

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ARTES Y DISEÑO

DIVISIÓN DE EDUCACIÓN CONTINUA Y EXTENSIÓN ACADÉMICA

2024-1

MEMORIA DE INVESTIGACIÓN - PRODUCCIÓN

DESARROLLO DE PÁGINAS RESPONSIVAS E INTERACTIVAS.

QUE PRESENTA

KEVIN JOSUE FERNANDEZ MENDOZA

PROYECTO

ONLINE ROKU COMPILER

RESPONSABLE ACADÉMICO

DR. AMADOR ULISES ROSAS GARCÍA

ASESOR METODOLÓGICO

DR. AMADOR ULISES ROSAS GARCÍA



UNAM
FACULTAD
DE ARTES
Y DISEÑO



DIVISIÓN
DE EDUCACIÓN CONTINUA
Y EXTENSIÓN ACADÉMICA



CONTENIDO

Introducción	4
Marco Teórico.....	5
SceneGraph:	5
BrightScript:.....	6
Introducción al Problema:.....	8
Hipótesis:.....	8
Objetivo General:	8
Objetivos Particulares:	9
Capítulo I: Diseño	10
Objetivo	10
Específico.....	10
Medible	10
Alcanzable	10
Relevante.....	11
Tiempo.....	11
Misión.....	11
Visión.....	11
Valores.....	11
Buyer person (Cliente ideal)	13
Diseño Visual	14
Moodboard	14
Arquitectura de la información	17
Mapa de Sitio	17
Wireframes y wireflows.....	20
Wireframes.....	20
Mockups	20
Prototypes	20
Wireflows	21
Maquetación	21
Capítulo II: Desarrollo.....	38
Introducción al Entorno de Desarrollo:.....	38

Back-End	38
Front-End.....	39
Pruebas y Depuración:	47
Conclusión	48
Referencias	49

Introducción

La creación de herramientas que simplifiquen y agilicen el desarrollo de software se ha convertido en una prioridad, por ende, los compiladores juegan un papel fundamental al traducir el código fuente de un lenguaje de programación a un formato ejecutable, facilitando así la depuración de código, la creación de aplicaciones y sistemas complejos. En este contexto, el presente trabajo aborda la creación de un compilador web el cuál será de ayuda a la hora de desarrollar aplicaciones para dispositivos Roku.

El lenguaje de programación Roku ha ganado popularidad en el ámbito del desarrollo de aplicaciones para dispositivos de transmisión de contenido multimedia. No son solo datos al aire, los dispositivos Roku son los más usados para la distribución digital de contenido multimedia o streaming, tan solo en Estados Unidos, según Evan Shapīro quién es un gran referente en la industria, sin embargo, el desarrollo de aplicaciones en Roku aún está en pañales, las herramientas existentes son muy pocas, debido a que Roku tiene su propio sistema operativo, por ende, es un sistema prácticamente cerrado

La cantidad de desarrolladores para este lenguaje también es algo a lo que incluso podríamos llamar una comunidad. Pero la relevancia de la reducida comunidad de desarrolladores dedicados a Roku en comparación con otros entornos de desarrollo ha acrecentado la ausencia de un compilador en línea accesible y eficiente, lo cual representa un obstáculo en la fluidez del proceso de desarrollo.

La falta de acceso a los dispositivos físicos necesarios para probar y depurar las aplicaciones Roku también puede ser un gran impedimento significativo, por lo que este proyecto se presenta como una solución que no solo beneficia a los desarrolladores experimentados al ofrecerles una herramienta eficiente, sino que también proporciona un entorno de aprendizaje accesible y económico para aquellos que están dando sus primeros pasos en el desarrollo de aplicaciones Roku, evitando así una inversión inicial considerable en hardware especializado.

Con todo lo dicho anteriormente, este proyecto buscará proporcionar una plataforma en línea que permita a la comunidad, escribir, depurar y compilar código de Roku, de una manera ágil, asequible y fiable que fomente la creación de aplicaciones innovadoras para la plataforma; de igual forma se busca eliminar barreras de entrada para desarrolladores novatos o aquellos que están incursionando en el mundo de Roku.

A lo largo de esta memoria, se documentará exhaustivamente el proceso creativo, diseño, desarrollo, responsividad e interactividad, así como el análisis del compilador en línea para el lenguaje de programación Roku, haciendo uso de las herramientas y tecnologías aprendidas durante el diplomado y también haciendo hincapié en cómo esta herramienta tiene el potencial de ampliar el campo de desarrollo en la comunidad dedicada a los sistemas de streaming y principalmente en el desarrollo de Roku.

Marco Teórico

El desarrollo de aplicaciones para dispositivos Roku implica el uso de dos lenguajes de programación: SceneGraph y BrightScript. Ambos lenguajes están pensados para complementarse de igual manera que HTML y JavaScript lo hacen.

Esta analogía se materializa en la forma en que SceneGraph despliega la estructura visual de la aplicación de manera similar a HTML, mientras que BrightScript actúa como el motor detrás de la interactividad, de manera análoga a cómo JavaScript potencia las acciones dinámicas en las páginas web. Esta simbiosis de lenguajes en el entorno de desarrollo de Roku no solo simplifica la construcción de aplicaciones, sino que también permite a los desarrolladores aprovechar lo mejor de cada lenguaje para lograr un producto final robusto y eficiente.

SceneGraph:

Roku SceneGraph es un framework en XML orientado a Objetos. Su función principal se centra en la creación de la Interfaz de Usuario (UI), proporcionando un entorno estructurado para diseñar y desarrollar elementos visuales.

SceneGraph usa una estructura de árbol, en la que los nodos representan elementos de imagen los cuales definen una escena interactiva. Los nodos más profundos en la estructura se mostrarán por encima de los nodos precedentes. Cada nodo en la estructura de árbol es un objeto cuyos estados son guardados como atributos en un conjunto de campos (fields). Conforme se va recorriendo la estructura, se van conservando los estados desde el nodo padre el cual controlará como van a ser mostrados los nodos hijos.

Los nodos que se agrupen de forma jerárquica heredaran las propiedades del nodo padre, por ejemplo, si el nodo padre tiene por opacidad el valor de 0.5 (dado en un rango de 0 a 1), el valor de opacidad de los hijos será multiplicado hasta 0.5 dado que el nodo padre lo definió de este modo.

Dentro de SceneGraph existen nodos que pueden ser mostrados y otros que no (Renderable/non-renderable nodes). Los nodos que no se muestran, son ignorados a la hora de recorrer la estructura de árbol ya que generalmente definen información extra que será usada en la escena, ejemplos de estos nodos son temporizadores y animaciones (timers, animations).

Todos los nodos y la mayoría de sus campos son observables, esto quiere decir que los cambios que reciban serán continuamente monitoreados, por lo que es posible establecer funciones de callback.

Como en la mayoría de los lenguajes, es posible crear nodos o componentes personalizados. Estos componentes personalizados permiten a los desarrolladores ir más allá de las funcionalidades predefinidas y diseñar elementos visuales específicos que se ajusten a las necesidades particulares de sus aplicaciones.

Un componente personalizado en SceneGraph es esencialmente un conjunto de nodos XML y su correspondiente lógica en BrightScript que se encapsulan en una única entidad. Los componentes personalizados heredarán siempre de algún nodo ya existente.

BrightScript:

Roku BrightScript es el lenguaje de scripting encargado de definir la lógica de la aplicación, es un lenguaje que soporta el tipado dinámico y tipado declarado, el cual nos ayuda a construir aplicaciones multimedia y en red para dispositivos basados en Roku O.S.

El lenguaje está 100% desarrollado en C para su eficiencia y portabilidad, es un lenguaje que compila código en bytecode el cuál es ejecutado por un intérprete, esto se hace cada vez que un script se carga y se ejecuta.

Este lenguaje tiene una librería de componentes nativos integrados, los cuales hacen el desarrollo mucho más sencillo. La sintaxis del lenguaje es similar a la manejada por Python o Ruby. El manejo de estructuras de datos es mediante arreglos asociativos, listas y los nodos de los componentes.

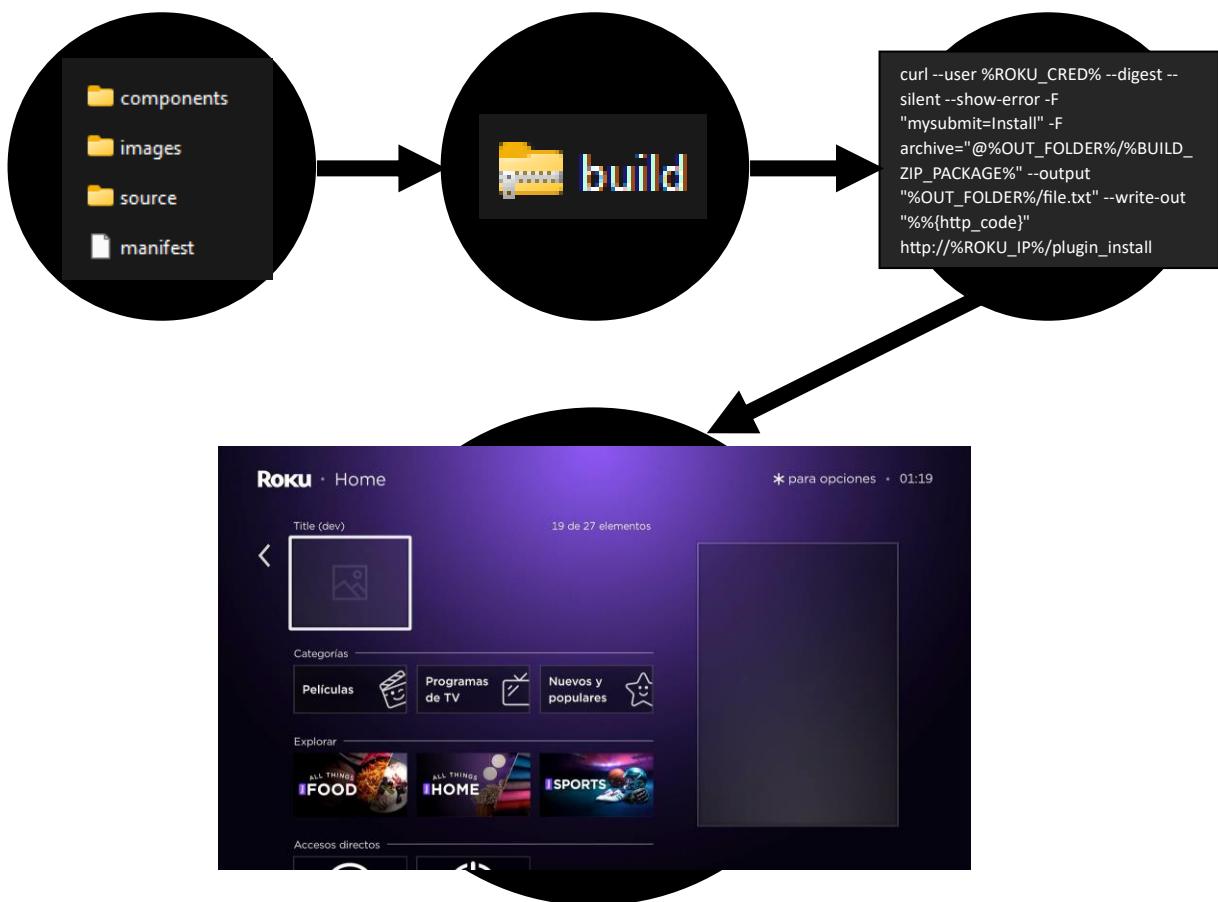
En BrightScript se hace un manejo extenso del tipo de dato “enteros”, debido a su complejidad en los procesadores de los dispositivos Roku, los tipos de datos “flotantes” se reservan para casos donde sean realmente necesarios.

Como ya se ha comentado, las aplicaciones y herramientas existentes para SceneGraph y BrightScript son muy pocas, ya que al ser lenguajes creados únicamente para el desarrollo de aplicaciones en dispositivos Roku su campo de uso es más que limitado.

Sumergiéndonos más en la parte técnica, la construcción de un proyecto de Roku funciona de la siguiente manera:

- El proyecto requiere de un archivo `manifest`, en el cual se especifican el título del canal, la versión, la resolución a mostrar, entre otras variables necesarias para el diseño del canal. Una carpeta `source` la cual contendrá el archivo `main.brs`, el cuál es el que instancia todo el canal. Una carpeta `components`, la cual contendrá los componentes que sean creados durante el desarrollo. Una carpeta `images`.

- El desarrollador debe tener un dispositivo Roku con el modo desarrollador (developer mode) activado y una cuenta inscrita al programa de desarrolladores Roku.
- El desarrollador deberá publicar su aplicación de forma local a su dispositivo Roku. La aplicación deberá ser un archivo de tipo `.zip` la cuál contendrá todo el proyecto comprimido. Existen dos formas de publicar localmente (deploy) un canal de desarrollo. Primero, por medio de la URL (Roku IP) provista por el dispositivo Roku, ahí encontraremos una aplicación web cuya interfaz nos ayudará a subir el archivo `.zip`. La segunda opción es haciendo uso de los comandos de `make` y/o la librería `curl`.
- Existen dos formas de depuración de código (debugging), primero haciendo uso del comando `telnet roku-ip-address 8085` y segundo, con los registros en terminal colocados dentro del código; mediante la extensión para visual studio code desarrollada por la comunidad.
- Solo es posible tener un canal de desarrollo publicado a la vez.



Introducción al Problema:

El desarrollo de aplicaciones para dispositivos Roku ha experimentado un crecimiento significativo debido a la popularidad de estos dispositivos en la distribución digital de contenido multimedia. Sin embargo, enfrenta desafíos considerables, ya que las herramientas de desarrollo existentes son limitadas y el acceso a dispositivos físicos Roku para pruebas puede ser restrictivo. La falta de un compilador en línea eficiente y accesible agrega complejidad al proceso de desarrollo y representa un obstáculo para la expansión de la comunidad de desarrolladores dedicados a Roku. En este contexto, surge la necesidad de desarrollar un compilador web que brinde una solución integral, mejorando la eficiencia del desarrollo y proporcionando un entorno accesible y económico para todos los niveles de desarrolladores.

Hipótesis:

Dada la creciente popularidad del lenguaje de programación Roku y las limitaciones existentes en las herramientas de desarrollo disponibles, se plantea que la creación de un compilador web para Roku contribuirá significativamente a la eficiencia y accesibilidad en el desarrollo de aplicaciones para dispositivos Roku. Se espera que este compilador en línea simplifique el proceso de codificación, depuración y prueba, eliminando barreras de entrada para desarrolladores novatos y proporcionando una solución práctica para aquellos que carecen de acceso a dispositivos físicos Roku.

Objetivo General:

Desarrollar un compilador web en línea que permita a los desarrolladores escribir, depurar y compilar código para dispositivos Roku de manera eficiente y accesible, fomentando la innovación en la creación de aplicaciones para la plataforma y superando las limitaciones actuales en las herramientas de desarrollo disponibles.

Objetivos Particulares:

1. Crear una plataforma web intuitiva y fácil de usar que admita los lenguajes de programación SceneGraph y BrightScript utilizados en el entorno de desarrollo de Roku.
2. Proporcionar a los usuarios la capacidad de visualizar registros en tiempo real, incluyendo advertencias y errores, para facilitar la identificación y corrección de problemas en el código.
3. Garantizar la responsividad y accesibilidad de la aplicación en dispositivos móviles y de escritorio, priorizando la experiencia del usuario desde cualquier ubicación y dispositivo.

Se definirán los objetivos de forma más clara y precisa en el capítulo 1.

Capítulo I: Diseño

Objetivo

El objetivo será desarrollar un compilador en línea en el cual se tenga la capacidad de trabajar proyectos Roku desde cualquier parte y sin la necesidad de tener un dispositivo físico. De igual modo, se busca tener una herramienta que facilite ejecuciones de líneas de código muy cortas de modo que si tu dispositivo se encuentra ocupado con una aplicación más pesada no tengas que reiniciar el proceso de publicación, así, las pequeñas pruebas en el desarrollo serán mucho más eficientes.

Específico

Crear una aplicación web capaz de crear, administrar y publicar proyectos para la creación de aplicaciones para dispositivos Roku.

El usuario deberá:

- Ser capaz de hacer uso de los lenguajes de programación SceneGraph y BrightScript en su totalidad.
- Tener la posibilidad de visualizar los registros (logs) brindados en la ejecución del proyecto en tiempo real, así como las advertencias y errores que existan en el código.
- Ser capaz de visualizar la aplicación en ejecución mediante un reproductor de video.

Medible

La aplicación deberá ser construida basándose en el tipo de entregas del framework SCRUM en la que toda versión sacada, por más mínima que sea, deberá proveer valor a la aplicación. Entregando siempre funcionalidad nueva.

Alcanzable

Realizar un compilador en línea es un trabajo bastante complejo, el cual requiere conocimientos en múltiples campos de las tecnologías de la información, es por eso, que se espera entregar una primera versión del sistema; una versión que no incluya un sistema de transpilación, sino uno de lectura de archivos y ejecución en el servidor. Se espera que con los conocimientos que se vayan adquiriendo a lo largo del proyecto, se pueda completar en tiempo y forma esta primera versión.

Un punto importante por considerar es que cada usuario necesitaría acceder a un dispositivo Roku el cuál se mostraría en el reproductor de video, debido a esto, es

por lo que en un inicio el alcance del reproductor de video en el proyecto se mantendrá de manera personal.

Relevante

En un inicio el proyecto tiene un valor personal muy alto, pero sin duda al ser una herramienta que aún no ha sido desarrollada, también tiene un gran valor entre la comunidad entusiasta del desarrollo de aplicaciones para dispositivos Roku.

Tiempo

El plazo específico del proyecto será de 6 meses dividiéndolo en módulos con una duración específica para cada uno. Para el primer módulo se destinarán 6 semanas, en las que se entregará el diseño, interfaz de usuario, y la interactividad de la aplicación.

La mayoría de los obstáculos con los que podríamos encontrarnos vendrían dados por la falta de experiencia o conocimientos, así como el tiempo dedicado al desarrollo del proyecto.

Misión

Diseñar y desarrollar una herramienta web con la cuál reducir tiempos de prueba y desarrollo en el día a día de los desarrolladores trabajando en la construcción de aplicaciones Roku.

Proveer una interfaz cómoda e intuitiva, extendiendo así lo máximo posible los beneficios del desarrollo y depuración de código de aplicaciones Roku, de modo que todo lo necesario se encuentre en un único sitio web.

Visión

Tener y mantener la aplicación de un compilador Roku en línea, brindar una aplicación web capaz de sustituir el desarrollo local o simplemente ser un complemento de este. Actualizar el sistema, de modo que cumpla con temas de seguridad, responsividad, interactividad, reactividad y visibilidad, para así pensar en la opción de sacar el proyecto al público como una herramienta más disponible para el desarrollo de aplicaciones Roku.

Valores.

Simplicidad: Se busca que el sistema sea fácil de entender y de usar.

Confidencialidad: En caso de hacer uso de información sensible y que tenga que ser almacenada, tratarla con la mayor confidencialidad y respeto.

Responsividad: Se busca que la aplicación funcione en todo tipo de dispositivo, sin que el tamaño de pantalla o forma del dispositivo afecte a la interfaz de usuario ni la lógica del sistema.

Accesibilidad: Debe ser viable, en su mayoría, el uso de la aplicación sin importar las capacidades físicas o motoras del usuario.

Visibilidad: La aplicación debe cumplir con las reglas de SEO para su fácil indexación en buscadores, de modo que los usuarios accedan fácilmente sin tener que morir en el intento de la búsqueda.

Colaboración: Debe pensarse como una herramienta de ayuda para los interesados en la construcción de aplicaciones en dispositivos Roku.

Buyer person (Cliente ideal)

A continuación, mostraremos un perfil de audiencia o usuario el cuál será, primordialmente, más no únicamente, quién use o consuma nuestra aplicación.

 <p>DANI</p> <p>ESTUDIANTE DE SECUNDARIA REALIZANDO PRÁCTICAS O UN LÍDER TÉCNICO TRABAJANDO PARA UNA MULTINACIONAL O CUALQUIERA RELACIONADO A LAS TECNOLOGÍAS Y EL DESARROLLO DE SOFTWARE.</p> <p>Edad: Entre 15-60 años</p> <p>Sexo: No relevante</p> <p>Ubicación: En cualquier parte del mundo.</p> <p>Ingresos: No es relevante.</p> <p>Nivel Educacional: Optimista autodidacta, o alguien especializado en tecnologías de la información.</p> <p>Situación sentimental: No relevante</p> <p>Detalles Profesionales: Freelancer, o un ejecutivo de empresa, incluso alguien con su emprendimiento propio.</p>	<p>De mi: Me apoyo del compilador en línea para facilitar el desarrollo de mi aplicación para dispositivos Roku. No solo uso el sitio por necesidad, sino porque me apasiona mucho el desarrollo y no tengo la capacidad de obtener un dispositivo Roku en este momento, también porque solo quiero una forma ligera de desarrollar, haciendo uso de la facilidad de la portabilidad de la web desde cualquier dispositivo.</p> <p>Responsabilidades Desde entregar una tarea para recibir una evaluación en la escuela hasta administrar equipos multidisciplinarios en diferentes países.</p> <p>Objetivos Espero hacer uso de la herramienta para lograr un objetivo personal o profesional.</p> <p>Desafíos No puedo acceder a un dispositivo Roku, mi ordenador no es capaz de ejecutar programas de desarrollo, quiero agilizar mis pruebas y cambiar mis canales de desarrollo desperdicia mucho mi tiempo.</p> <p>Frustraciones La dificultad y falta de herramientas para el desarrollo de aplicaciones en Roku.</p>
---	--

Diseño Visual

Moodboard

Un moodboard o por su traducción al español, tablero de inspiración, es una forma de representar muchas cosas, tus gustos, mostrar una marca o un sentimiento. El moodboard da a mostrar todo esto de forma visual, haciendo uso de imágenes, colores y textos cortos.

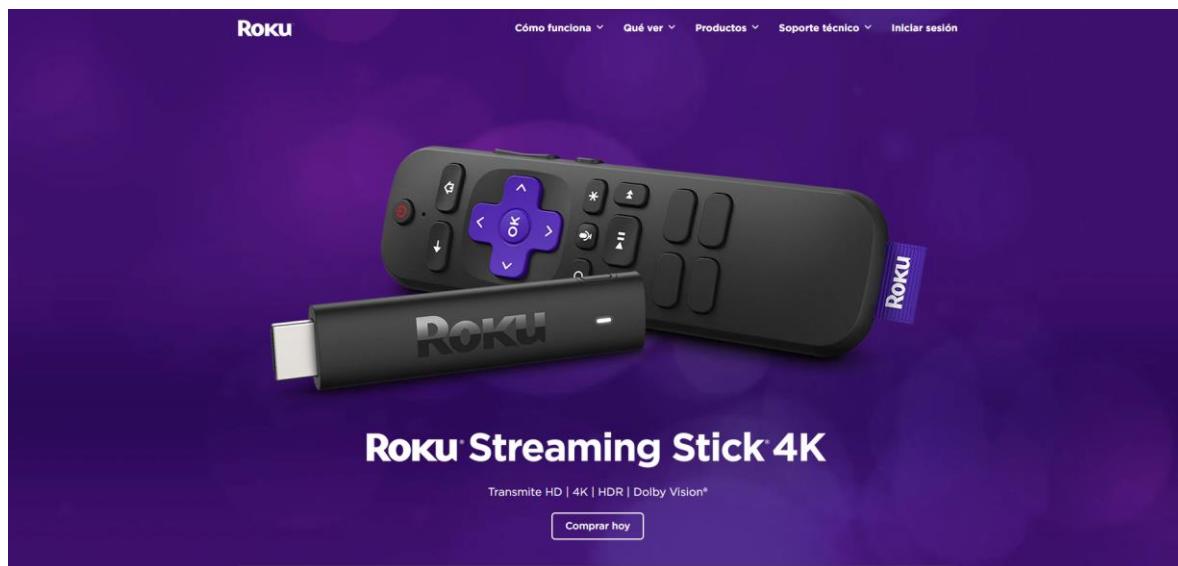
En ese sentido, el moodboard del proyecto representará la esencia de la aplicación, se mostrarán colores que definan la personalidad del sitio, imágenes que plasmen el carácter y textos que den a entender el moodboard de un modo más verbal.

Colores

La aplicación está enfocada para los desarrolladores de aplicaciones Roku, en este sentido, la aplicación debe hacernos sentir justamente eso, que está enfocada en Roku.

Los colores de Roku son básicamente el morado, diferentes variaciones de sus tonalidades y el blanco.

Si revisamos su sitio web, podemos ver que su página inicial es sencilla, mostrando al cien el clásico color morado que los representa, su clásico control remoto y uno de sus productos más top. Notamos que el branding o los logos se muestran por todo el sitio, también se muestran los textos en tamaños dependiendo de la importancia que se les quiere dar. Los colores de los textos se van jugando de modo que sobresalgan sobre los colores de fondo y las imágenes.



La aplicación será un compilador; por este motivo, tendrá terminales, selector de archivos, editor de textos y botones. En este aspecto, no es mentira que los desarrolladores prefieren los colores oscuros u oscuros suaves a los colores claros, por lo tanto, tendremos que buscar colores que armonicen con los colores de Roku y los usados comúnmente en los temas de los editores de texto más populares.

Teniendo todo esto en cuenta se han elegido 5 colores principales que formarán parte de la esencia del sitio:



#662d91 (Roku color)

Principalmente para el fondo y títulos llamativos.



#854a88

Botones o texto que sobresalga sobre el color del editor de texto y terminales.



#303030

Se usará en los fondos del editor de texto, terminales y consolas.



#474747

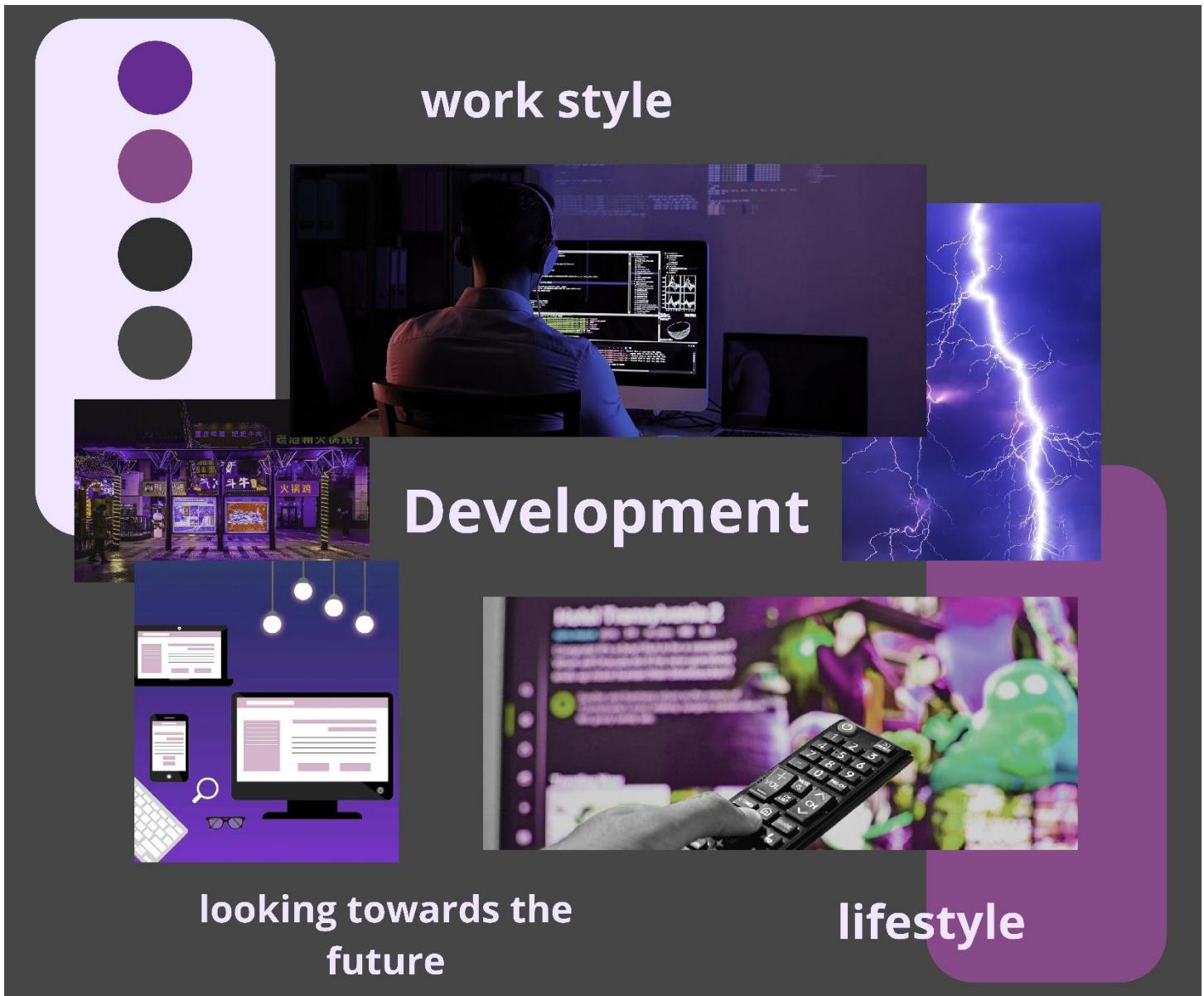
Color más suave para diferenciar ciertas secciones como la barra lateral (sidebar).



#F2E6FF

Textos que contrasten con botones y los fondos oscuros.

Una vez habiendo analizado los colores que usaremos en el proyecto, se ha creado el siguiente moodboard el cuál trata de plasmar el carácter del compilador en línea.



Arquitectura de la información

La arquitectura de la información es una disciplina que busca organizar cualquier producto o servicio como si fueran bloques de información, de modo que estos puedan ser optimizados y ordenados, facilitando su comprensión y haciéndolos más fácil de encontrar.

Un ejemplo de esta organización viene dado en las aplicaciones de venta en línea, donde separan sus productos por categorías, ropa, tecnología, calzado, videojuegos y demás, pero esta separación es muy particular para un sitio web de ventas, si tratamos de realizar una separación más general, encontraremos productos, ofertas, listas, entre otras. Comúnmente estas separaciones son una particularidad más en un desarrollo de software, pantallas de carga, pantalla de inicio, servicio al cliente, términos y condiciones y entre muchas más secciones definen una aplicación web.

La arquitectura de la información, aparte de separar la información, también la organiza, es así, que existen herramientas como los mapas del sitio, los cuales nos ayudan a mostrar la relación entre los elementos de información.

Mapa de Sitio

Es importante empezar el diseño de las páginas o pantallas de la aplicación que se está construyendo.

Como ya se ha mencionado, la idea general del proyecto es construir un compilador, pero ¿qué significa esto en términos de visualización? ¿Qué verá el usuario al abrir nuestra aplicación? ¿El usuario será capaz de usar el sitio web desde su tableta, desde su teléfono?

Responsividad

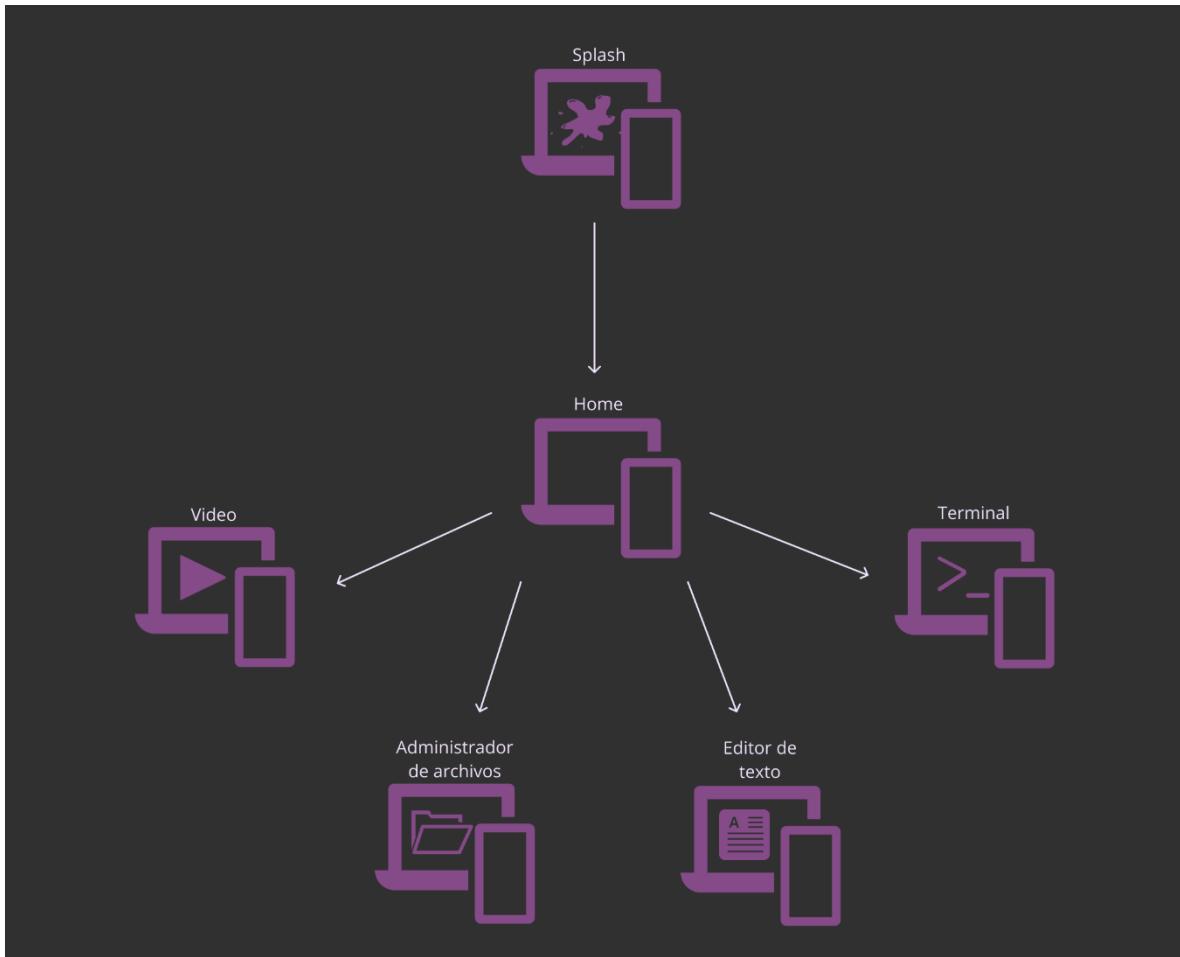
Uno de los conceptos más importantes en el desarrollo y diseño web responsivo es el de “mobile first”, el cual nos dice que debemos priorizar el diseño y desarrollo para dispositivos móviles por encima de los dispositivos de escritorio. De este modo garantizamos el correcto uso del sitio para todos los usuarios, sin importar desde qué dispositivo accedan.

La arquitectura responsiva es una disciplina que busca decidir cómo se administran los espacios de los diferentes elementos que conforman una página web, haciendo que estos respondan a la interacción y presencia del usuario.

En este sentido, se ha pensado dividir el sitio en diferentes pantallas que formarán la aplicación general, de modo que cada pantalla sea presentada como una página única sin obcecarn al usuario con un diseño burdo. En tal caso y haciendo uso del “mobile first”, la distribución de pantallas será de la siguiente manera:

- Una pantalla splash que muestre el branding o los logos, alguna pequeña frase que introduzca al usuario y de ser necesario algún botón.
- La pantalla de inicio la cual tendrá principalmente botones que redirigirán a las subpantallas consecuentes. Aquí es donde se le ubicará al usuario en todo momento.
- La pantalla de edición de texto. Esta será por defecto la subpantalla o pestaña que se mostrará en la pantalla de inicio; aquí es donde el usuario será capaz de escribir el código en lenguaje BrightScript para después ser procesado en el servidor. Esta sección de la aplicación tendrá sus respectivos botones de ser necesario.
- La pantalla de consola o terminal. En esta pestaña o subpantalla de inicio, el usuario será capaz de visualizar los registros (logs) del código que haya sido ejecutado. Esta sección también podrá tener botones de ser necesario.
- La pantalla de administración de archivos. En esta pestaña, el usuario será capaz de visualizar los archivos del proyecto en el cual está trabajando en ese momento. Es importante señalar, que, aunque no sea parte del diseño sino de la funcionalidad, no se espera que los usuarios puedan guardar sus proyectos, serán sesiones únicas y lo que no guarden por su cuenta lo perderán. Cada ícono presentado como un archivo o carpeta deberá ser seleccionable.
- La pantalla de reproductor de video será una pestaña/subpantalla opcional. Su alcance ha sido definido previamente en los objetivos. Se creará el diseño y su implementación, pero es un tema bastante complejo en funcionalidad, debido a que se requeriría un dispositivo Roku por usuario.

Finalmente, teniendo en cuenta esto, presentamos el mapa del sitio:



Como se ha dicho previamente, se pensó, principalmente en el uso del sitio en dispositivos móviles y dispositivos de escritorio. Para el sitio de escritorio el diseño es más sencillo, pensando que el espacio es mayor y la cantidad de elementos que el usuario podrá ver será más que suficiente.

Wireframes y wireflows

En el desarrollo de software existen 3 términos que comúnmente son usados para describir lo mismo, aunque ciertamente no lo son, estos términos son wireframe, mockup y prototype.

El error de usar estos 3 términos indistintamente viene de que en realidad estos 3 términos forman un proceso único en el diseño del software. Este proceso es la presentación física de un producto muy básico solo para dar a entender al usuario cómo funcionará y se verá el producto final.

Así, pues, este proceso de modelado o prototipado empezaría con los wireframes.

Wireframes

Los wireframes son generalmente un bosquejo de cómo se piensa que sería el diseño final del software, aquí los diseñadores empiezan a imaginar la aplicación, los lugares que cada componente y elemento ocuparía, su funcionamiento, sus flujos, animaciones y demás. Estas son ideas muy vagas ya que generalmente este proceso es realizado en etapas muy tempranas del proyecto.

Mockups

Los mockups normalmente continúan el trabajo desde donde quedaron los wireframes, estos se enfocan más en mostrar un diseño más realista, se quitan cosas que en el wireframe no son realistas o están muy por encima del objetivo definido del desarrollo.

En este proceso se agregan nuevos detalles visuales, elementos como colores, imágenes, fuentes de texto, tipografías y demás cosas que terminan por dar personalidad al software. Por lo que este proceso tiene un impacto más visual que funcional.

Prototypes

El prototipo es el paso final en este proceso. Aquí se muestra una simulación de la aplicación casi fidedigna al producto final, en este paso tratamos de reflejar la funcionalidad completa que tendría el producto.

En este paso también se busca estudiar el comportamiento del usuario con la aplicación, se busca ver si el prototipo es intuitivo.

Una vez conociendo los tres términos del proceso de prototipado, es importante recalcar que uno depende del otro, deben crearse en el orden dado y se trabajan durante todo el ciclo de vida del software.

Wireflows

Este término refiere a los flujos que existan dentro del prototipado, generalmente representan interacciones. Este elemento visual es muy útil y habitualmente se presenta como conexiones entre prototipos.

Maquetación

Para la creación de los diseños, se usó la aplicación en línea Figma. Todos los íconos y diseños fueron hechos completamente desde cero haciendo uso de las herramientas en Figma.

En cuanto a las medidas generales, se usaron las siguientes:

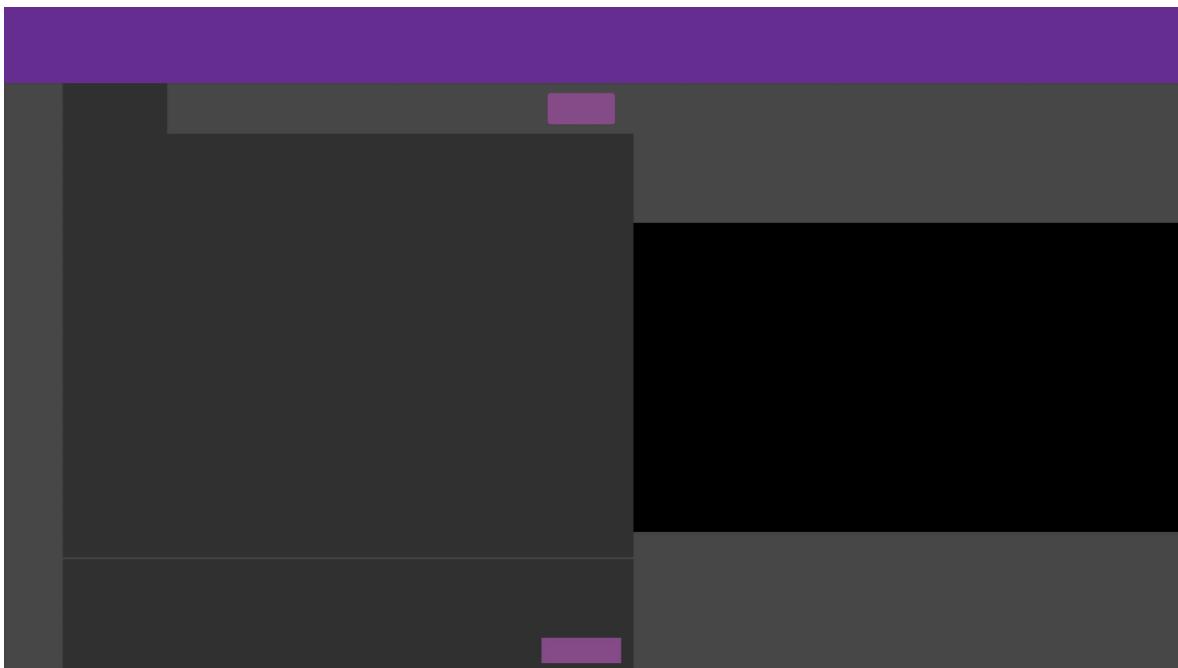
- Dispositivos móviles: 360x800
- Tabletas: 744x1133
- Ordenadores: 1440x1024

Basándonos en lo descrito en la arquitectura de la información y el flujo general de las pantallas que la aplicación tendrá, definimos cada flujo (wireflow), dando a entender que elemento del sitio web será el que lleve al usuario al siguiente paso del flujo.

A continuación, se presentarán los wireframes o maquetas del cómo se verá la aplicación para dispositivos móviles, tabletas y de escritorio.

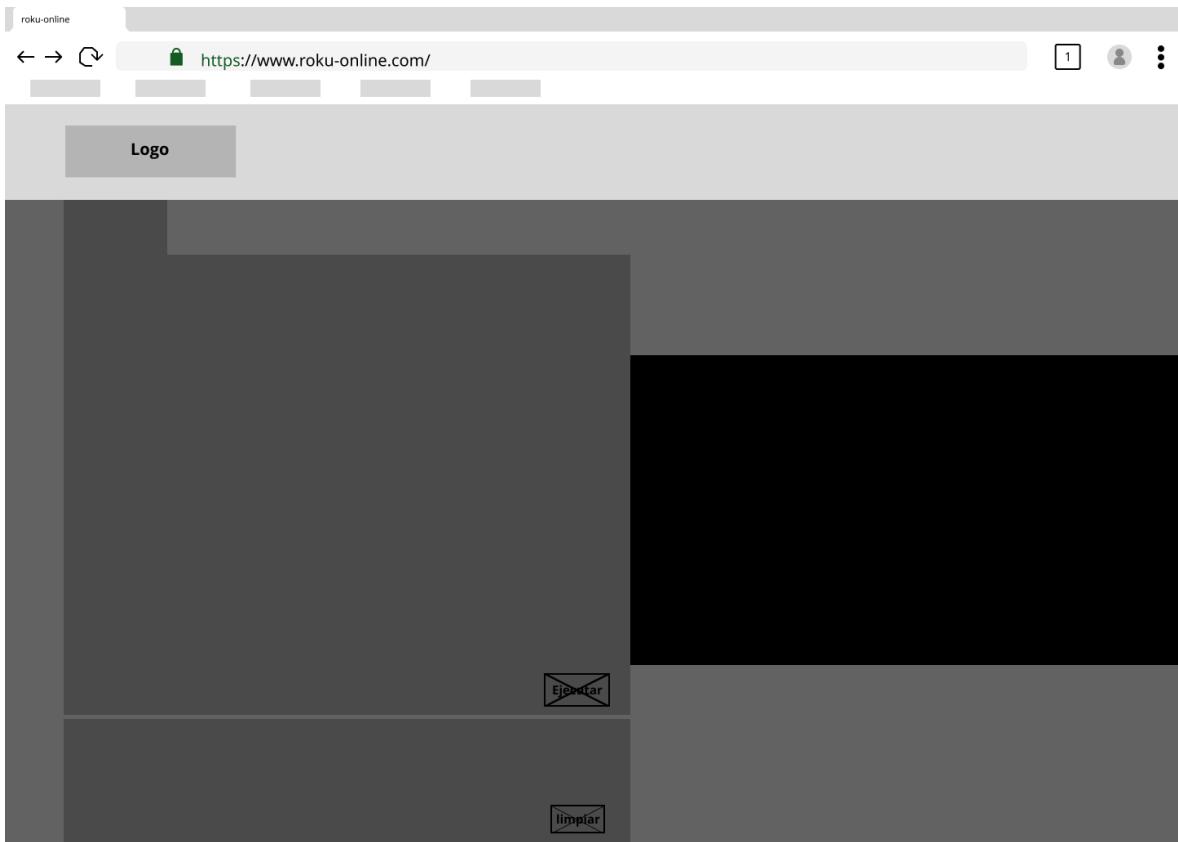
Primer wireframe y mockup





Segundo wireframe

Este es el segundo wireframe del cuál basaremos los wireframes finales.



12:05 PM



🔒 <https://www.roku-online.com/>



Archives



Editor



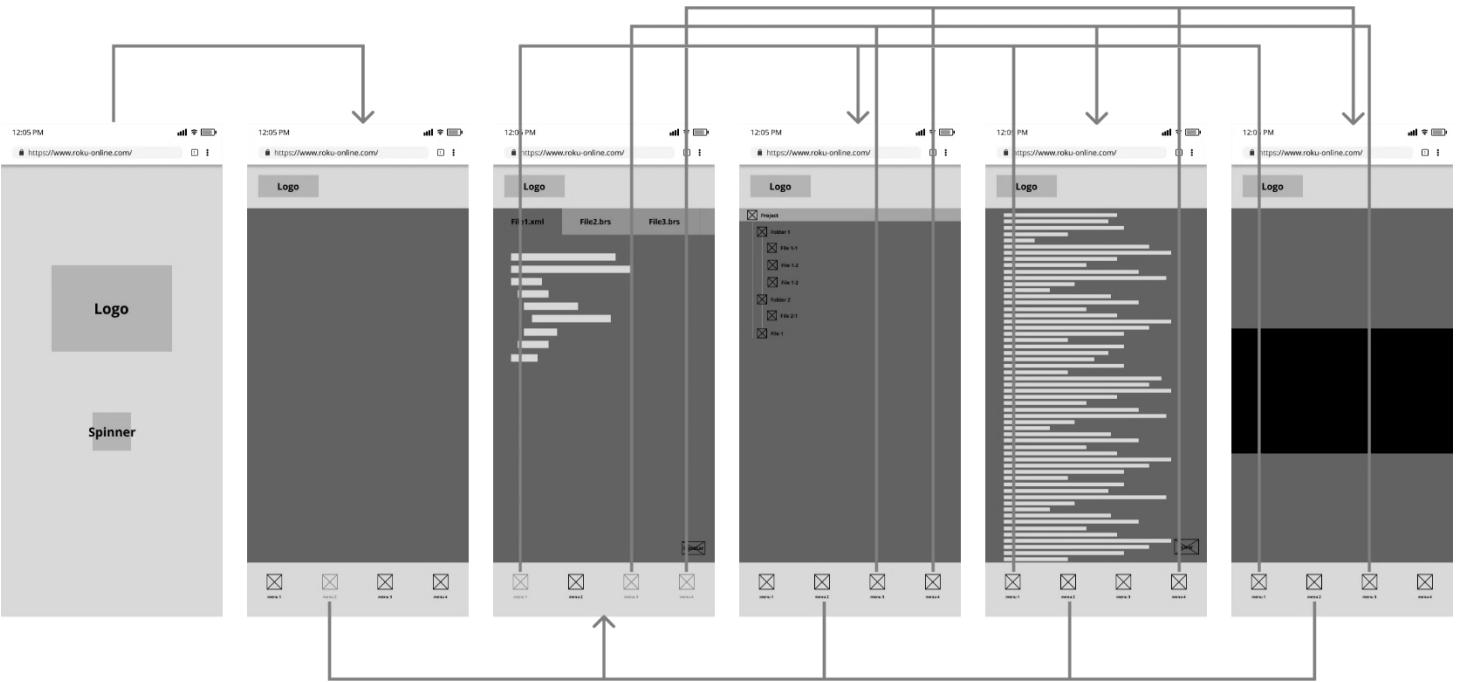
Terminal



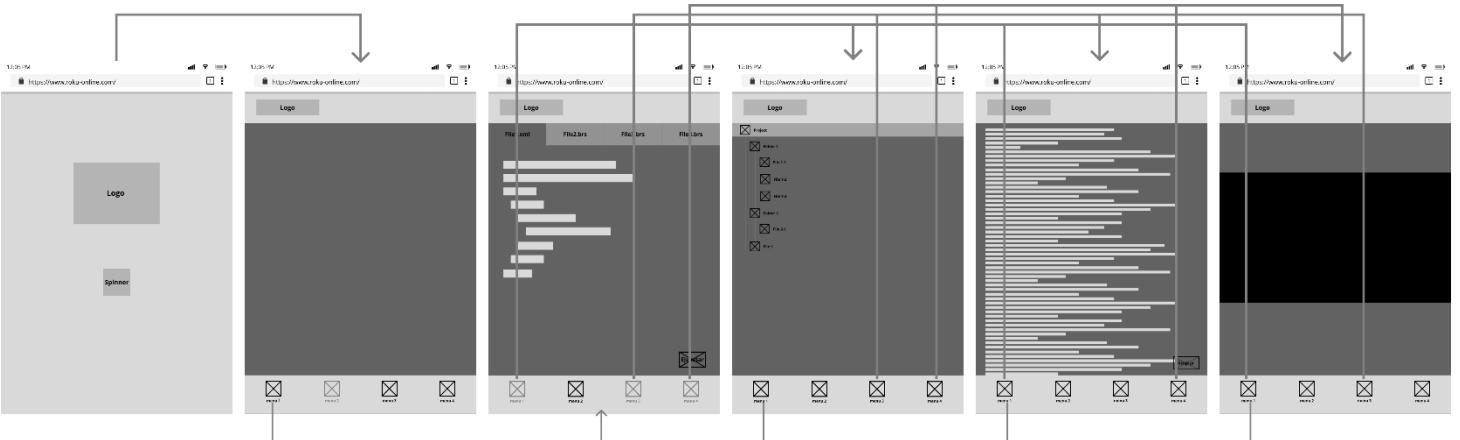
Video

Wireframes y wireflows

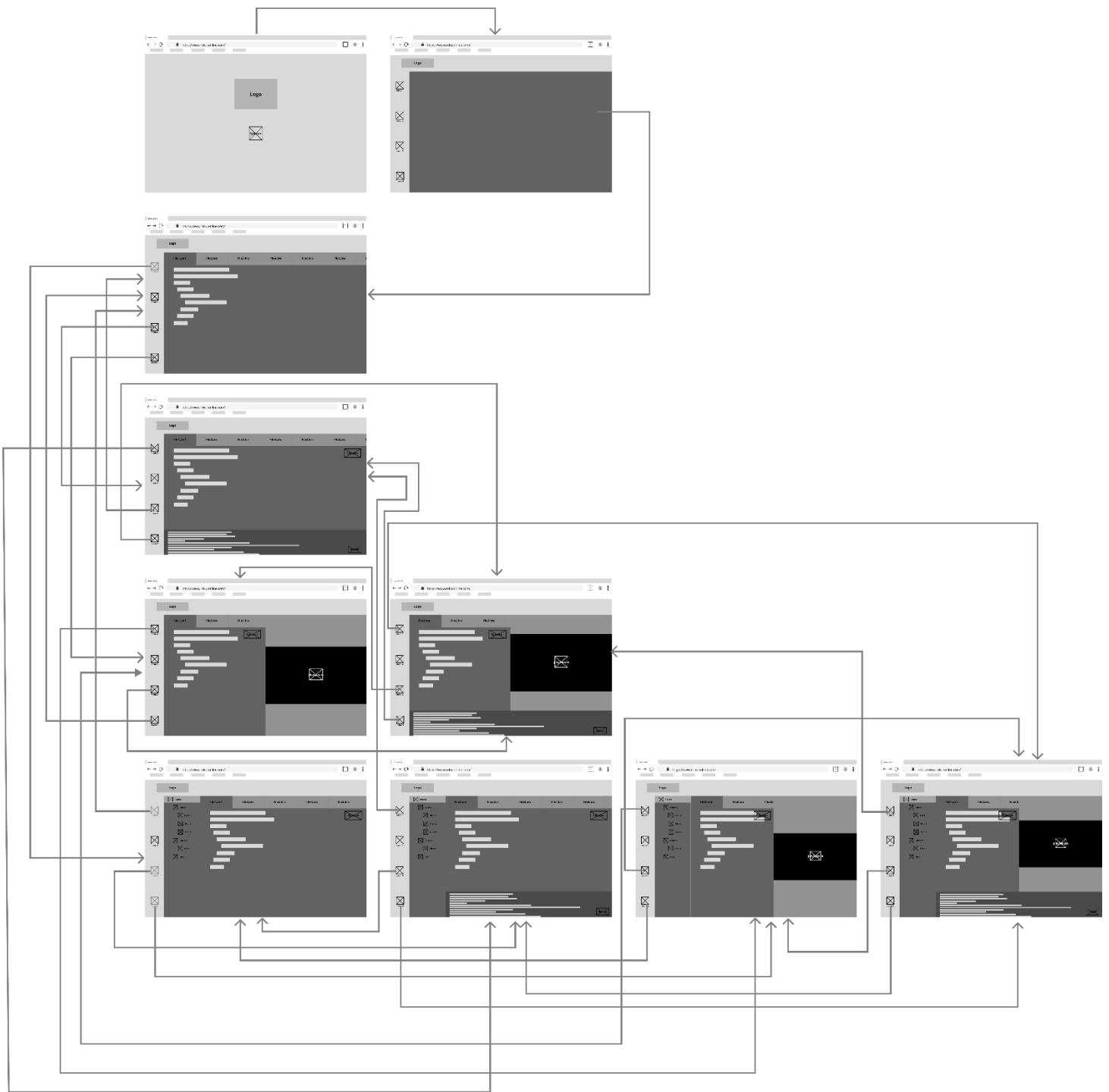
Dispositivos móviles



Tablets



Ordenador

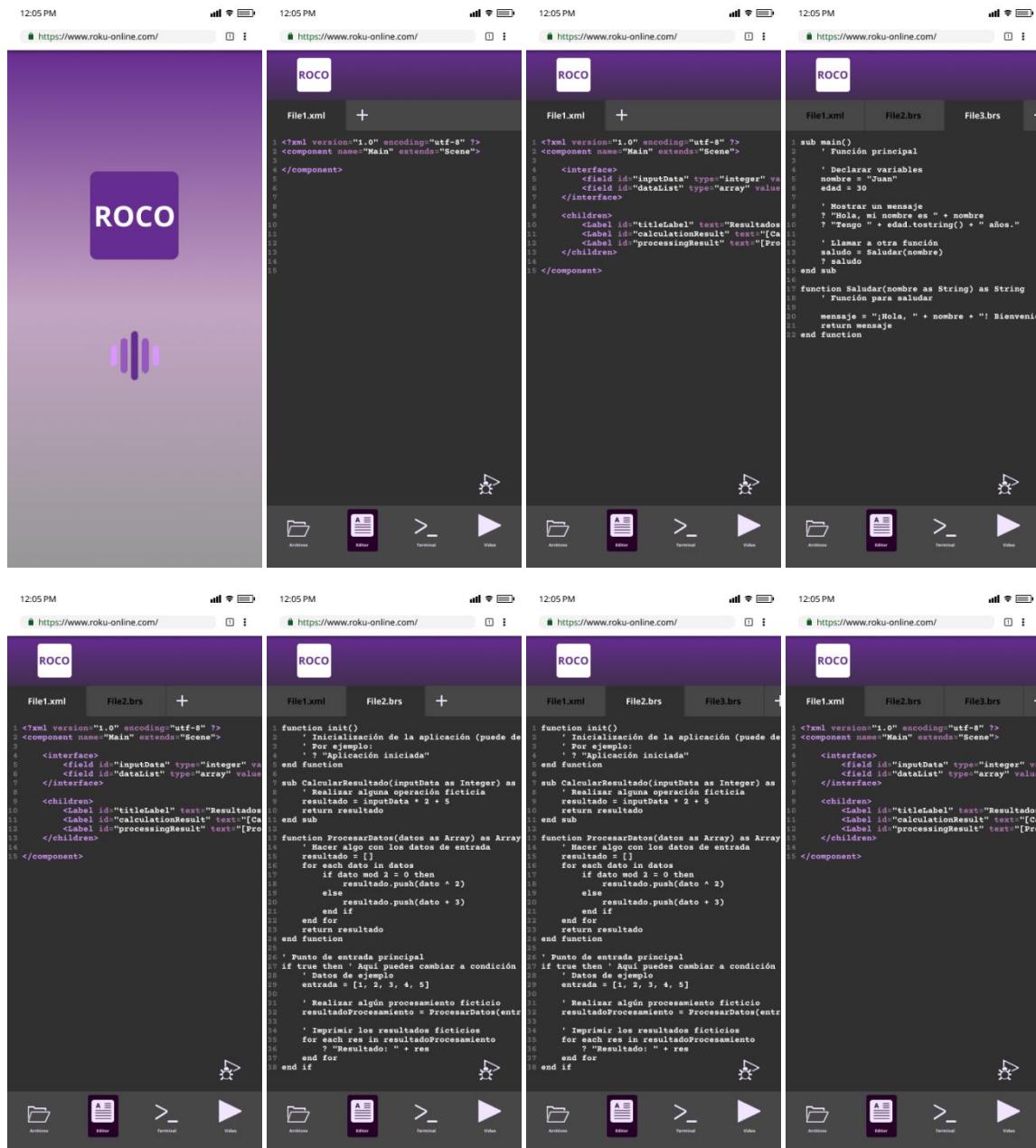


Se dejará el enlace al diseño en el sitio de Figma, en donde será más fácil apreciar a detalle los elementos y sus respectivos flujos.

Prototipos

A continuación, mostraremos un par de los prototipos finales de la aplicación. Para ser capaces de interactuar con los prototipos, la herramienta Figma tiene funcionalidades de realizar animaciones e interacciones, por lo que se adjuntarán los respectivos enlaces a los prototipos en la sección de referencias.

Dispositivos móviles



The image displays four screenshots of the ROCO mobile application interface, showing code snippets in different panels:

- Panel 1 (File1.xml):**

```
<?xml version="1.0" encoding="utf-8" ?>
<component name="Main" extends="Scene">
</component>
```
- Panel 2 (File1.xml):**

```
<?xml version="1.0" encoding="utf-8" ?>
<component name="Main" extends="Scene">
<interface>
<field id="inputData" type="integer" value="0" />
<field id="dataList" type="array" value="[]"/>
</interface>
<children>
<label id="titleLabel" text="Resultado" />
<label id="calculationResult" text="["/gt;>
<label id="processingResult" text="["/gt;>
</children>
</component>
```
- Panel 3 (File1.xml):**

```
sub main()
    ' Función principal
    ' Declarar variables
    nombre = "Juan"
    edad = 30

    ' Mostrar un mensaje
    ? "Hola, mi nombre es " + nombre
    ? "Tengo " + edad.tostring() + " años."
    llamar a otra función
    saludar = Saludar(nombre)
    saludar
end sub

function Saludar(nombre as String) as String
    ' Función para saludar
    mensaje = "Hola, " + nombre + "! Bienvenido"
    return mensaje
end function
```
- Panel 4 (File1.xml):**

```
function init()
    ' Inicialización de la aplicación (puede de
    ' Por ejemplo:
    ' - Inicialización iniciada'
end function

sub CalcularResultado(inputData as Integer) as
    ' Realizar alguna operación ficticia
    resultado = inputData * 2 + 5
    return resultado
end sub

function ProcesarDatos(datos as Array) as Array
    ' Procesar datos
    resultado = []
    for each dato in datos
        if dato mod 2 = 0 then
            resultado.push(dato * 2)
        else
            resultado.push(dato + 3)
        end if
    end for
    return resultado
end function

' Punto de entrada principal
if true then ' Aquí puedes cambiar a condición
    ' Datos de ejemplo
    entrada = [1, 2, 3, 4, 5]
    ' Realizar algún procesamiento ficticio
    resultadoProcesamiento = ProcesarDatos(entrada)
    ' Imprimir los resultados ficticios
    for each res in resultadoProcesamiento
        ? "Resultado: " + res
    end for
end if
```
- Panel 5 (File2.brs):**

```
<?xml version="1.0" encoding="utf-8" ?>
<component name="Main" extends="Scene">
<interface>
<field id="inputData" type="integer" value="0" />
<field id="dataList" type="array" value="[]"/>
</interface>
<children>
<label id="titleLabel" text="Resultado" />
<label id="calculationResult" text="["/gt;>
<label id="processingResult" text="["/gt;>
</children>
</component>
```
- Panel 6 (File2.brs):**

```
function init()
    ' Inicialización de la aplicación (puede de
    ' Por ejemplo:
    ' - Inicialización iniciada'
end function

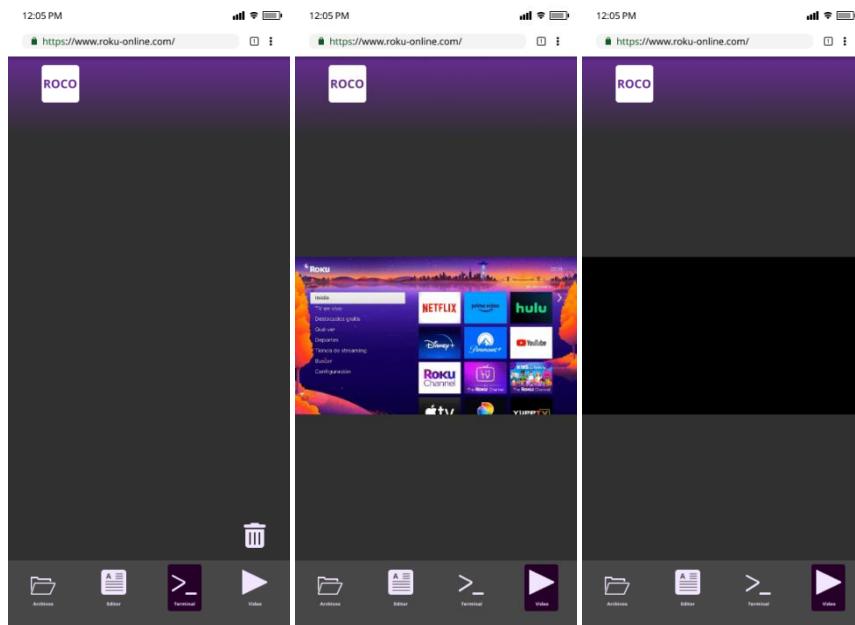
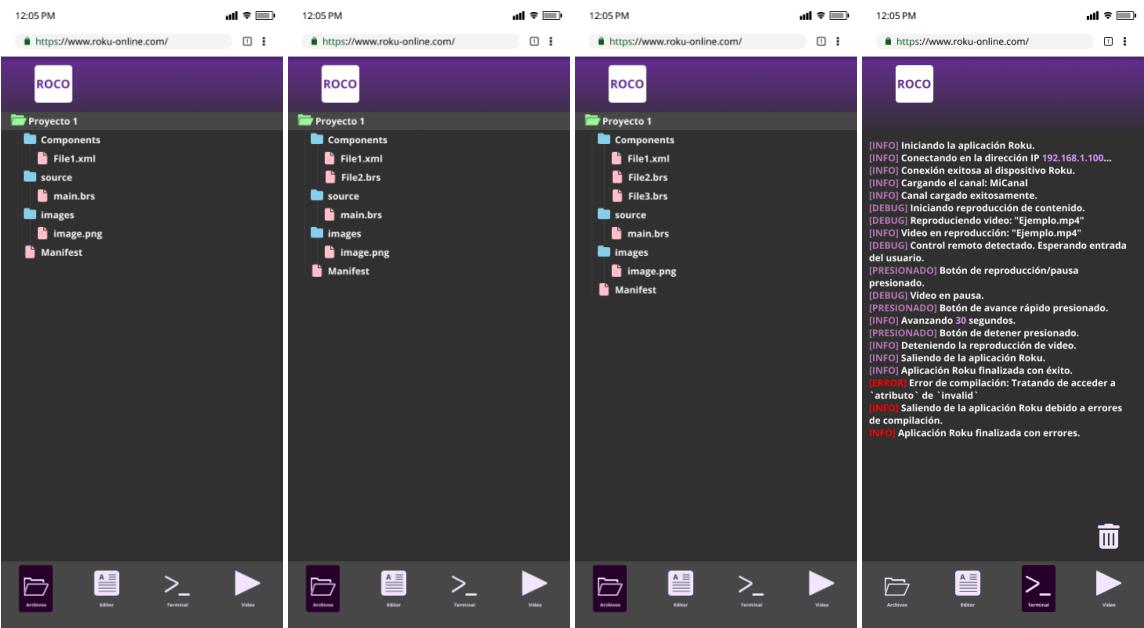
sub CalcularResultado(inputData as Integer) as
    ' Realizar alguna operación ficticia
    resultado = inputData * 2 + 5
    return resultado
end sub

function ProcesarDatos(datos as Array) as Array
    ' Procesar datos
    resultado = []
    for each dato in datos
        if dato mod 2 = 0 then
            resultado.push(dato * 2)
        else
            resultado.push(dato + 3)
        end if
    end for
    return resultado
end function

' Punto de entrada principal
if true then ' Aquí puedes cambiar a condición
    ' Datos de ejemplo
    entrada = [1, 2, 3, 4, 5]
    ' Realizar algún procesamiento ficticio
    resultadoProcesamiento = ProcesarDatos(entrada)
    ' Imprimir los resultados ficticios
    for each res in resultadoProcesamiento
        ? "Resultado: " + res
    end for
end if
```
- Panel 7 (File3.brs):**

```
<?xml version="1.0" encoding="utf-8" ?>
<component name="Main" extends="Scene">
<interface>
<field id="inputData" type="integer" value="0" />
<field id="dataList" type="array" value="[]"/>
</interface>
<children>
<label id="titleLabel" text="Resultado" />
<label id="calculationResult" text="["/gt;>
<label id="processingResult" text="["/gt;>
</children>
</component>
```
- Panel 8 (File3.brs):**

```
<?xml version="1.0" encoding="utf-8" ?>
<component name="Main" extends="Scene">
<interface>
<field id="inputData" type="integer" value="0" />
<field id="dataList" type="array" value="[]"/>
</interface>
<children>
<label id=" titleLabel" text="Resultado" />
<label id=" calculationResult" text="["/gt;>
<label id=" processingResult" text="["/gt;>
</children>
</component>
```



Tablets

The image displays a 4x3 grid of screenshots from the ROCO development environment, showing its interface and functionality across multiple tablet devices.

- Row 1:** Shows the ROCO logo and interface on three different tablet screens. The rightmost screen shows a code editor with XML code for a component named "Main".
- Row 2:** Shows the interface with a file list containing "File1.xml", "File2.brs", and "File3.xml". The rightmost screen shows the XML code for "File3.xml".
- Row 3:** Shows the interface with a file list containing "File1.xml", "File2.brs", "File3.xml", and "File4.brs". The rightmost screen shows the XML code for "File4.brs".
- Row 4:** Shows the interface with a file list containing "File1.xml", "File2.brs", "File3.xml", and "File4.brs". The rightmost screen shows the XML code for "File4.brs".

Code Examples (Rightmost Screenshots):

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <component name="Main" extends="Scene">
3
4   <interface>
5     <field id="inputData" type="integer" value="10" />
6     <field id="dataArray" type="array" values="[1, 2, 3]" />
7   </interface>
8
9   <children>
10    <label id="titleLabel" text="Resultados ficticios:"/>
11    <label id="calculationResult" text="CalcularResultado" />
12    <label id="processingResult" text="ProcesarDatos(d)" />
13  </children>
14
15 </component>
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <component name="Main" extends="Scene">
3
4   <interface>
5     <field id="inputData" type="integer" value="10" />
6     <field id="dataArray" type="array" values="[1, 2, 3]" />
7   </interface>
8
9   <children>
10    <label id="titleLabel" text="Resultados ficticios:"/>
11    <label id="calculationResult" text="CalcularResultado" />
12    <label id="processingResult" text="ProcesarDatos(d)" />
13  </children>
14
15 </component>
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <component name="Main" extends="Scene">
3
4   <interface>
5     <field id="inputData" type="integer" value="10" />
6     <field id="dataArray" type="array" values="[1, 2, 3]" />
7   </interface>
8
9   <children>
10    <label id="titleLabel" text="Resultados ficticios:"/>
11    <label id="calculationResult" text="CalcularResultado" />
12    <label id="processingResult" text="ProcesarDatos(d)" />
13  </children>
14
15 </component>
```

```

1 function init()
2   ' Inicialización de la aplicación (puede dejarla vacía o con algún ejemplo)
3   ' Por ejemplo:
4   ' ? "Aplicación iniciada"
5 end function

6 sub CalcularResultado(datos as Integer) as Integer
7   ' Realizar alguna operación ficticia
8   resultado = inputData * 2 + 5
9   return resultado
10 end sub

11 function ProcesarDatos(datos as Array) as Array
12   ' Hacer algo con los datos de entrada
13   resultado = []
14   for each dato in datos
15     if dato mod 2 = 0 then
16       resultado.push(dato ^ 2)
17     else
18       resultado.push(dato + 3)
19     end if
20   end for
21   return resultado
22 end function

23 ' Punto de entrada principal
24 if true then ' Aquí puedes cambiar a condición adecuada para el punto de ejecución
25   ' Ejemplo de datos de prueba
26   entrada = [1, 2, 3, 4, 5]
27
28   ' Realizar algún procesamiento ficticio
29   resultadoProcesamiento = ProcesarDatos(entrada)
30
31   ' Realizar algún procesamiento ficticio
32   resultadoProcesamiento = ProcesarDatos(entrada)
33
```

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <component name="Main" extends="Scene">
3   <interface>
4     <function name="onContentChanged" />
5   </interface>
6   <children>
7     <Label id="titleLabel" text="ítulo" translation="1"/>
8   </children>
9 </component>
10
11
12
13
14
15

```

```

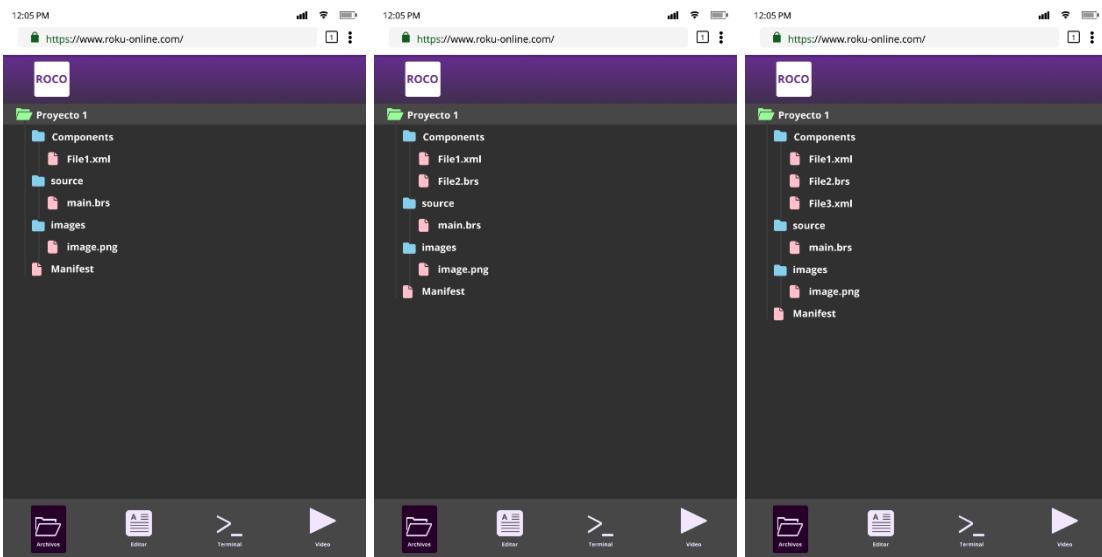
1 <?xml version="1.0" encoding="utf-8" ?>
2 <component name="Main" extends="Scene">
3   <interface>
4     <function name="onContentChanged" />
5   </interface>
6   <children>
7     <Label id="titleLabel" text="ítulo" translation="1"/>
8   </children>
9 </component>
10
11
12
13
14
15

```

```

1 sub init()
2   ' Función de inicialización
3   ' La aplicación se ha iniciado.
4
5   ' Configurar el reproductor de video
6   player = createObject("roVideoPlayer")
7   content = createObject("roTextContent")
8   StreamURL: "https://www.ejemplo.com/video1.m3u8"
9   content.StreamFormat: "hls"
10  StreamURL: "https://www.ejemplo.com/video1.m3u8"
11  Player.SetContent(content)
12 end sub
13
14 function onContentChanged()
15   ' Manejar eventos cuando el contenido cambia
16   ' El contenido ha cambiado.
17
18   ' Realizar acciones adicionales
19   ' Por ejemplo, actualizar la interfaz de usuario
20 end function

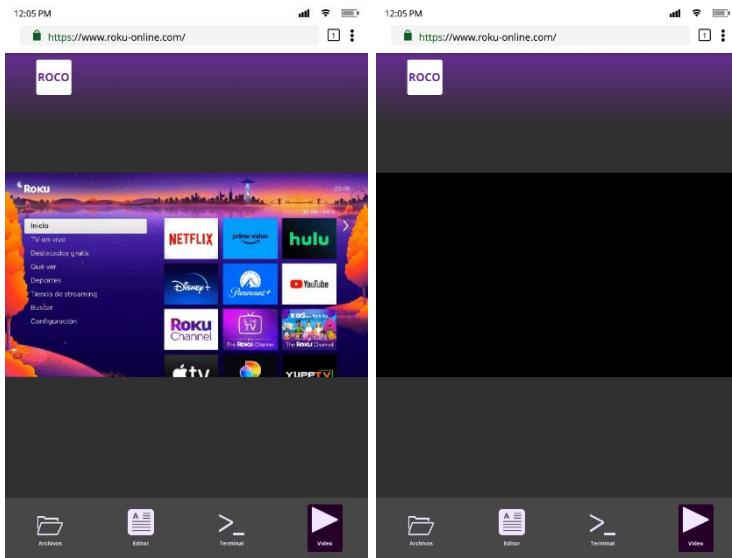
```



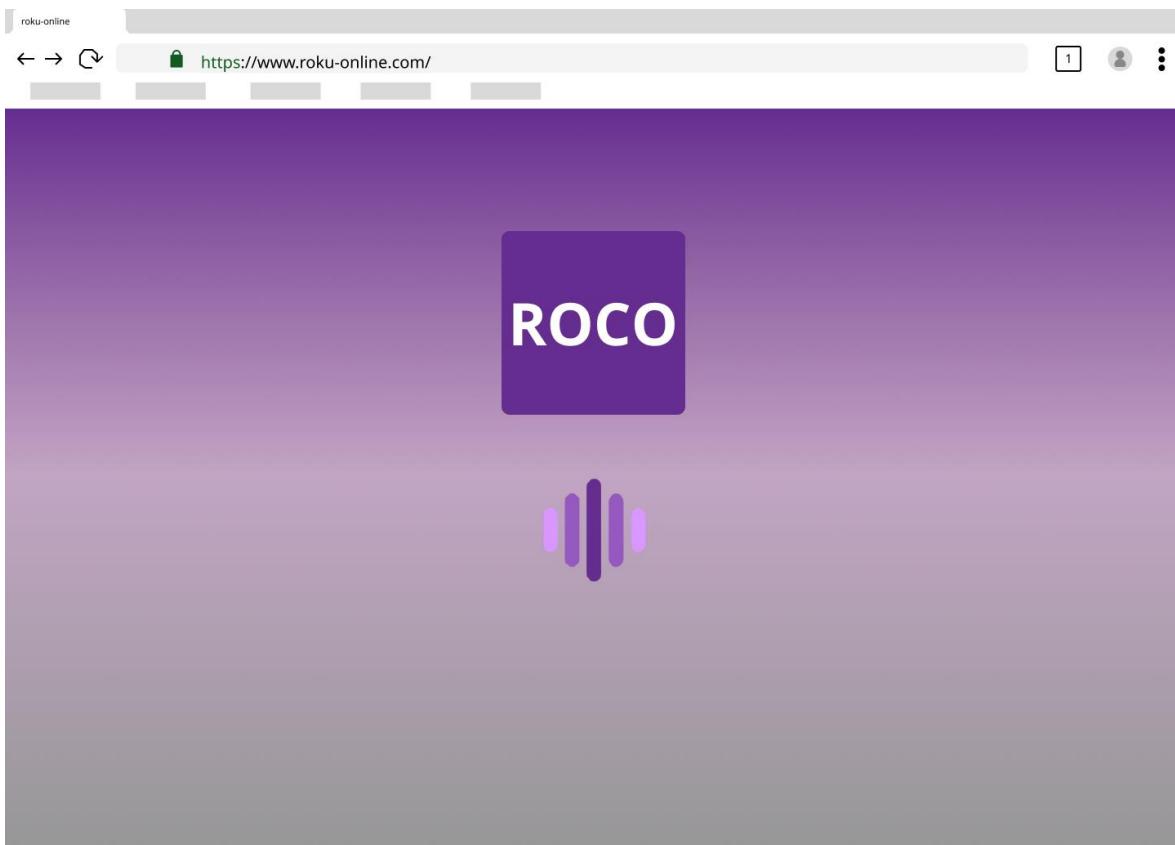
```

[INFO] Iniciando la aplicación Roku.
[INFO] Conectando en la dirección IP 192.168.1.100...
[INFO] Conexión exitosa al dispositivo Roku.
[INFO] Cargando el canal: MiCanal.
[INFO] Cálculo de los elementos.
[DEBUG] Iniciando la reproducción de contenido.
[DEBUG] Reproduciendo video: "Ejemplo.mp4"
[INFO] Video en reproducción: "Ejemplo.mp4"
[DEBUG] Control remoto detectado. Esperando entrada del usuario.
[PRESIONADO] Botón de reproducción/pausa presionado.
[DEBUG] Video en pausa.
[PRESIONADO] Botón de avance rápido presionado.
[INFO] Avanzando 30 segundos.
[PRESIONADO] Botón de detener presionado.
[INFO] Deteniendo la reproducción de video.
[INFO] Saliendo de la aplicación Roku.
[INFO] Aplicación Roku finalizada con éxito.
[ERROR] Error de compilación: Tratando de acceder a 'atributo' de 'invalid'.
[INFO] Saliendo de la aplicación Roku debido a errores de compilación.
[INFO] Aplicación Roku finalizada con errores.

```



Ordenador



roku-online

https://www.roku-online.com/

ROCO

File1.xml File2.brs +

Archivos Editor Terminal Video

```
function init()
    ' Inicialización de la aplicación (puede dejarla vacía o agregar código de inicio)
    ' Por ejemplo:
    ' ? "Aplicación iniciada"
end function

sub CalcularResultado(inputData as Integer) as Integer
    ' Realizar alguna operación ficticia
    resultado = inputData * 2 + 5
    return resultado
end sub

function ProcesarDatos(datos as Array) as Array
    ' Hacer algo con los datos de entrada
    resultado = []
    for each dato in datos
        if dato mod 2 = 0 then
            resultado.push(dato ^ 2)
        else
            resultado.push(dato + 3)
        end if
    end for
    return resultado
end function

' Punto de entrada principal
if true then ' Aquí puedes cambiar a condición adecuada para tu aplicación
    ' Datos de ejemplo

```

roku-online

https://www.roku-online.com/

ROCO

Proyecto 1 File1.xml File2.brs +

Archivos Editor Terminal Video

```
<?xml version="1.0" encoding="utf-8" ?>
<component name="Main" extends="Scene">

    <interface>
        <field id="inputData" type="integer" value="10" />
        <field id="dataList" type="array" value="[1, 2, 3, 4, 5]" />
    </interface>

    <children>
        <Label id="titleLabel" text="Resultados ficticios:" translation="[10, 1, 1, 1]" />
        <Label id="calculationResult" text="[CalcularResultado(inputData)]" translation="[10, 1, 1, 1]" />
        <Label id="processingResult" text="[ProcesarDatos(dataList)]" translation="[10, 1, 1, 1]" />
    </children>
</component>
```

roku-online https://www.roku-online.com/ 1 🌐

ROCO

Proyecto 1

File1.xml File2.brs +

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <component name="Main" extends="Scene">
3
4     <interface>
5         <field id="inputData" type="integer" value="10" />
6         <field id="dataList" type="array" value="[1, 2, 3, 4, 5]" />
7     </interface>
8
9     <children>
10        <Label id="titleLabel" text="Resultados ficticios:" translation="[10, 1]
11        <Label id="calculationResult" text="[CalcularResultado(inputData)]" tra
12        <Label id="processingResult" text="[ProcesarDatos(dataList)]" translati
13    </children>
14
15 </component>
```

Archivos Editor Terminal Video

roku-online https://www.roku-online.com/ 1 🌐

ROCO

Proyecto 1

File1.xml File2.brs +

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <component name="Main" extends="Scene">
3
4     <interface>
5         <field id="inputData" type="integer" value="10" />
6         <field id="dataList" type="array" value="[1, 2, 3, 4, 5]" />
7     </interface>
8
9     <children>
10        <Label id="titleLabel" text="Resultados ficticios:" translation="[10, 1]
11        <Label id="calculationResult" text="[CalcularResultado(inputData)]" tra
12        <Label id="processingResult" text="[ProcesarDatos(dataList)]" translati
13    </children>
14
15 </component>
```

[INFO] Iniciando la aplicación Roku.
[INFO] Conectando en la dirección IP 192.168.1.100...
[INFO] Conexión exitosa al dispositivo Roku.
[INFO] Cargando el canal: MiCanal
[INFO] Canal cargado exitosamente.

Archivos Editor Terminal Video

roku-online https://www.roku-online.com/ 1 1 ···

ROCO

Proyecto 1

File1.xml File2.brs +

Components

- File1.xml
- File2.brs

source

- main.brs

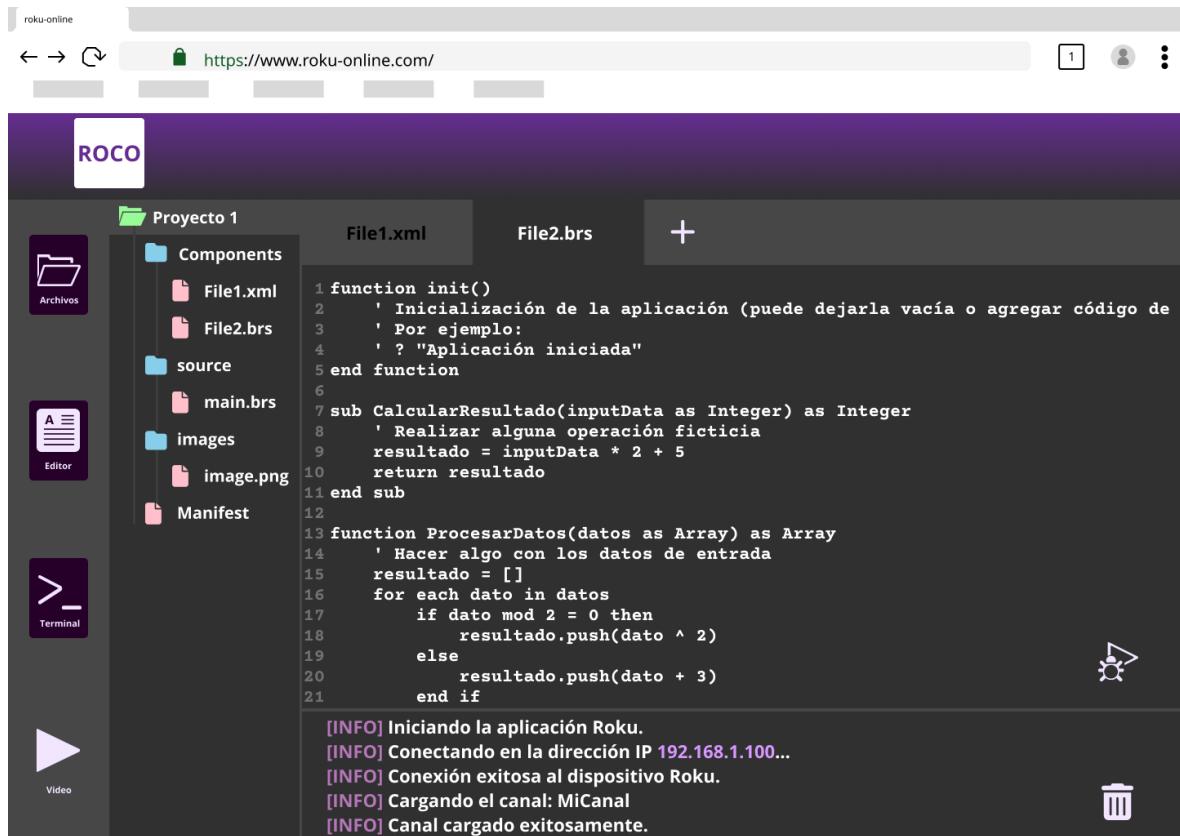
images

- image.png

Manifest

```
1 function init()
2     ' Inicialización de la aplicación (puede dejarla vacía o agregar código de ejemplo)
3     ' ? "Aplicación iniciada"
4 end function
5
6 sub CalcularResultado(inputData as Integer) as Integer
7     ' Realizar alguna operación ficticia
8     resultado = inputData * 2 + 5
9     return resultado
10 end sub
11
12 function ProcesarDatos(datos as Array) as Array
13     ' Hacer algo con los datos de entrada
14     resultado = []
15     for each dato in datos
16         if dato mod 2 = 0 then
17             resultado.push(dato ^ 2)
18         else
19             resultado.push(dato + 3)
20         end if
21     end for
22
23     [INFO] Iniciando la aplicación Roku.
24     [INFO] Conectando en la dirección IP 192.168.1.100...
25     [INFO] Conexión exitosa al dispositivo Roku.
26     [INFO] Cargando el canal: MiCanal
27     [INFO] Canal cargado exitosamente.
```

Archivos Editor Terminal Video



roku-online https://www.roku-online.com/ 1 1 ···

ROCO

Proyecto 1

File1.xml File2.brs +

Components

- File1.xml
- File2.brs

source

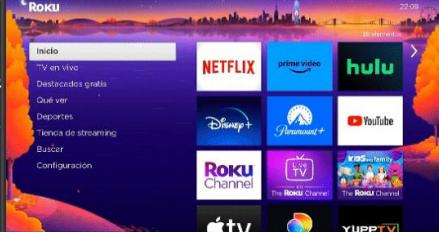
- main.brs

images

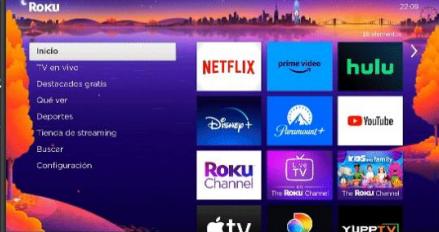
- image.png

Manifest

```
1 <?xml version="1.0" encoding="utf-8" ?
2 <component name="Main" extends="Scene"
3
4     <interface>
5         <field id="inputData" type="int"/>
6         <field id="dataList" type="array<int>"/>
7     </interface>
8
9     <children>
10        <Label id="titleLabel" text="Roku"/>
11        <Label id="calculationResult" text="0"/>
12        <Label id="processingResult" text="0"/>
13    </children>
14
15 </component>
```



Archivos Editor Terminal Video



roku-online https://www.roku-online.com/ 1 · ·

ROCO

File1.xml File2.brs +

Archivos Editor Terminal Video

```
1 function init()
2     ' Inicialización de la aplicación (puede dejarla vacía o agregar código de inicio)
3     ' Por ejemplo:
4     ' ? "Aplicación iniciada"
5 end function

6
7 sub CalcularResultado(inputData as Integer) as Integer
8     ' Realizar alguna operación ficticia
9     resultado = inputData * 2 + 5
10    return resultado
11 end sub

12
13 function ProcesarDatos(datos as Array) as Array
14     ' Hacer algo con los datos de entrada
15     resultado = []
16     for each dato in datos
17         if dato mod 2 = 0 then
18             resultado.push(dato ^ 2)
19         else
20             resultado.push(dato + 3)
21         end if

```

Terminal >

Video

roku-online https://www.roku-online.com/ 1 · ·

ROCO

File1.xml File2.brs +

Archivos Editor Terminal Video

```
1 function init()
2     ' Inicialización de la aplicación (puede dejarla vacía o agregar código de inicio)
3     ' Por ejemplo:
4     ' ? "Aplicación iniciada"
5 end function

6
7 sub CalcularResultado(inputData as Integer) as Integer
8     ' Realizar alguna operación ficticia
9     resultado = inputData * 2 + 5
10    return resultado
11 end sub

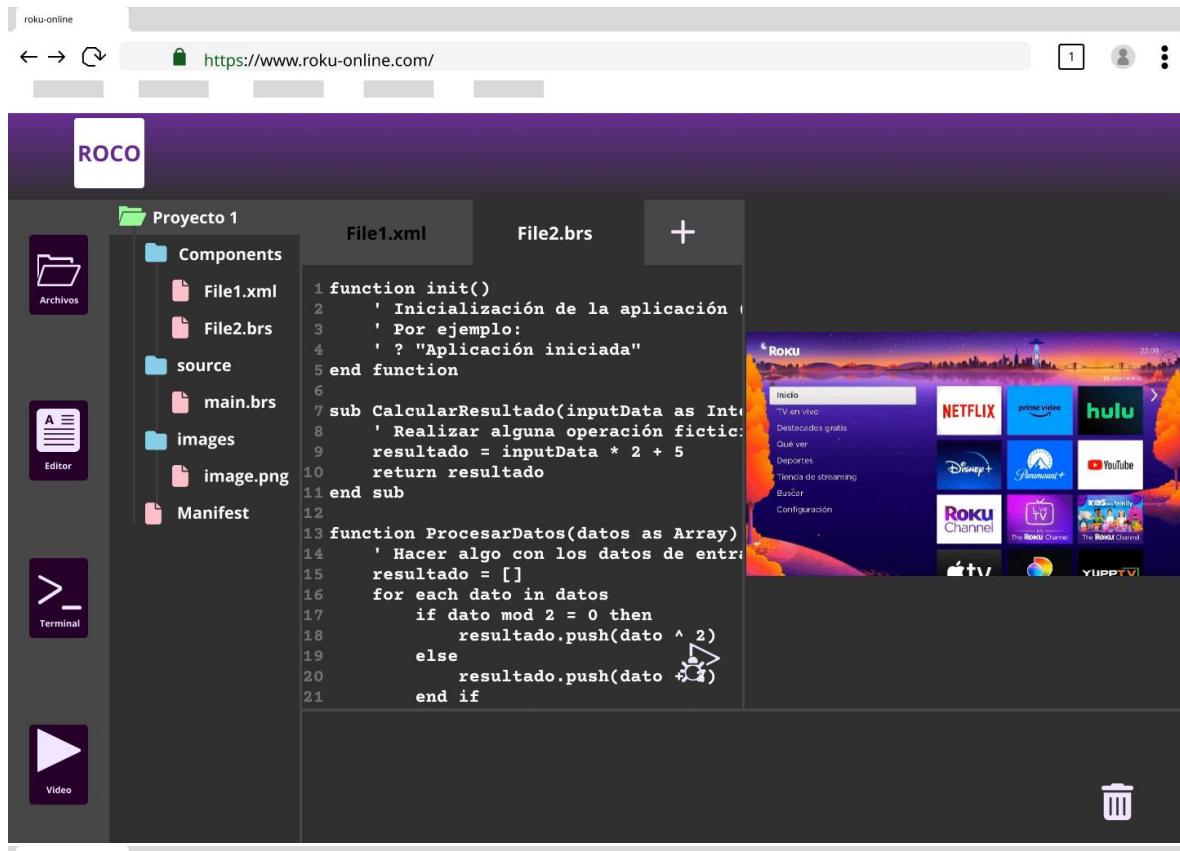
12
13 function ProcesarDatos(datos as Array) as Array
14     ' Hacer algo con los datos de entrada
15     resultado = []
16     for each dato in datos
17         if dato mod 2 = 0 then
18             resultado.push(dato ^ 2)
19         else
20             resultado.push(dato + 3)
21         end if

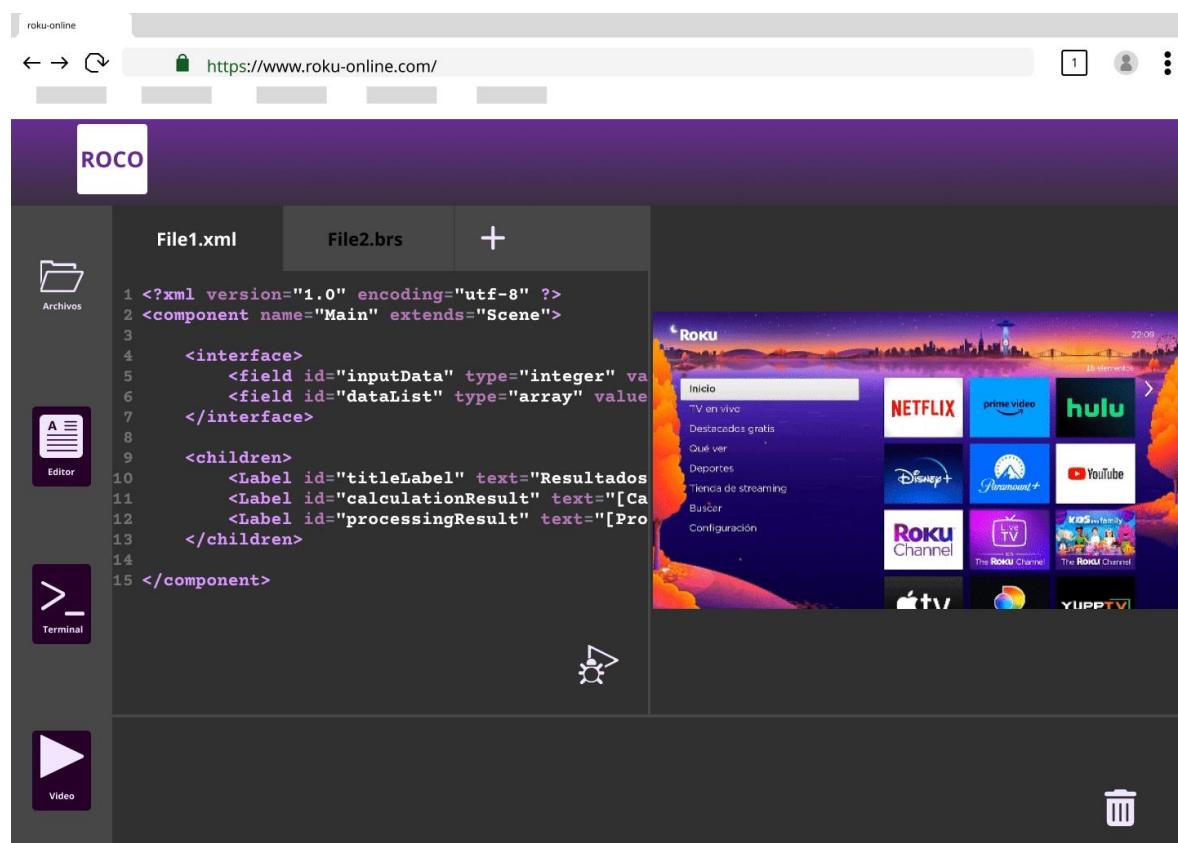
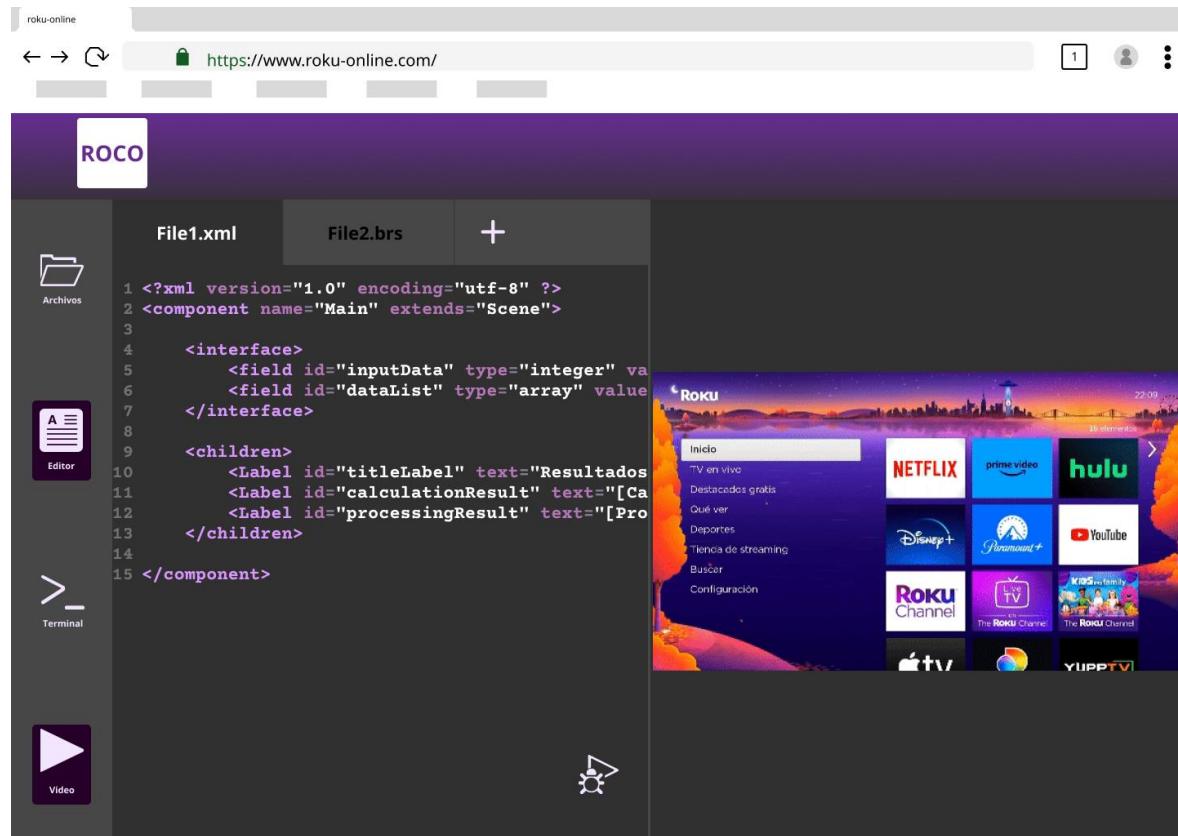
```

[INFO] Iniciando la aplicación Roku.
[INFO] Conectando en la dirección IP 192.168.1.100...
[INFO] Conexión exitosa al dispositivo Roku.
[INFO] Cargando el canal: MiCanal
[INFO] Canal cargado exitosamente.

Terminal >

Video





roku-online

https://www.roku-online.com/

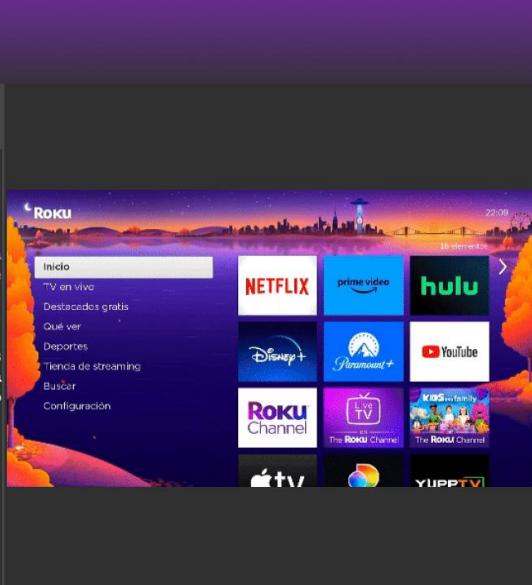
ROCO

File1.xml File1.xml +

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <component name="Main" extends="Scene">
3
4     <interface>
5         <field id="inputData" type="integer" va...
6         <field id="dataList" type="array" value...
7     </interface>
8
9     <children>
10        <Label id="titleLabel" text="Resultados"/>
11        <Label id="calculationResult" text="["Ca...
12        <Label id="processingResult" text="["Pro...
13    </children>
14
15 </component>
```

Archivos Editor Terminal Video

[INFO] Iniciando la aplicación Roku.
[INFO] Conectando en la dirección IP 192.168.1.100...
[INFO] Conexión exitosa al dispositivo Roku.
[INFO] Cargando el canal: MiCanal
[INFO] Canal cargado exitosamente.



roku-online

https://www.roku-online.com/

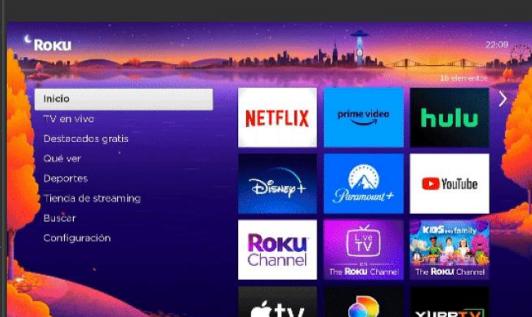
ROCO

File1.xml File2.brs +

```
1 function init()
2     ' Inicialización de la aplicación (puede de...
3     ' Por ejemplo:
4     ' ? "Aplicación iniciada"
5 end function
6
7 sub CalcularResultado(inputData as Integer) as ...
8     ' Realizar alguna operación ficticia
9     resultado = inputData * 2 + 5
10    return resultado
11 end sub
12
13 function ProcesarDatos(datos as Array) as Array
14     ' Hacer algo con los datos de entrada
15     resultado = []
16     for each dato in datos
17         if dato mod 2 = 0 then
18             resultado.push(dato ^ 2)
19         else
20             resultado.push(dato + 3)
21         end if
22     end for
23 end function
```

Archivos Editor Terminal Video

[INFO] Iniciando la aplicación Roku.
[INFO] Conectando en la dirección IP 192.168.1.100...
[INFO] Conexión exitosa al dispositivo Roku.
[INFO] Cargando el canal: MiCanal
[INFO] Canal cargado exitosamente.



Capítulo II: Desarrollo

Introducción al Entorno de Desarrollo:

En el proceso del desarrollo de un software, la etapa justamente de desarrollo es, la etapa más larga, pudiendo durar meses o incluso años. Afortunada o desafortunadamente, para el desarrollo de este proyecto se contó con un par de meses. Por lo que la eficiencia y la toma de decisiones en cuanto a las tecnologías, modelos y configuraciones fueron clave para finalizarlo y crear la aplicación lo más cercano a lo descrito en los objetivos iniciales.

La aplicación se realizó haciendo uso de las tecnologías aprendidas durante el curso; en este sentido se dividió el desarrollo en dos partes:

El back-end o servidor y el front-end, es decir, la página web.

Back-End

El servidor se creará haciendo uso de el lenguaje de programación nodejs, el cuál es una variante de javascript, desde el servidor se servirá la **data** o la información que será usada en la aplicación.

¿Qué información necesita nuestra aplicación? En la etapa de diseño ya definimos ciertos componentes de la aplicación y en ellos definimos que características tendrían, en consecuencia, la información que necesitaría cada uno de los componentes quedaría de la siguiente forma:

- Sidebar: Para el sidebar podríamos enviar que elementos tendrían que mostrarse, pero se decidió mantener la construcción de este componente en el front-end. Esta decisión podría no ser la mejor a futuro cuando se quiera agregar más menús al componente.
- Administrador de archivos: En el servidor se construirá el proyecto y se enviará a front-end un **json**, el cuál es un tipo de dato en forma de directorio, en el que la información se organiza por llaves únicas. En este **json** enviaremos información del directorio del proyecto, en donde cada llave contendrá: nombre del archivo o carpeta, su ruta en el directorio, una bandera que indique si es archivo o carpeta.

Es importante tener en cuenta el identificador único de cada archivo, será su ruta más su nombre

- Editor de texto: Aquí utilizaremos el identificador que mencionamos en el punto anterior, ya que la aplicación front-end haría un llamado al servidor cada que se abra un archivo nuevo, mandando el identificador del archivo, de modo que el servidor retornaría el texto que el archivo contenga, si el archivo es de tipo binario u otro imposible de abrir, el servidor retornará un error específico.
- Terminal o consola: Cuando el usuario ejecute el proyecto desde la aplicación front-end, se deberá mandar una petición al servidor, del cual retornará la información de la ejecución del programa, hecho en el mismo servidor.
- Reproductor de video: Debido al tiempo esta parte no pudo ser implementada al cien por ciento, pero la idea para seguir el desarrollo a futuro es que, al ejecutarse el programa, retornar al mismo tiempo información en segmentos de tipo stream, los cuales serían reproducidos desde el front-end.

¿Por qué se decidió usar Nodejs para la creación del servidor? La versatilidad del lenguaje de programación ayuda a facilitar el trabajo, ya que existen librerías precargadas en el lenguaje para resolver casi cualquier problema que surgió durante el desarrollo, también al ser un lenguaje basado en javascript no tendríamos que preocuparnos por aprender una tecnología diferente a las ya vistas en el curso.

En cuanto a la arquitectura usada en el servidor, no es nada muy complejo, tan solo se usa una arquitectura modular por rutas, de modo que lo único que se tiene que hacer desde el front-end es solicitar una url base y la ruta.

Front-End

Para el desarrollo de la aplicación web se trabajó haciendo uso de las siguientes tecnologías: React.js, vite, monaco-editor, HTML y CSS.

Cuando se habla de desarrollo de aplicaciones web, hay 3 prescindibles que no pueden faltar, HTML, CSS y JavaScript (JS).

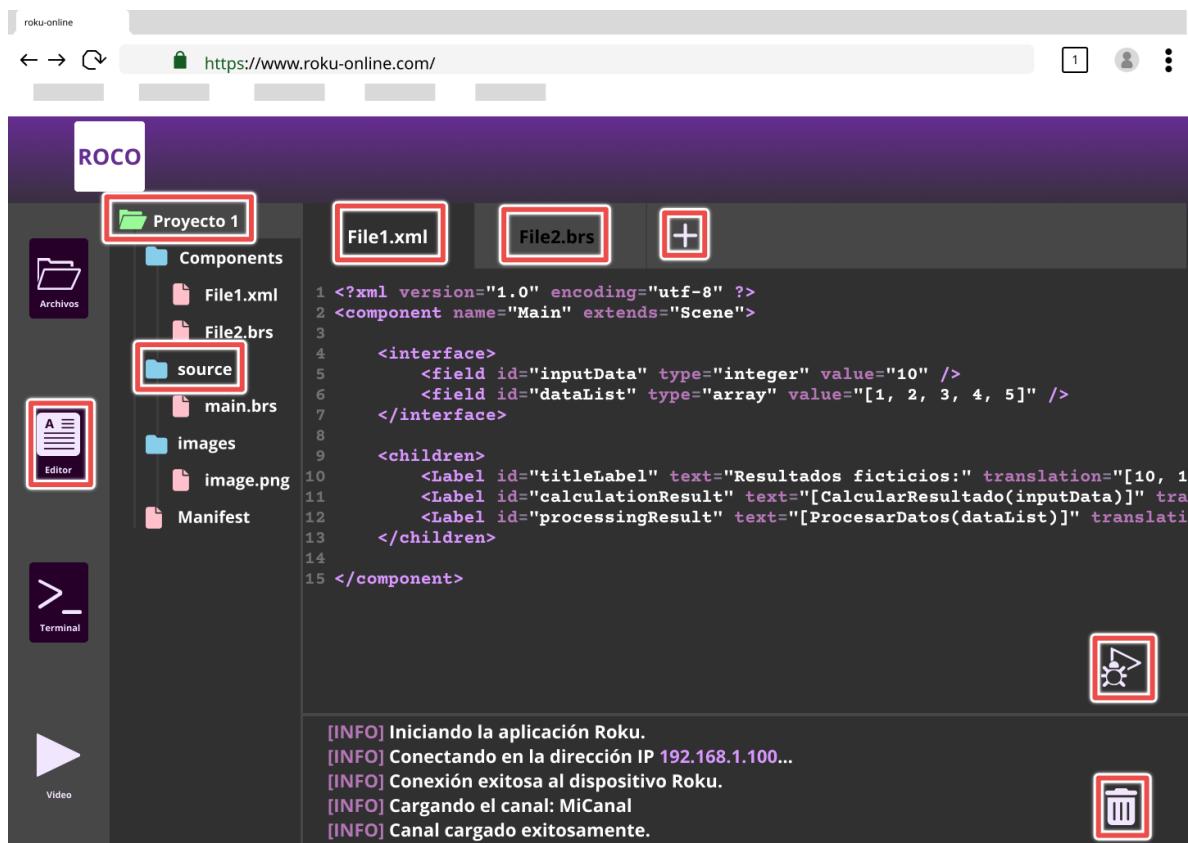
Aunque JavaScript es poderoso y puede ser suficiente para proyectos más pequeños, su uso implicaría un desarrollo muy tardado y complicado en proporción al uso del framework React. De modo que React aborda desafíos específicos en el desarrollo de aplicaciones web modernas, ofreciendo una estructura más organizada, herramientas que simplifican el desarrollo con buenas prácticas de código pasando de una escritura imperativa a una escritura declarativa.

Entre muchas ventajas, React fomenta la creación de aplicaciones a través de componentes, los cuales son bloques de construcción reutilizables, optimizando así, las líneas de código

necesarias al escribir el programa, esto no solo facilita la creación del código sino el mantenimiento y su reutilización.

En cuanto a vite, es una tecnología que nos ayudará a proporcionar una mayor eficiencia al desarrollo ya que proporciona un tiempo de arranque muy rápido con una mínima configuración del marco de desarrollo.

La construcción del sitio web se inició analizando la estructura de componentes. Se empezó buscando desde lo más elemental a lo más general, se buscó el elemento más pequeño que pudiera ser reutilizado dentro de los diseños, llegando así al componente más simple; el cuál es tan solo el ícono y texto



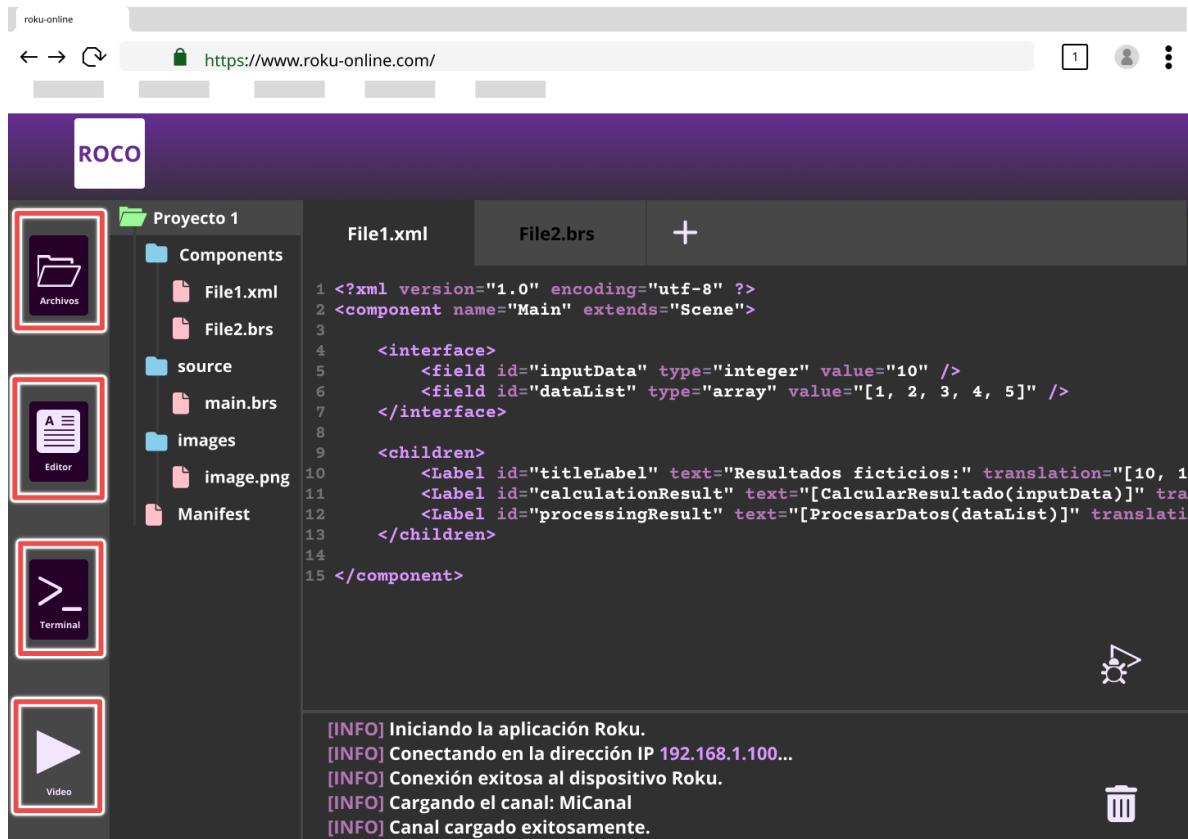
A este componente lo llamamos **HyperLink**, ya que se encargaría de manejar la visibilidad y asignación de íconos y textos y porque está pensado para ser clickable.

La idea de un componente en React es básicamente ser reutilizable, de modo que, en nuestra aplicación, este componente acepta de parámetro un objeto el cuál contendrá información de las clases para los estilos y la ruta al ícono

```
export function HyperLinkItem({children}) {  
  
    return (  
        <section>  
            <img className={children.iconClassName} src={children.icon} />  
            <span className={children.textClassName}>  
                {children.text}  
            </span>  
        </section>  
    )  
}
```

Ejemplo del componente React más sencillo en la aplicación.

Posteriormente, se decidió la creación del siguiente componente, este tenía que ser, obviamente un componente que contuviera al componente previo, pero que de igual forma pudiera ser reutilizable. Así se llegó a la idea de crear los botones del sidebar. Este componente estaría compuesto por un contenedor que se pinta en caso de estar seleccionado o no y el componente HyperLink.



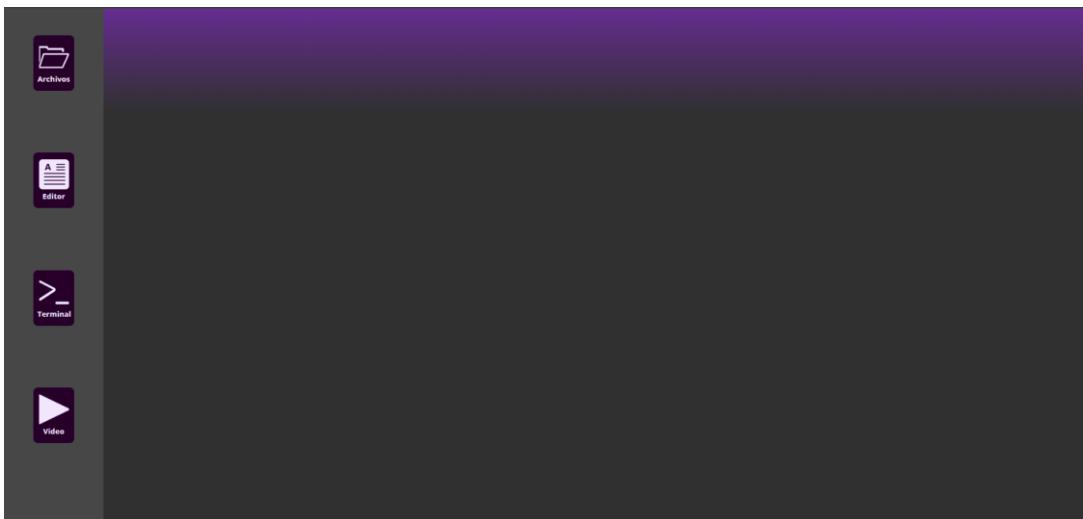
Este componente realizará una operación de iteración sobre un arreglo de objetos el cuál definirá cada botón. El objeto de cada botón tendría la siguiente forma:

```
{  
  "id": "",  
  "iconClassName": "",  
  "textClassName": "",  
  "icon": "",  
  "text": ""  
},
```

De modo que el componente queda de la siguiente forma:

```
export function SidebarItem() {  
  return (  
    <>  
    {  
      buttonsObj.map(element => {  
        return (  
          <button key={element.id} className='sidebar-element-button'>  
            <HyperLinkItem key={element.id}>  
              {element}  
            </HyperLinkItem>  
          </button>  
        )  
      })  
    }  
  )  
}
```

Ya teniendo el componente más general, lo único que nos queda por hacer es crear el componente Sidebar el cuál renderizará nuestro componente SidebarItem y crear los estilos para el componente, de modo que el componente al final queda de la siguiente manera:



Componente Sidebar en la aplicación real

Una vez terminado, probablemente, el componente más sencillo de la aplicación pasaremos con la creación del componente del Administrador de Archivos

Para este componente usaremos la data proveniente del servidor. Aquí no tenemos un contenedor por lo que el componente más general será tal cuál el componente del Administrador de Archivos y el componente más elemental es el del `HyperLink`.

Al empezar el desarrollo nos encontramos con un gran problema, la data que nos llega está ordenada por ruta de archivo, por ende, si iterara sobre ese objeto tendría todos los elementos iniciales en la parte superior. Para resolver este problema, se creó una clase para un objeto de un árbol y una subclase para cada nodo del árbol.

Una clase en programación es un conjunto de código que define un tipo de objeto, especificando sus atributos y comportamientos. Las clases sirven como plantillas o moldes.

Cada subclase nodo tendrá las propiedades siguientes:

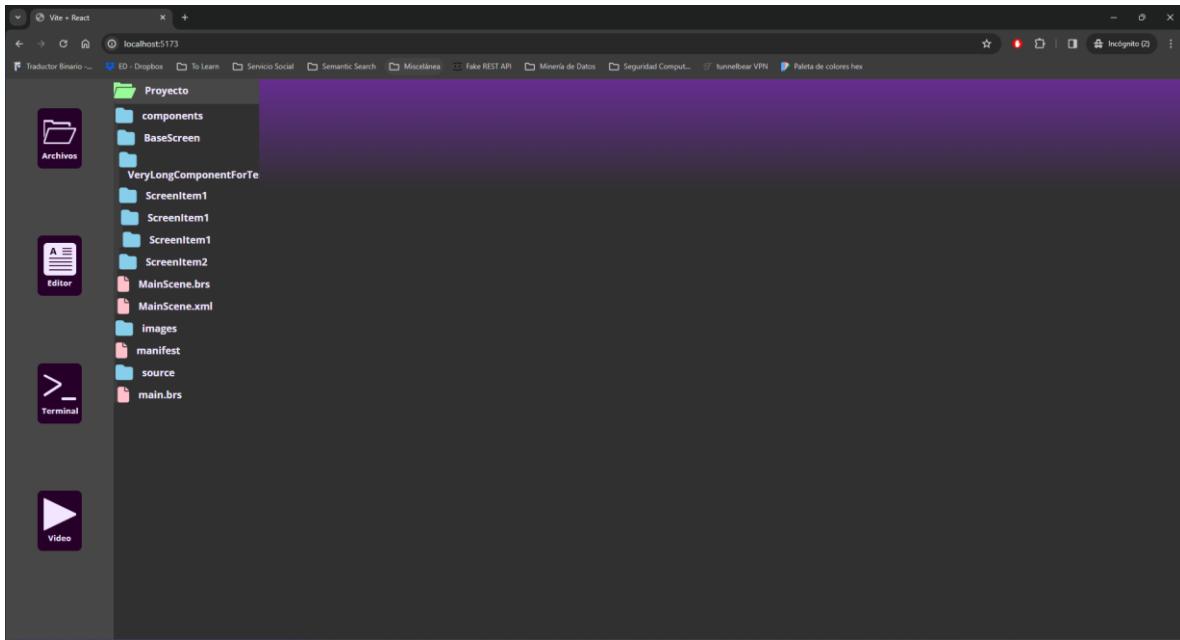
- `name`: Define el nombre del nodo
- `childNodes`: Define los nodos hijos para ese nodo
- `isFolder`: Define si el nodo es un folder o un archivo
- `parent`: Define el nodo padre para ese nodo

La clase Árbol de Nodos contiene los siguientes atributos y métodos(funciones):

- `childrenArray`: Arreglo donde se guardarán los componentes `HyperLink` junto con sus estilos.
- `root`: Define el nodo raíz del cuál ramificarán todos los demás nodos hijos. Este nodo es del tipo de la subclase nodo.
- `addNewNode`: Define una función para agregar un nuevo nodo. Se revisa el parent del nodo y se asigna en su atributo `childNodes`.
- `getRenderable`: Define una función donde se crean los subcomponentes con los datos en el árbol y los nodos hijos.

Una vez creados todos los nodos y habiendo llamado a la función `getRenderable` nos encontramos con dos nuevos problemas:

Los estilos se rompen y cuando hay componentes con nombres muy largos se pierden, también si tuviéramos un directorio enorme no podríamos ver todos los nodos hijos.



Este problema se resolvió asignando un padding o espaciado para cada elemento de unos 20px cada vez que sea un nuevo nivel de hijos en el árbol y respecto a poder ver más elementos cuando el árbol tenga muchísimos hijos se creó un nuevo componente el cuál permitió el movimiento para componentes que lo requieran. De este componente hablaremos después ya que su creación vino dada en otro momento del desarrollo.

El componente siguiente es el que llevó más tiempo de todos, a pesar de que la lógica en el administrador de archivos fue muy compleja. Este componente es el Editor de Texto, este componente se podría separar en dos componentes diferentes, los cuales serían el selector de archivos y el editor de texto, pero se decidió que estén en un mismo componente como subcomponentes.

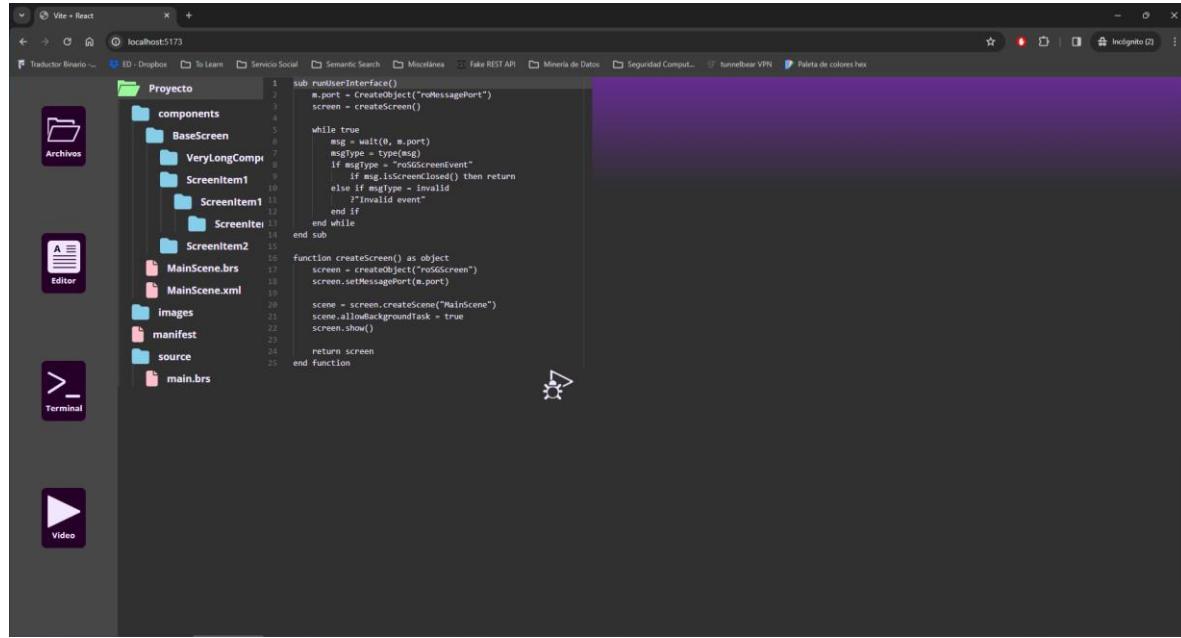
Se inició el desarrollo de este componente con la investigación de cómo se podría crear un editor de texto desde cero, se inició usando la etiqueta `textarea` de HTML, pero hacían falta muchísimas funcionalidades, como asignar colores, movimiento dentro del área de texto, el ajuste del texto y el recorte de las palabras, poner el número de líneas escritas y que se coordinen según donde nos encontramos dentro del área de texto y muchísimos problemas más. Como vemos, el desarrollo de este componente es, un proyecto completo en sí mismo, por lo que se decidió usar una paquetería ya creada. Se encontraron dos ideas, las cuales son las más usadas por la comunidad

1. [codemirror](#)
2. [monaco-editor](#)

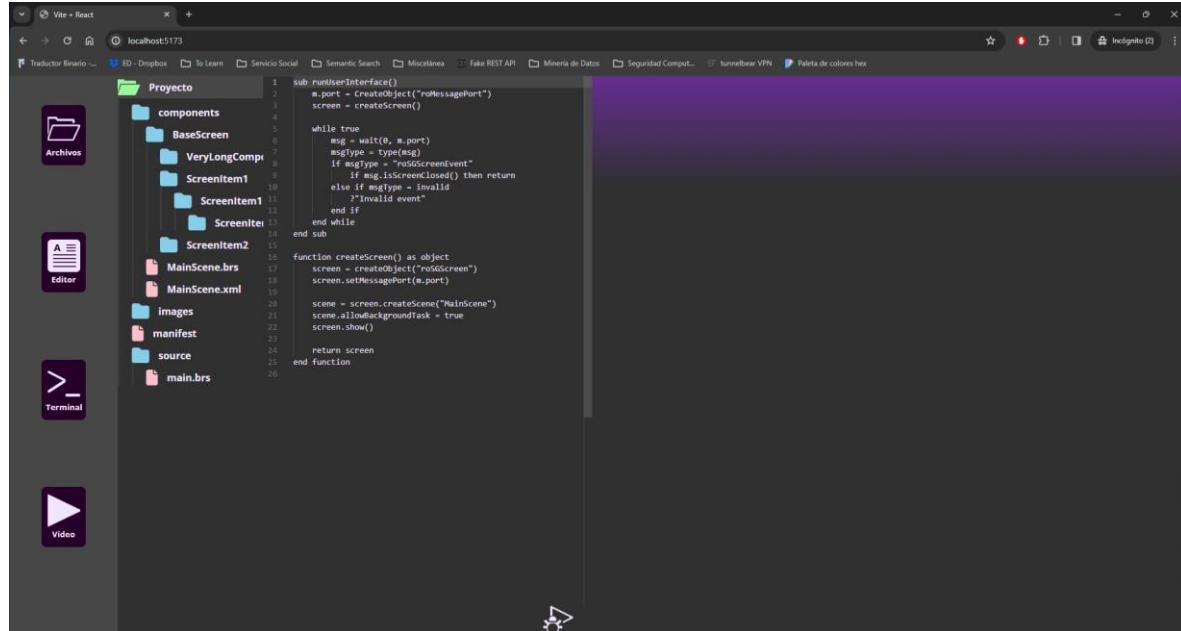
Al final se decidió usar el editor de texto monaco creado por Microsoft, fue a mi consideración el más completo y se ajustaba perfectamente como pieza de rompecabezas a nuestro desarrollo.

Una vez descargada e instalada la paquetería se empezó a configurar, se definió el tema que sería usado para que concordara con los colores y el branding de la página, se definió también la colorización de corchetes, paréntesis y llaves y el recorte de palabras en caso de superar el tamaño.

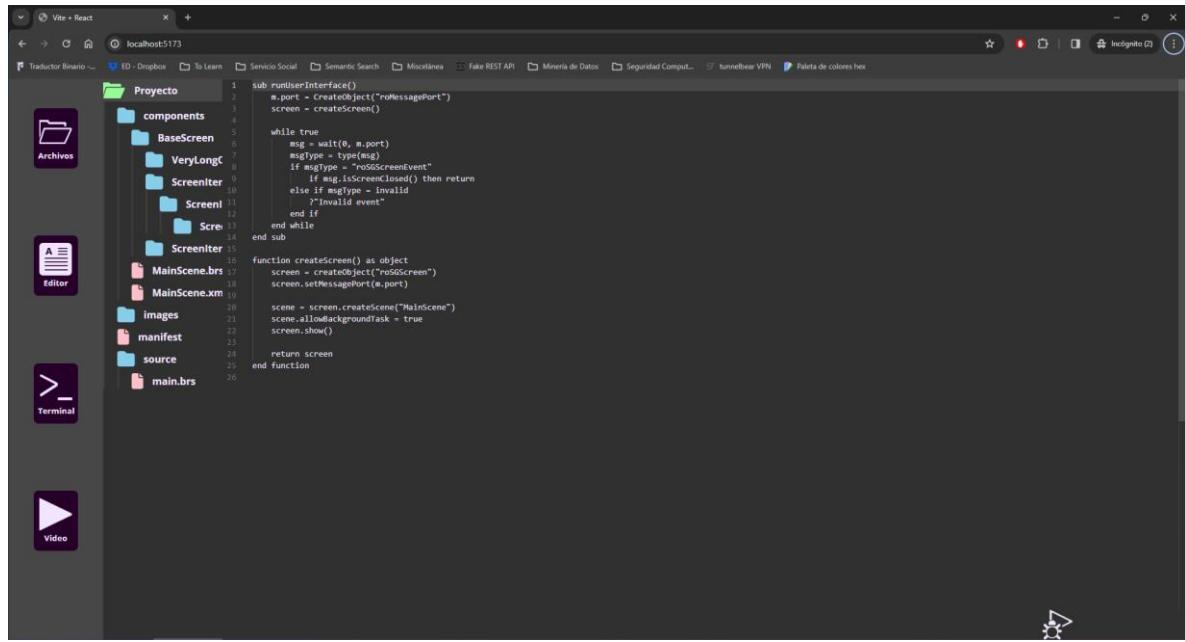
Se definieron los estilos para cuando existan los demás componentes:



The screenshot shows a web browser window with the URL `localhost:5173`. The left sidebar displays a project structure with files like `Archivos`, `Editor`, `Terminal`, `Video`, and a main folder containing `components`, `BaseScreen`, `VeryLongCompi`, `ScreenItem1`, `ScreenItem2`, `MainScene.brs`, `MainScene.xml`, `images`, `manifest`, `source`, and `main.brs`. The main content area shows a large portion of the `main.brs` file, which contains a complex script with many nested loops and conditionals. The script includes functions for creating screens and handling messages. The browser interface has a dark theme with purple highlights.



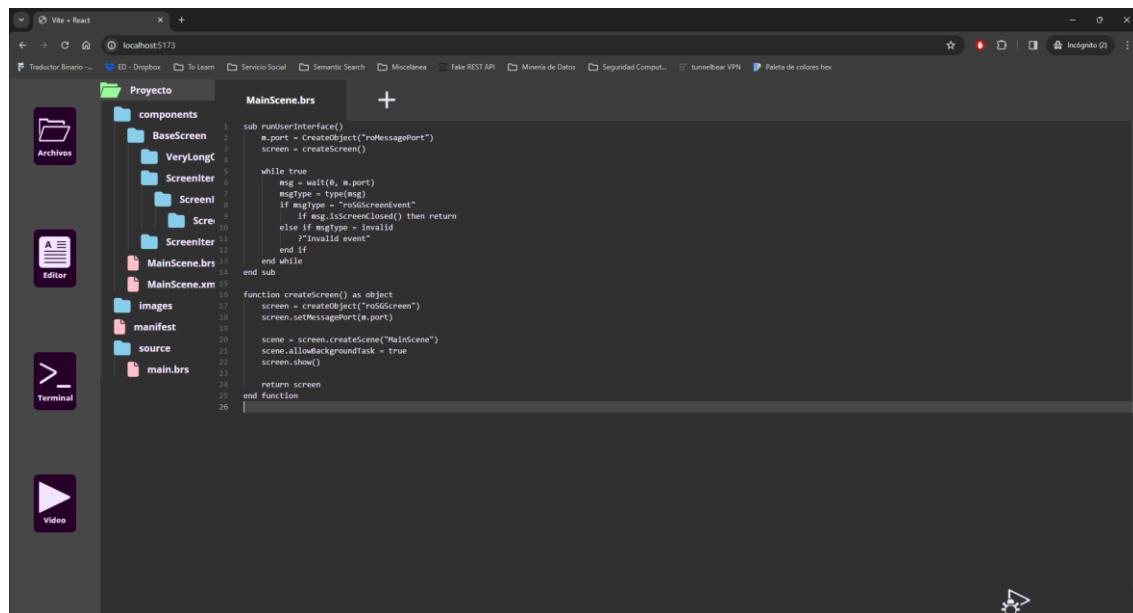
This screenshot is similar to the one above, but it shows a smaller portion of the `main.brs` file. The visible code is identical to the first screenshot, but the scroll position is lower, showing lines 1 through approximately 25 of the script. The browser interface remains consistent with the first screenshot.



```
1 sub runUserInterface()
2     m.port = CreateObject("roMessagePort")
3     screen = CreateScreen()
4
5     while true
6         msg = wait(0, m.port)
7         msgType = type(msg)
8         If msgType = "roSScreenEvent"
9             If msg.isScreenClosed() then return
10            else if msgType = invalid
11                ?"Invalid event"
12            end if
13        end while
14
15    function createScreen() as object
16        screen = CreateObject("roSGScreen")
17        screen.setMessagePort(m.port)
18
19        scene = screen.createScene("MainScene")
20        scene.allowBackgroundTask = true
21        screen.show()
22
23        return screen
24    end function
25
26
```

La data que usará este componente vendrá desde el servidor, para esto, al seleccionar un archivo dentro del componente del administrador de archivos, haremos la petición mandando en nombre y ruta del archivo, así recibiremos el texto al archivo correspondiente.

Para la creación del componente de selector de archivos sí se terminó creando un subcomponente para cada ítem, muy parecido a lo hecho en el sidebar, donde tendremos el contenedor que cambiará de color dependiendo de si está seleccionado o no y el componente HyperLink. La data que se necesita en este componente es construida dentro de la misma aplicación, ahí definiríamos el nombre del archivo y si está seleccionado o no. De modo que, al iterar y crear los elementos, quedaría de la siguiente manera:



```
1 sub runUserInterface()
2     m.port = CreateObject("roMessagePort")
3     screen = CreateScreen()
4
5     while true
6         msg = wait(0, m.port)
7         msgType = type(msg)
8         If msgType = "roSScreenEvent"
9             If msg.isScreenClosed() then return
10            else if msgType = invalid
11                ?"Invalid event"
12            end if
13        end while
14
15    function createScreen() as object
16        screen = CreateObject("roSGScreen")
17        screen.setMessagePort(m.port)
18
19        scene = screen.createScene("MainScene")
20        scene.allowBackgroundTask = true
21        screen.show()
22
23        return screen
24    end function
25
```

Este componente trajo consigo en su desarrollo un problema extra, y es el mismo problema que se tuvo con el administrador de archivos; cuando se tienen muchos elementos, se pierden por el tamaño del contenedor mismo, por lo cuál se tuvo que crear un componente nuevo al cuál se le llamó **Draggable** esto para poder movernos dentro de componentes que sean muy grandes, este componente calcula la ubicación del mouse al seleccionar dentro del componente y hasta donde se mueve, para así luego mover el componente dentro de su contenedor.

El componente **Draggable** recibe dentro de sus subcomponentes, el componente que se desea móvil.

Por último, se creó el componente del logger o terminal, este componente fue algo más sencillo ya que se usó lo aprendido en el editor de texto y se usó una instancia parecida.

Pruebas y Depuración:

El desarrollo se realizó haciendo uso del editor de código fuente visual studio code.

Para la ejecución del proyecto usamos el entorno de desarrollo de vite, de modo que tan solo se necesitaba abrir una terminal del sistema y escribir el comando **npm run dev**.

Esta es exactamente la misma forma en la que ejecutamos nuestro servidor.

npm es el administrador de paquetes de Node, el mismo lenguaje que se usó para el servidor, esto debido a que React y vite usan node para facilitar las ejecuciones.

Se probaron diferentes medidas para verificar la responsividad del sitio, también se usaron diferentes dispositivos tanto móviles, como tabletas y ordenadores.

Conclusión

Este proyecto ha explorado la creación de un compilador en línea destinado al desarrollo de aplicaciones Roku, abordando desafíos significativos en el proceso de codificación para esta plataforma. Al centrarnos en el entorno de desarrollo de Roku, que involucra los lenguajes SceneGraph y BrightScript, hemos delineado la importancia de superar las limitaciones existentes y proporcionar a la comunidad de desarrolladores una herramienta eficiente y accesible.

El diseño de la aplicación, respaldado por la arquitectura Vite+ReactJS y un backend Node.js, ha sido guiado por la búsqueda de eficiencia y facilidad de uso. La elección de utilizar ReactJS sobre JavaScript vanilla ha permitido una estructuración más organizada y la creación de componentes reutilizables, mientras que la adopción de Vite ha impulsado el desarrollo rápido y eficiente.

El resultado es un compilador en línea que no solo satisface las necesidades prácticas de los desarrolladores de Roku, sino que también representa un paso hacia adelante en la democratización del desarrollo para esta plataforma. Al eliminar las barreras de acceso y mejorar la eficiencia del proceso, aspiramos a catalizar la innovación y el crecimiento en la comunidad de desarrollo de aplicaciones Roku.

Este proyecto no solo es una herramienta técnica, sino un esfuerzo por potenciar y expandir el campo de posibilidades para aquellos que buscan explorar el desarrollo de contenido multimedia a través de dispositivos Roku.

Referencias

Introducción y Marco teórico

1. https://www.linkedin.com/posts/eshap-media-cartographer_ces2024-ces-activity-7148656604597407745-pvGJ?utm_source=share&utm_medium=member_desktop
2. <https://developer.roku.com/es-mx/docs/developer-program/getting-started/first-steps.md>
3. <https://developer.roku.com/es-mx/docs/developer-program/getting-started/developer-setup.md#sideloading-channels>
4. <https://developer.roku.com/es-co/docs/references/brightscript/language/brightscript-language-reference.md>
5. <https://developer.roku.com/en-gb/docs/developer-program/core-concepts/scenegraph-xml/overview.md>
6. <https://marketplace.visualstudio.com/items?itemName=RokuCommunity.brightscript>
7. <https://developer.roku.com/es-mx/docs/developer-program/getting-started/hello-world.md>

Diseño

MoodBoard

1. <https://www.roku.com/es-mx/>
2. <https://color.adobe.com/es/create/color-contrast-analyzer>
3. [https://color-register.org/color/roku-purple#:~:text=Basic%20facts%20about%20the%20digital,%2C%2045%2C%20145\)%%20](https://color-register.org/color/roku-purple#:~:text=Basic%20facts%20about%20the%20digital,%2C%2045%2C%20145)%%20)
4. <https://websitehurdles.com/why-programmers-prefer-dark-mode/>
5. <https://vscodethemes.com/>
6. Imágenes sacadas de:
 - a. <https://pixabay.com/es/>
 - b. <https://pixabay.com/es/photos/desarrollador-de-software-6521720/>
 - c. <https://pixabay.com/es/illustrations/sitio-web-sensible-creativo-dise%C3%B1o-3374825/>
 - d. <https://pixabay.com/es/photos/netflix-peliculas-youtube-digitales-3733812/>
 - e. <https://pixabay.com/es/photos/rel%C3%A1mpago-tormenta-superzelle-2568381/>
7. Imágenes editadas con: <https://www.photopea.com/>
8. Paleta de colores analizados con: <https://color.adobe.com/es/create/image>

9. Moodboard hecho con: <https://www.canva.com/>

Mapa de sitio

1. Information Architecture: For the Web and Beyond pdf:
https://openaccess.uoc.edu/bitstream/10609/143950/1/Arquitectura%20de%20la%20informacion_Guia%20de%20lectura%20de%20Information%20Architecture.%20For%20the%20Web%20and%20Beyond%20de%20Rosenfeld%2C%20Morville%20y%20Arango%202015.pdf
2. https://developer.mozilla.org/es/docs/Glossary/Mobile_First
3. <https://www.programiz.com/python-programming/online-compiler/>
4. El mapa, los íconos y todos los diseños fueron realizados en:
<https://www.figma.com/>
5. Para la creación de algunos íconos se usó la aplicación de Paint de Windows y para la conversión de .svg se usó:
<https://www.photopea.com/>

Wireframes y wireflows

1. <https://moqups.com/blog/wireframe-vs-mockup-vs-prototype/>
2. Diseños de los Wireframes y wireflows en Figma:
<https://www.figma.com/file/a2nCxyc05MtbumITxkT0m7Z/Untitled?type=design&node-id=43-8&mode=design>
3. Prototipo para los dispositivos móviles:
<https://www.figma.com/proto/a2nCxyc05MtbumITxkT0m7Z/Untitled?type=design&node-id=204-3595&t=dKr91RWWCC6pI03J-0&scaling=scale-down&page-id=204%3A2877&starting-point-node-id=204%3A3595>
4. Prototipo para tablets:
<https://www.figma.com/proto/a2nCxyc05MtbumITxkT0m7Z/Untitled?type=design&node-id=431-733&t=dKr91RWWCC6pI03J-0&scaling=scale-down&page-id=431%3A732&starting-point-node-id=431%3A733>
5. Prototipo para ordenadores:
<https://www.figma.com/proto/a2nCxyc05MtbumITxkT0m7Z/Untitled?type=design&node-id=508-3675&t=dKr91RWWCC6pI03J-0&scaling=scale-down&page-id=508%3A2492&starting-point-node-id=508%3A3675>
6. <https://alistapart.com/article/responsive-web-design/>

Desarrollo

BackEnd

1. <https://curl.se/>
2. <https://nodejs.org/api/>
3. <https://expressjs.com/en/4x/api.html>
4. <https://www.npmjs.com/package/dotenv>

5. <https://medium.com/@mmajdanski/express-body-parser-and-why-may-not-need-it-335803cd048c>

FronEnd

1. Monaco editor: <https://www.npmjs.com/package/@monaco-editor/react>
2. Edición de temas: <https://code.visualstudio.com/api/references/theme-color>
3. Draggable component: <https://stackoverflow.com/questions/75833107/click-and-drag-to-scroll-with-mouse-react-typescript-component>