba general de mensajería del asistente**

Descripción	El usuario inserta tuberías en el tablero, en primer lugar deja tuberías sin conectar. Luego lo corrige pero realiza una conexión de tipos de tubería distintos. Luego de corregirlo inserta dos tuberías finales para posteriormente corregirlo eliminando una de estas tuberías. Procede a guardar lo construido como función y continuar trabajando. Luego se obtiene el resultado de la ejecución y cierra sesión.
Precondiciones	1. El usuario tiene sesión iniciada.
Flujo principal	1. Se actualiza el mensaje de alerta de tuberías sin conectar en el componente DisplayResult.jsx. 2. Se actualiza el mensaje de error referido a error de tipos en el componente DisplayResult.jsx. 3. Se actualiza el mensaje de error referido a más de una tubería de salida en el componente DisplayResult.jsx. 4. Se actualiza el mensaje de éxito mostrando el nombre de la función guardada en el componente DisplayResult.jsx. 5. Se actualiza el mensaje mostrando el resultado obtenido en en el componente DisplayResult.jsx. 6. Se invoca al servicio Rest para cerrar la sesión de usuario. 7. Se limpia la sesión y la matriz del WebStorage. 8. Se vuelve a la pantalla de inicio de sesión.

Tabla 7.14: Prueba general de mensajería del asistente

15. **Prueba general de multitubería**

Descripción	El usuario construye una tubería múltiple con las siguientes entradas: entero (1) a la izquierda y color rojo a la derecha. A la tubería resultado se le anida una nueva tubería múltiple con una circunferencia de radio 3 por derecha. Nuevamente el resultado se anida y se agrega un punto con valor (2,2). Se conecta la tubería de salida y se obtiene la multitubería para evaluar el resultado. Previo a la tubería resultado se inserta una tubería izquierda y una tubería derecha para evaluar el resultado obtenido. Se sustituye el paso anterior y en su lugar se insertan las siguientes tuberías: izquierda, izquierda, derecha y se evalúa el nuevo resultado.
Precondiciones	1. El usuario tiene sesión iniciada. 2. La sesión se encuentra seteada en dificultad “Avanzado”.
Flujo principal	1. Se actualiza el tablero la multitubería resultado de tipo “Numero—Color”. 2. Se actualiza el tablero la multitubería resultado de tipo “Multipipe—Figura”. 3. Se actualiza el tablero la multitubería resultado de tipo “Multipipe—Multipipe”. 4. Se retorna el siguiente resultado: [[[[1,Color],Figura],(2,2)]]. 5. Se retorna la circunferencia de diámetro 3. 6. Se retorna el color rojo.

Tabla 7.15: Prueba general de multitubería

Capítulo 8

Anexo B. Pruebas

Con el objetivo de validar y documentar el correcto funcionamiento del sistema, se realizaron pruebas que permitieron analizar el comportamiento del mismo en los casos de uso presentados en el capítulo anterior. Esto comprendió validar tanto la correctitud de nuevos requerimientos como el funcionamiento original del sistema y sus funcionalidades. Tal como se mencionó en el Capítulo 5 se siguió el enfoque de testing exploratorio para la ejecución de esta etapa del proyecto.
A continuación se detallan los resultados de las pruebas indicando los ajustes que fueron realizados en cada caso.

1. **Extensión de tablero simple horizontal**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
Se arrastra una tubería hacia alguna de las últimas tres columnas del borde derecho del tablero.	Se expande el tablero con tres nuevas columnas	**OK**	**OK**

Tabla 8.1: Extensión de tablero simple horizontal

2. **Extensión de tablero máxima horizontal**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
Se arrastra una tubería hacia alguna de las últimas tres columnas del borde derecho del tablero.	Se expande el tablero con tres nuevas columnas y se muestra una alerta indicando que esta fue la última extensión del tablero.	**OK**	**OK**
Se arrastra nuevamente una tubería hacia alguna de las últimas tres columnas del borde derecho del tablero.	El tablero se mantiene del mismo tamaño y se muestra una alerta indicando que no es posible extender el tablero.	**OK**	**OK**
Se arrastra nuevamente una tubería hacia alguna de las últimas tres columnas del borde derecho del tablero.	El tablero se mantiene del mismo tamaño y no se muestran más alertas.	**ERROR**	**OK**

Tabla 8.2: Extensión de tablero máxima horizontal

Detalle de error: Una vez mostrada tanto la alerta de última extensión del tablero como la alerta que indica que ya no es posible extenderlo, si el usuario intentaba colocar tuberías sobre el borde derecho del tablero, la alerta volvía a ser mostrada nuevamente. Para simplificar este comportamiento, el usuario es notificado una única vez por cada tipo de alerta, para que luego de esto pueda seguir operando sobre el tablero sin recibir las mismas alertas reiteradas veces.

3. **Extensión de tablero simple vertical**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
Se arrastra una tubería hacia alguna de las últimas tres filas del borde inferior del tablero.	Se expande el tablero con tres nuevas filas	**OK**	**OK**

Tabla 8.3: Extensión de tablero simple vertical

4. **Extensión de tablero máxima vertical**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
Se arrastra una tubería hacia alguna de las últimas tres filas del borde inferior del tablero.	Se expande el tablero con tres nuevas filas y se muestra una alerta indicando que esta fue la última extensión del tablero.	**OK**	**OK**
Se arrastra nuevamente una tubería hacia alguna de las últimas tres filas del borde inferior del tablero.	El tablero se mantiene del mismo tamaño y se muestra una alerta indicando que no es posible extender el tablero.	**OK**	**OK**
Se arrastra nuevamente una tubería hacia alguna de las últimas tres filas del borde inferior del tablero.	El tablero se mantiene del mismo tamaño y no se muestran más alertas.	**ERROR**	**OK**

Tabla 8.4: Extensión de tablero máxima vertical

Detalle de error: Al igual que en el caso 2, se encuentra el error de múltiple aparición de las mismas alertas. Este error se ajusta de forma análoga al caso antes mencionado.

5. **Selección de tuberías para reubicación en el tablero**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
El usuario selecciona y arrastra un conjunto de tuberías desde la posición original al destino elegido válido.	Se muestra en color verde la celda destino y se trasladan las tuberías seleccionadas al nuevo destino	**OK**	**OK**
El usuario selecciona y arrastra un conjunto de tuberías desde la posición original a un destino elegido inválido.	Se muestra en color rojo la celda destino y no se efectúa ninguna acción.	**OK**	**OK**

Tabla 8.5: Selección de tuberías para reubicación en el tablero

6. **Selección de tuberías para borrado**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
El usuario selecciona y arrastra un conjunto de tuberías hacia la papelera.	Se borran del tablero las tuberías seleccionadas	**OK**	**OK**

Tabla 8.6: Selección de tuberías para borrado

7. **Creación de función a partir de selección**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
El usuario selecciona un conjunto válido a ser guardado como una función y presiona el botón “Guardar seleccionado”.	La función es guardada con el nombre y/o icono seleccionado y se remplazan en el tablero las tuberías seleccionadas por la función guardada.	**ERROR**	**OK**
El usuario selecciona un conjunto inválido a ser guardado.	El botón “Guardar seleccionado”no está habilitado. seleccionadas	**OK**	**OK**

Tabla 8.7: Creación de función a partir de selección

Detalle de error: En algunas pruebas la conexión de la función creada con las tuberías originales no se conectaban luego de guardada la función. Se trabajo en la mejora del algoritmo para resolver estos casos más complejos.

8. **Ocultar asistente y sección de resultados**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
El usuario presiona el botón de ocultar la sección de resultado.	Se oculta la sección de resultados y el asistente.	**OK**	**OK**

Tabla 8.8: Ocultar asistente y sección de resultados

9. **Seleccionar desafío**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
El usuario presiona el botón de desafíos y selecciona dificultad, tipo y desafío en la pantalla de selección.	Comienza el desafío, se muestra la letra del mismo y aparecen disponibles los botones de desafío.	**OK**	**OK**

Tabla 8.9: Seleccionar desafío

10. **Mostrar letra desafío**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
El usuario presiona el botón de mostrar la consigna del desafío	Se muestra la consigna del desafío.	**OK**	**OK**

Tabla 8.10: Mostrar letra desafío

11. **Mostrar pista desafío**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
El usuario presiona el botón de mostrar pista del desafío	Se muestra la pista del desafío.	**OK**	**OK**

Tabla 8.11: Mostrar pista desafío

12. **Ejecutar solución de desafío**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
Se obtiene el resultado esperado y cumple el atributo a ser chequeado.	Se muestra el resultado junto con un mensaje de éxito y el asistente cambia a estado “fun”.	**OK**	**OK**
Se obtiene el resultado esperado y no cumple el atributo a ser chequeado.	Se muestra el resultado junto con un mensaje de éxito, pero indicando que la solución puede ser mejorada y el asistente cambia a estado “fun”.	**OK**	**OK**
Si el sistema de tuberías no genera el resultado esperado.	Se muestra el resultado junto con un mensaje de error y el asistente cambia a estado “ko”.	**OK**	**OK**

Tabla 8.12: Ejecutar solución de desafío

13. **Salir de un desafío**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
El usuario presiona el botón de salir del desafío.	Se retorna a la pantalla inicial de trabajo.	**ERROR**	**OK**

Tabla 8.13: Salir de un desafío

Detalle de error: luego de salir del desafío no volvía a cargar la matriz previa al inicio del desafío. Este error fue ajustado y testeado nuevamente.

14. **Prueba general de mensajería del asistente**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
Se insertan tuberías sin conectar.	Se muestra la alerta asociada y el gesto gráfico del asistente cambia al estado “shocked”.	**OK**	**OK**
Se realiza una conexión de tipos de tubería distintos.	Se muestra el error asociado y el gesto gráfico del asistente cambia al estado “ko”.	**OK**	**OK**
Se insertan dos tuberías de salida finales.	Se muestra el error asociado y el gesto gráfico del asistente cambia al estado “ko”.	**OK**	**OK**
Se guarda lo construido como función, seteanado nombre e icono.	Se muestra el mensaje de éxito junto con el nombre de la función, el asistente cambia al estado “fun”.	**OK**	**OK**
Se ejecuta el sistema de tuberías construido.	Se muestra el mensaje de resultado.	**OK**	**OK**
El usuario cierra sesión	Se retorna a la pantalla de login.	**OK**	**OK**

Tabla 8.14: Prueba general de mensajería del asistente

15. **Prueba general de multitubería**

Escenario	*Comportamiento esperado*	*Resultado Inicial*	*Resultado Final*
Se inserta la multitubería con entradas entero (1) a la izquierda y color (rojo) a la derecha.	Se actualiza la salida de la multitubería de tipo “Número—Color”.	**OK**	**OK**
Se anida a la multitubería una circunferencia de diámetro 3 por derecha.	Se actualiza la salida de la multitubería de tipo “Multipipe—Figura”.	**OK**	**OK**
Se anida por derecha insertando un punto con valor (2,2). Se conecta una tubería de salida y se ejecuta.	Se retorna el siguiente resultado: [[[[1,Color],Figura],(2,2)]]	**ERROR**	**OK**
Se inserta una tubería de selección izquierda y se obtiene el resultado.	Se retorna una circunferencia de diámetro 3.	**OK**	**OK**
Se sustituye el paso anterior y en su lugar se insertan las siguientes tuberías: izquierda, izquierda, derecha y se evalúa el nuevo resultado.	Se retorna el color rojo.	**OK**	**OK**

Tabla 8.15: Prueba general de multitubería

Capítulo 9

Anexo C. Integración para obtener desafíos

Una vez implementada la funcionalidad de guardado y obtención de desafíos en el Servidor, exponiendo los servicios correspondientes, se deberá modificar Matefun Infantil para completar la integración.

1. En primera instancia se deberá crear una función dentro del archivo **service.js** que permita realizar un get de los desafíos según el usuario, según como se haya implementado la API.

2. Luego se deberá utilizar la función creada para obtener los desafíos correspondientes, modificando el archivo Configuration.jsx para intentar en primera instancia obtener los desafíos desde el servidor mediante un request. En caso de error, obtener los desafíos cargados en el archivo ya utilizado.

Índice de tablas

Índice de figuras

Referencias

[1] Repositorio Cartoon-Avatars. *https://github.com/jcolombo/cartoon-avatars* (visitado 15-10-2021).

[2] CODE. *https://code.org* (visitado 19-07-2021).

[3] Entorno de Prueba React infinite scroller. *https://codesandbox.io/s/yk7637p62z* (visitado 19-07-2021).

[4] Ejemplo de uso React Selectable. *http://unclecheese.github.io/react-selectable/example/* (visitado 23-02-2022).

[5] DOM. *https://www.w3.org/TR/REC-DOM-Level-1/introduction.html* (visitado 10-12-2021).

[6] Recursión en Scratch. *https://en.scratch-wiki.info/wiki/Recursion* (visitado 19-07-2021).

[7] Componente React infinite scroller. *https://github.com/ankeetmaini/react-infinite-scroll-component* (visitado 19-07-2021).

[8] Demo React infinite scroller. *https://danbovey.uk/react-infinite-scroller/demo/* (visitado 19-07-2021).

[9] Repositorio React infinite scroller. *https://github.com/danbovey/react-infinite-scroller* (visitado 19-07-2021).

[10] JavaFX. *https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm* (visitado 19-07-2021).

[11] Javascript. *https://www.javascript.com/* (visitado 15-10-2021).

[12] Cem Kaner. *A Tutorial in Exploratory Testing*. 2008.

[13] Lightbot. *https://lightbot.com/* (visitado 19-07-2021).

[14] Simon Marlow. *Haskell 2010 Language Report*. *https://www.haskell.org/onlinereport/haskell2010/* (visitado 19-07-2021).

[15] Meta. *https://about.facebook.com/meta* (visitado 15-10-2021).

[16] Marcelo Estayno, Marisa Panizzi. *Participación de los usuarios en el proceso de desarrollo de software*. 2009.

[17] Emiliano San Roman, Federico Luongo, Stéfano Pesamosca. *Extensión de Funcionalidades y Validación de MateFun Infantil - Plataforma Web (Estado del Arte)*. Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, 2021.

[18] PilasBloques. *https://pilasbloques.program.ar/online/* (visitado 19-07-2022).

[19] React. *https://es.reactjs.org/* (visitado 15-10-2021).

[20] React-Dnd. *https://react-dnd.github.io/react-dnd/about* (visitado 19-07-2021).

[21] React-Kawaii. *https://react-kawaii.vercel.app/* (visitado 15-10-2021).

[22] React-Redux. *https://react-redux.js.org/* (visitado 15-10-2021).

[23] Repositorio React Selectable. *https://github.com/unclecheese/react-selectable* (visitado 23-02-2022).

[24] Demo React Selectable Simple. *https://jsfiddle.net/xkwp4z8y/* (visitado 19-07-2021).

[25] Repositorio React Selectable Simple. *https://github.com/kwyoung11/react-selectable-simple* (visitado 19-07-2021).

[26] Ian Sommerville. *Ingeniería de Software*. Pearson, 9 edition, 2011. ISBN E-BOOK: 978-607-32-0604-4.

[27] Squeak. *https://squeak.org/* (visitado 23-02-2022).

[28] SVG. *https://www.w3schools.com/graphics/svg_intro.asp* (visitado 10-12-2021).

[29] Viskell. *https://github.com/viskell/viskell* (visitado 19-07-2021).

[30] Nicolás Vázquez. *Mejoras al Intérprete MateFun*. Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, 2019.

[31] Lucía Labat y Felipe Parodi. *MateFun Infantil - Plataforma Web*. Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, 2020.

Glosario

Android: Sistema operativo móvil basado en el núcleo Linux y otros software de código abierto.

API: Application Programming Interface (interfaz de programación de aplicaciones) es un conjunto de definiciones y protocolos que se usa para diseñar e integrar el software de las aplicaciones. Permite que sus componentes se comuniquen con otros sin necesidad de saber cómo están implementados.

Aplicación: Programa informático diseñado como una herramienta para realizar operaciones o funciones específicas.

Aplicación web: Aplicación que los usuarios pueden utilizar accediendo a un servidor web a través de internet o de una intranet mediante un navegador.

Backtracking: Estrategia para encontrar soluciones a problemas que satisfacen restricciones. Se busca encontrar la mejor combinación posible en un momento determinado, por eso, se dice que este tipo de algoritmo es una búsqueda en profundidad.

Biblioteca: Conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.

Bloques: Es la unidad mínima de manipulación de un programa, en las aplicaciones de programación por bloques. En el caso de la plataforma web de MateFun Infantil también puede ser nombrado como "Tubería".

Código: En el contexto de programación, refiere al lenguaje por el cual funcionan las computadoras. Comprende un conjunto de instrucciones y datos a ser procesados automáticamente

Código modularizado: Código al cual se le aplicó un proceso de selección y agrupación de instrucciones de programación.

Celda: En MateFun Infantil es el espacio del tablero donde se colocan las tuberías.

Compilación: Proceso por el cual un programa traduce código escrito en un lenguaje de programación a otro lenguaje.

Depuración: Es el proceso de identificar y corregir errores de programación.

Dummy: En MateFun Infantil refiere a la tubería que conecta tuberías de función, variable, constante o resultado.

Especialista en interfaz de usuario: Profesional que trabaja con el medio que permite la interacción del usuario con el producto, en este caso software, y se preocupa por la usabilidad que el mismo posee.

Estado del Arte: Modalidad de investigación documental que permite el estudio del conocimiento acumulado dentro de un área específica.

Función: Regla de correspondencia entre dos conjuntos de tal manera que a cada elemento del primer conjunto le corresponde uno y sólo un elemento del segundo conjunto.

Funciones recursivas: Aquella función que se invoca a sí misma de forma repetida, hasta que satisface alguna determinada condición.

Haskell: Lenguaje de programación estandarizado multi-propósito, funcionalmente puro, con evaluación no estricta y memorizada, y fuerte tipificación estática.

Instituto Normal de Montevideo: Institución de enseñanza terciaria encargada de impartir la formación en magisterio perteneciente al Consejo de Formación en Educación de la Administración Nacional de Educación Pública del Uruguay.

Imperativo: Que se manifiesta como orden o imposición.

Java: Lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems.

Javascript: Lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

JSON: Formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera un formato independiente del lenguaje.

MAC: Macintosh, abreviado como Mac, es la línea de ordenadores personales diseñada, desarrollada y comercializada por Apple Inc.

Modal: Es una ventana que aparece superpuesta a toda la página actual.

Open Source: Es una licencia de software que permite que tanto el código fuente como los archivos binarios sean modificados y redistribuidos libremente y sin tener que pagar al autor original.

Panel de resultado: En MateFun Infantil es el sector de la pantalla en donde se despliegan los resultados de la ejecución de los programas.

Pipes: En MateFun Infantil es un término utilizado análogo a tubería.

Prototipo: Implementación realizada con técnicas de programación del sistema interactivo propuesto que reproduce el funcionamiento de una parte importante de las funcionalidades con el objetivo de probar determinados aspectos del sistema final.

React: Biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página.

Squeak: Busca ser un meta-medio. Un lugar donde conviven todos los medios de expresión conocidos hasta la fecha (texto, video, sonido, música, gráficos 2D, gráficos 3D, TextToSpeech, etc) y que sirve, a su vez, como soporte para el desarrollo de nuevos medios.

Terminología: Conjunto de términos o palabras propias utilizadas en una ciencia, técnica, o especialidad, o por un autor.

Tipo: Modelo ideal que reúne los caracteres esenciales de todos los seres de igual naturaleza. En MateFun Infantil los tipos son: color, figura, booleano, número, multitubería, lista.

Toast messages: Refiere a un elemento de control gráfico que comunica ciertos eventos al usuario sin forzarlo a reaccionar inmediatamente a esta notificación.

Tooltip: Herramienta de ayuda visual, que funciona al situar el cursor sobre algún elemento gráfico, mostrando una ayuda adicional para informar al usuario de la finalidad del elemento sobre el que se encuentra.

Token: Cuando el usuario se ha podido validar como un usuario de la aplicación, recibe una cadena encriptada como respuesta. Esa cadena es el token y sirve para que en los siguientes accesos, el usuario pueda informar al servidor que ya ha pasado por el proceso de autenticación.

Tuberías: Es la unidad mínima de manipulación de la plataforma web MateFun Infantil.

Usabilidad: Neologismo que refiere a la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto.

Validación: Proceso que demuestra que el sistema cumple con las funciones de las cuales fue designado.

Windows: Familia de distribuciones de software para PC, servidores, sistemas empotrados y antiguamente teléfonos inteligentes desarrollados y vendidos por Microsoft y disponibles para múltiples arquitecturas, tales como x86, x86-64 y ARM.