Manual Usuario

Compiladores 1

MINIMAL REACT CMS

proyecto 1

Introducción

Minimsl rect csm, un CMS web que permite la creación de sitios web de manera fácil a través de un editor de código que permita la creación de páginas web dinámicas.

Las páginas web que se creen deberán ser hechas por medio de archivos de Minimal React, este lenguaje combina lógica de Typescript y estructura de HTML para realizar páginas dinámicas que serán traducidas a HTML.

Por último la estructura de los sitios web deberá ser descrita por medio del lenguaje TOML, este sitio web estructurará que paginas hay dentro de un sitio web.

Página principal

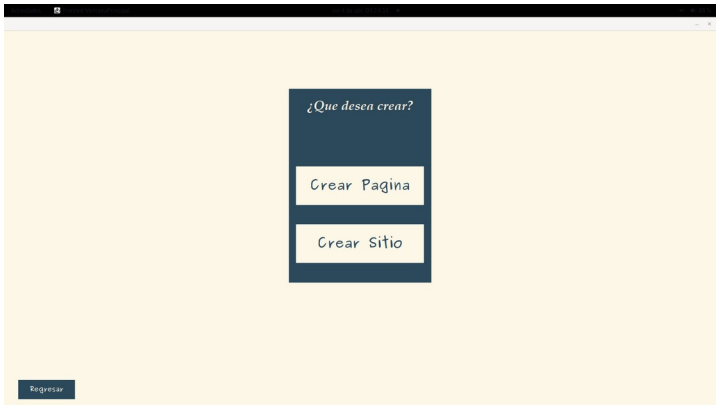
En la página principal el usuario podrá crear nuevos sitios web o editar los actuales.

El usuario puede elegir en la interfaz de la aplicación entre dos modos: Modo Edición y Modo Visualización.

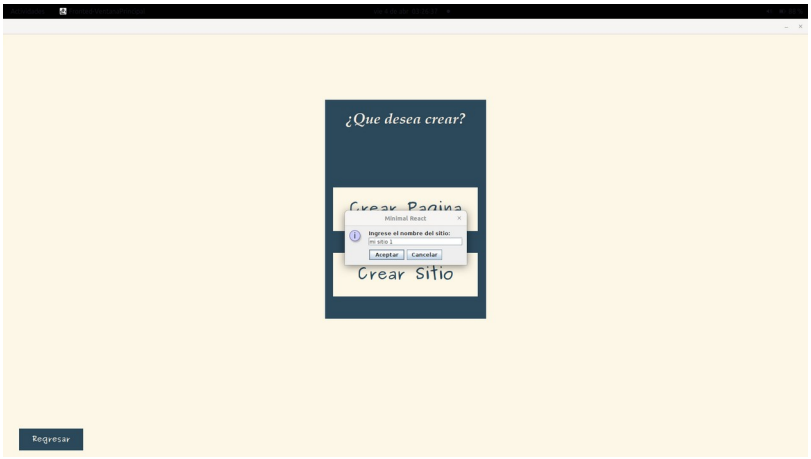
En esta página el usuario podrá ver detalles acerca de los sitios y podrá abrirlos para verlos, editarlos y también podría borrarlos.

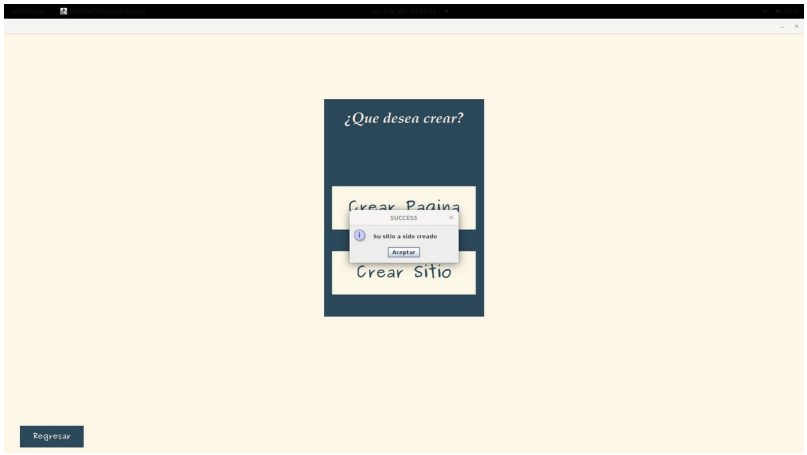
Modo Creación

En este modo el usuario puede crear una pagina individual, o un sitio.



El usuario ingresara el nombre de su sitio o pagina, luego presionara el botón de aceptar.



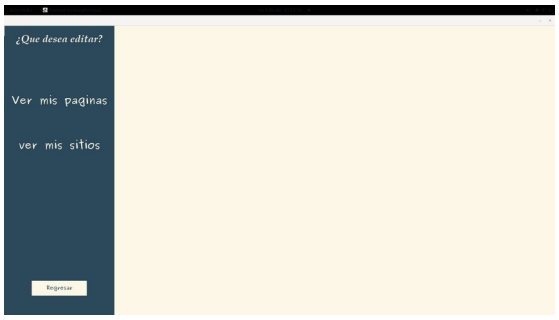


Modo Edición

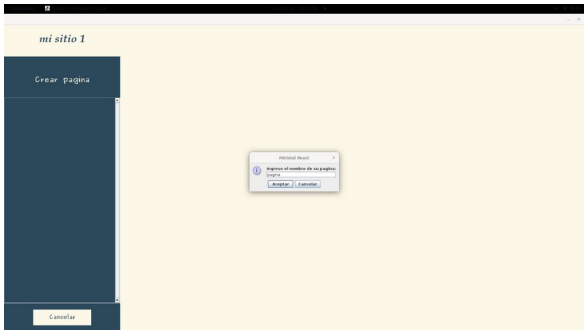
El modo edición de la aplicación incluirá un editor de texto para los archivos .mtsx

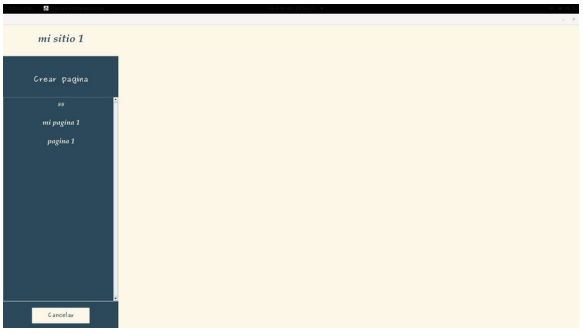
El usuario podrá abrir algún sitio web y sus páginas para poder editarlas y guardarlas.

Así como también puede agregar páginas al sitio y editarlas.



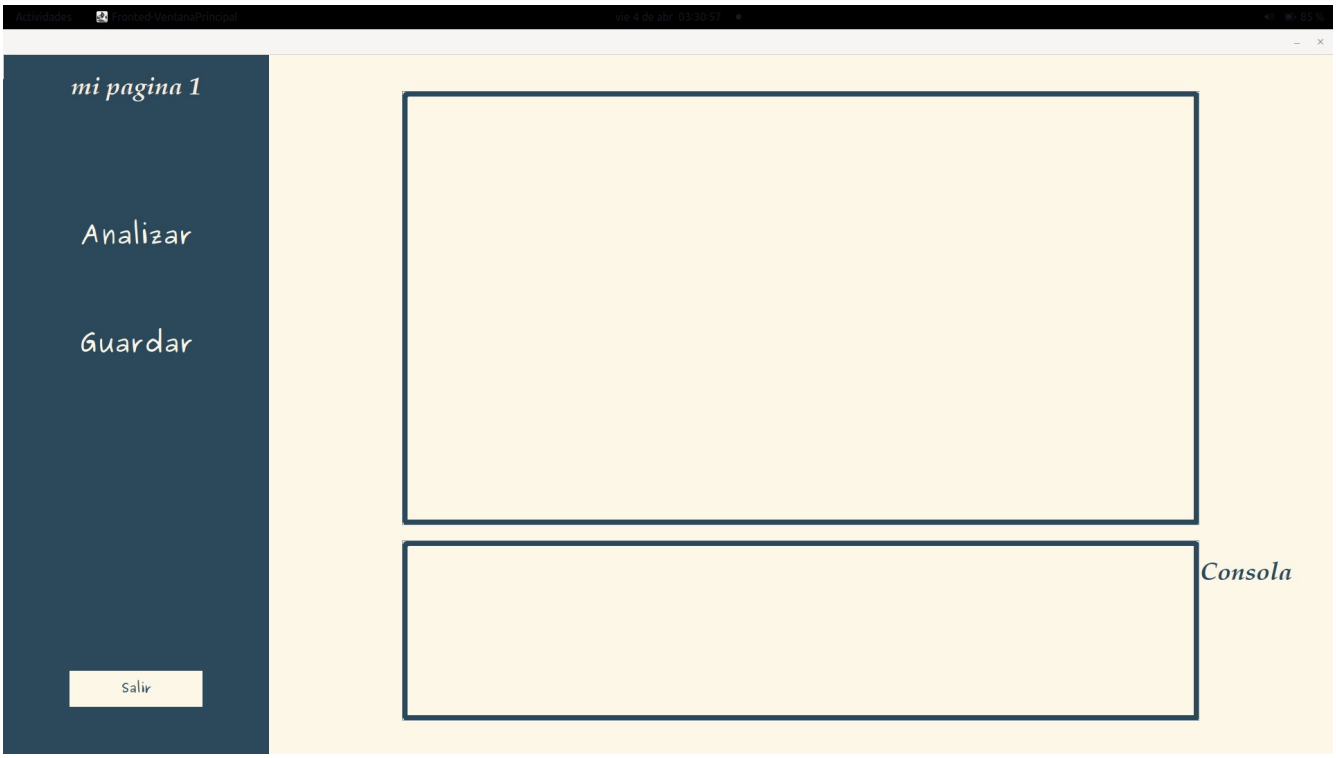
El usuario puede crear paginas para, dentro de sus sitios





Edición paginas:

El usuario puede editar ya sea paginas individuales o pagina de los sitios.

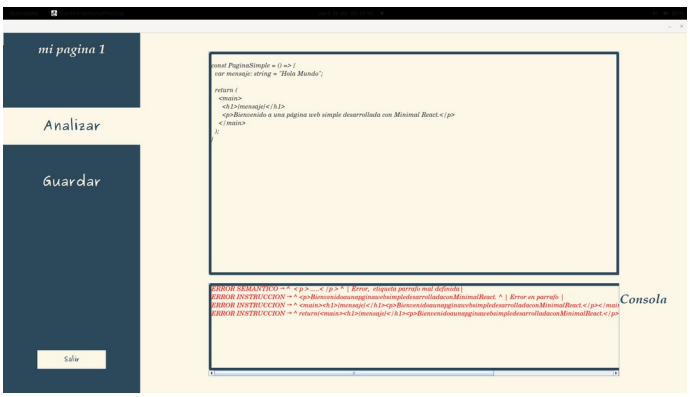


Botón Guardar: Este botón sirve para guardar los cambios en la pagina, si se detecta algún cambio en el código que no se a analizado, el botón se desactivar, hasta realizar el análisis.

Botón Analizar: Este botón sirve para analizar el código en el editor.

Editor: Es donde se ingresa el código minimal react.

Consola: Nos mostrara si existen errores en el análisis, o si las instrucciones son correctas.



Modo Visualización

El modo visualización de la aplicación le permitirá a los usuarios visualizar los sitios y páginas que estarán en formato HTML/JavaScript, es posible visualizar la estructura de un sitio como un mapa de sitio HTML para que el usuario pueda moverse entre páginas.

Lenguaje Minimal React

Minimal React (con extensión .mtsx) es un lenguaje de programación inspirado en React que es una mezcla entre HTML y Typescript que sirve para el desarrollo de páginas web dinámicas.

Para mostrar el resultado de esta página es necesario convertirla a HTML y JavaScript puro.

```
const NombrePagina = () => {
     var texto: string = “Hola Mundo”;
     var variable: number = 10;

     function funcion(): void {
     variable = variable + 1;
}

     return (
     <main>
          <h1>{texto}</h2>
          <input value={variable}/>
          <button onClick={funcion}>Boton</button>
     </main>
);
}
```

Ejemplo de página .mtsx

Typescript

Typescript es el lenguaje base sobre el cual se construirán las páginas, este lenguaje es fuertemente tipado por lo que es siempre necesario definir tipos para todo tipo de funcionalidad.

Todas las páginas estarán en archivos .mtsx y deben partir de la declaración de una arrow function y el retorno de esta contiene la descripción del sitio a través de un HTML dinámico.

Declaración de una Página

Todas las páginas deben declararse como arrow functions del siguiente modo:

```
const <nombre_pagina> = () => {
    ...contenido
    return(
    <contenido_html>
);
}
```

Return de Componente

Todas las declaraciones de componentes deben tener un return en donde se declara la estructura de la página en HTML dinámico, el cual se especificará en la siguiente sección de lenguaje.

```
return (
    ...<contenido_html>
);
```

Ejemplo de return de componente.

Tipos de Variables

Tipo	Descripción	Ejemplos
number	Números enteros o decimales	5, 0.14, 3, 24, -1, -3.14
string	Strings de texto, van entre comillas dobles	“Hola mundo”, “Texto”
char	Carácter entre comillas simples	‘c’, ‘a’, ‘x’
boolean	Booleano que representa verdadero o falso	true, false
void	Tipo especial para funciones que no retornan nada.	void

Asignación de Variables

La versión de Typescript de Minimal React es un lenguaje fuertemente tipado por lo que es necesario definir los tipos de todas las variables que se definan.

```
var variable: number = 3.14;
var texto: string = “Hola Mundo”;
```

Es posible el uso de variables para definir otras variables

```
var var1: number = 2.5 * 8;
var var2: number = 5.0 + var1;
var1 = 4;
```

Operaciones Matemáticas

Es posible la realización de operaciones matemáticas simples definidas por los siguientes operadores.

Símbolo	Descripción	Precedencia (de menor a mayor)
+	Suma	1
-	Resta	1
*	Multiplicación	2
/	División	2
^	Potenciación	3

Operaciones Relacionales

Es posible realizar operaciones relacionales que conviertan condiciones en booleanos.

Símbolo	Descripción	Precedencia (de menor a mayor)
>	Mayor que	4
<	Menor que	4
>=	Mayor o igual que	4
<=	Menor o igual que	4

Impresión en consola

A través de la consola integrada del editor es posible visualizar información impresa a través de la función console.log()

Esta función acepta cualquier tipo de parámetro del siguiente modo.

```
console.log(“texto”);

console.log(variable);

console.log(suma_1 + suma_2);
```

Funciones

Es posible definir funciones para encapsular lógica dentro de Typescript.

La estructura de una función es:

```
function <nombre_funcion>(parametros): tipo_retorno {
    ... contenido
}
```

Estas funciones pueden o no devolver un valor

Las funciones se pueden ejecutar en cualquier parte del código usando la siguiente estructura:

```
<nombre_funcion>(parametros);
```

También pueden asignarse a variables del siguiente modo.

```
var resultado: number = suma(4, 6);
```

Parámetros

Las funciones aceptan parámetros tanto en su definición como en su implementación.

Para definir parámetros en la definición de una función se hace del siguiente modo:

```
function <nombre_funcion>(nombre_parametro: tipo, nombre_parametro: tipo...){
    ...
}
```

Para usar estos parámetros cuando la función es implementada se hace del siguiente modo:

```
<nombre_funcion>(valor1, valor2...);
```

Retorno

Todas las funciones que especifique un tipo de retorno diferente a void deberán tener una sentencia return dentro para ser válidas, y este return deberá retornar el tipo con el cual se especificó la función.

```
function suma(num1, num2): number {
     return num1 + num2;
}
```

Ejemplo de retorno en una función.

Es posible usar return en una función void siempre y cuando no retorna nada

```
function condicion(variable): void {
     if (variable > 5) {
          return;
}
console.log("Hola mundo");
}
```

Ejemplo de retorno en una función void.

Condiciones

Es posible definir condicionales en Typescript usando la siguiente estructura.

```
if ( <condicion> ) {
     ...contenido
}
```

Este puede tener else e if-else del siguiente modo.

```
if ( <condicion> ) {
     ...bloque
} else {
     ...bloque
}
```

```
if ( <condicion> ) {
     ...bloque
} else if ( <otra_condicion> ) {
     ...bloque
}
```

Las condiciones deberán tener de tipo booleano, como resultado de alguna operación relacional o booleana.

HTML

El lenguaje HTML es usado de forma dinámica para definir la estructura de una página dentro de un sitio, este HTML se deberá

poder enlazar con las variables y funciones de Typescript usando las etiquetas que se definirán en esta sección.

Hay que denotar que todos los elementos entre llaves en esta sección ({ }) hará referencia a alguna variable o función de Typescript.

Este lenguaje sólo podrá ser usado en los returns de páginas definidos anteriormente.

Etiqueta main

La etiqueta main (<main></main>) sera una etiqueta que envolvera todo el codigo HTML dentro de un componente y todo el codigo HTML estara dentro de esta etiqueta.

```
<main>
    ...<todo el demás contenido>
</main>
```

Ejemplo de etiqueta main

Etiquetas de encabezados

Al igual que en HTML normal, el HTML dinámico usado usará etiquetas de headers (<h1></h1>) para denotar algún texto que se le aplicará formato, estas etiquetas irán desde la h1 hasta la h6.

Dentro de estas etiquetas es posible hacer referencia a variables de typescript.

```
<h1>Hola mi nombre es {nombre}</h1>
```

Ejemplo de etiqueta h1 usando una variable definida en Typescript.

Etiquetas de Párrafos

Al igual que las etiquetas de header, las etiquetas de párrafos sirven para presentar texto en pantalla, y es posible referenciar variables de Typescript en estas etiquetas.

```
<p>Hola mi nombre es {nombre}</p>
```

Inputs

Es posible usar inputs para obtener información del usuario, esta información podrá ser almacenada en variables para su posterior uso, o usada inmediatamente.

Todos los inputs deben estar asociados a una variable Typescript que guarde su valor a través de su atributo value.

```
var nombre_usuario: number = “”;
...
return (
    <input value={nombre_usuario} />
);
```

Buttons

Dentro del HTML dinámico es posible el uso de botones para alterar el comportamiento de la página, estos botones pueden tener acciones a partir de una funcion Typescript la cual es asociada al boton por medio de su atributo onClick

```
function incrementar(): void {
    variable = variable + 1;
}
...
return (
    <button onClick={incrementar}>Boton</button>
);
```

Las funciones usadas dentro de estos componentes pueden tener parámetros, estos parametros son variables definidas en el lenguaje Typescript.

```
var num1: number = 0;
var num2: number = 0;
var resultado: number = 0;
function suma(var1, var2): void {
    resultado = var1, var2;
}
...
return (
    <button onClick={suma(num1, num2)}>Sumar</button>
);
```