EWeb

Una nueva era para el desarrollo web en Free Pascal

Sumario

Introducción a EWeb

Bienvenido a EWeb, el framework web ligero y potente desarrollado íntegramente en ***Free Pascal*** desde cero.

EWeb nace con un objetivo claro: ofrecer una herramienta simple, eficiente y completamente controlable para crear aplicaciones web modernas sin las complicaciones y dependencias de otros frameworks. Hemos reescrito cada componente esencial para priorizar la claridad, el rendimiento y la facilidad de uso, permitiendo que los desarrolladores en ***Free Pascal/Lazarus*** construyan proyectos robustos con un mínimo de configuración.

EWeb nace con un objetivo claro: ofrecer una herramienta simple y eficiente sin las complicaciones de otros frameworks.

¿Por qué EWeb?

- **Simplicidad real**: Un gestor de rutas intuitivo, middlewares fáciles de encadenar y un contexto web extensible (***TEContext***) que simplifica la interacción en cada petición.
- **Todo desde cero**: Servidor HTTP propio, motor de plantillas ligero con soporte para variables, funciones y bucles, manejo moderno de formularios, uploads y sesiones.
- **Sin dependencias externas**: Abandonamos componentes como ~~WebLaz~~ o ~~HttpRoute~~ en favor de soluciones propias, más ligeras y adaptadas.
- **Extensible y seguro**: La versión Premium añade gestión de contraseñas, tokens CSRF, JWT y un gestor de bases de datos (SQLite y MariaDB) con serialización automática para integrar perfectamente con las plantillas.

EWeb está diseñado para proyectos de cualquier tamaño: desde APIs rápidas hasta aplicaciones web completas. Su arquitectura limpia te da control total sin sacrificar productividad.

Primeros pasos

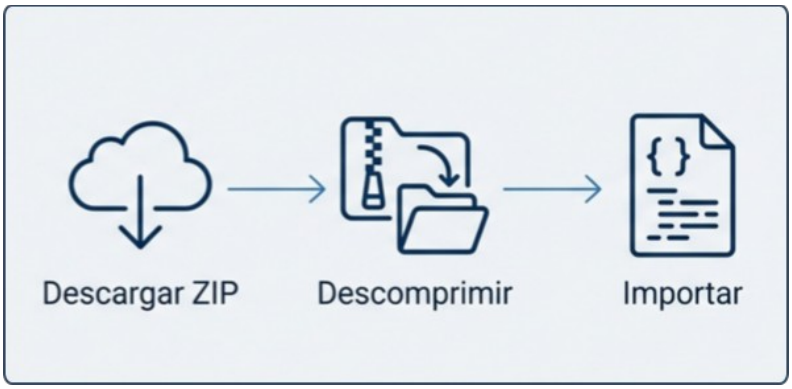
Para comenzar:

1. Descarga la versión actual desde la página oficial.
2. Crea un proyecto básico y descomprime ***EWeb*** dentro.
3. Ejecuta el servidor integrado y ve tu primera ruta en acción.

En las siguientes secciones encontrarás guías detalladas sobre instalación, configuración de rutas, uso del motor de plantillas, middlewares y mucho más.

¡Gracias por elegir EWeb! Si tienes sugerencias o encuentras algún problema, no dudes en contactarnos.

Proceso de instalación y configuración



EWeb está diseñado para integrarse fácilmente en proyectos Lazarus/Free Pascal, sin dependencias externas complejas.

Requisitos de instalación

Antes de comenzar, asegúrate de tener lo siguiente instalado:

- **Free Pascal compiler (**FPC 3.2.2 o superior) con **Lazarus IDE** (versión recomendada: 2.2 o superior) o cualquier editor de textos.
- **Sistema operativo compatible:** Windows, Linux o macOS (probado principalmente en Windows y Linux). Solo para sistemas de 64 bits.
- **Espacio en disco mínimo:** ~10 MB para la versión actual.

No se requieren paquetes adicionales de terceros; todo lo necesario viene incluido en ***EWeb***.

Descargar EWeb

1. Ve a la página oficial del proyecto:
   - ***https://edgardomlopez.ddns.net/software/eweb***
2. Descarga la versión **EWeb 0.4** (archivo ZIP) o la versión correspondiente.
   - Enlace directo: reemplaza la última versión, por ejemplo: https://edgardomlopez.ddns.net/download/eweb-0.4
3. Descomprime el archivo en una carpeta de tu elección (por ejemplo, C:\Proyectos\EWeb o ~/Proyectos/EWeb).
4. Crea una aplicación simple en Lazarus o un archivo ".pas" en un editor de textos.
5. Descomprime ***eweb*** dentro de la carpeta de tu proyecto e importa la API.

Estructura o anatomía de un proyecto EWbeb

Es recomendable tener dentro de la carpeta del proyecto la carpeta ***eweb*** extraída del *zip* descargado. No se debe cambiar el nombre ya que buscará dentro de esta carpeta los archivos necesarios para funcionar.

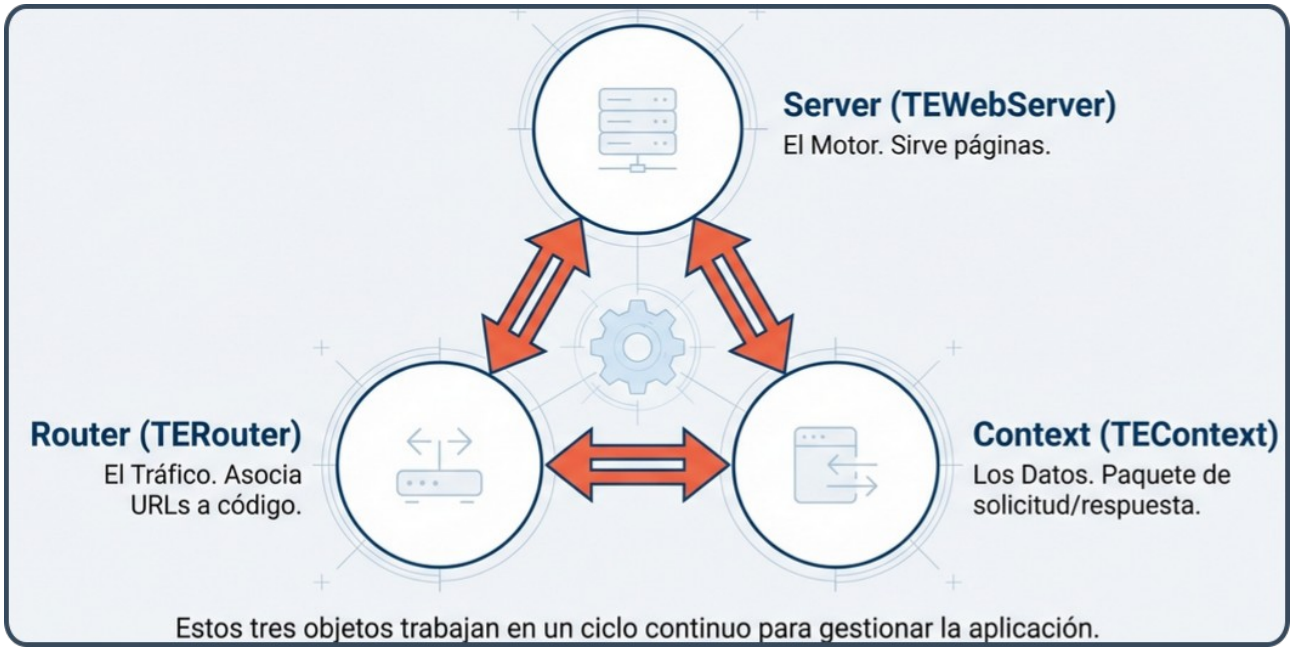
Para su integración solo debemos importar la api pública en nuestro archivo principal o en cualquier unidad que necesite interactuar con ***EWeb***.

```
uses
  WebAPI, public, routes,

var
  Server: TEWebServer;
  Router: TERouter;
```

La arquitectura de EWeb

Se denomina arquitectura a la forma que tiene el framework de trabajar. En ***EWeb*** la forma de trabajo es cíclica. Esto quiere decir que cada respuesta tiene un camino que se podría denominar como circular. En primer lugar tenemos el servidor, el cual se encarga de servir las páginas web, las sesiones, etc. El mismo interactúa con el Router que es el encargado de obtener las solicitudes desde el navegador (como una página en específico, el proceso de un formulario, la validación de sesión). Y todo interactúa con el contexto web, que es el encargado de agrupar toda la información necesaria para poder enviar una solicitud a través del Router al Servidor, del Servidor al Router. Y también se encarga de interactuar tanto con el Router como con el Servidor. Esto se puede interpretar como una comunicación circular de estos tres elementos entre sí y en cualquier sentido.

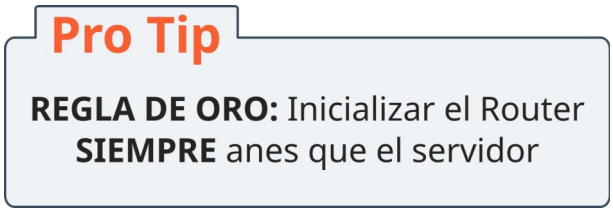


El Router

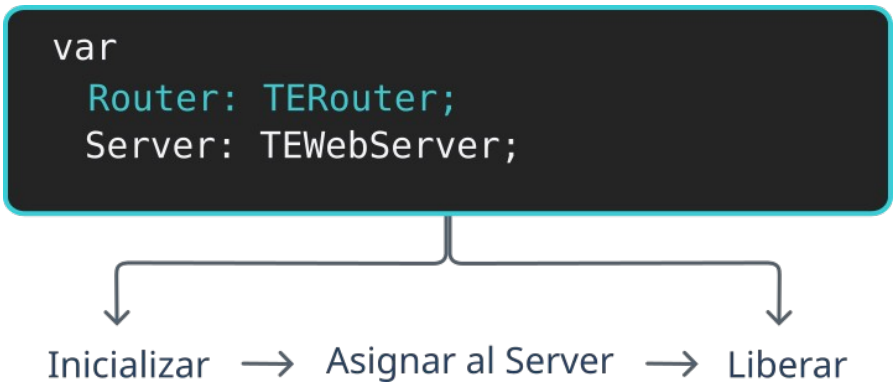
Es un módulo interno que se encarga de obtener toda la información recibida desde el navegador, enviarla a un ***Handler*** o ***Procedimiento*** para poder trabajar con la información recibida y enviar una respuesta al navegador.

Un ejemplo sería recibir la solicitud de cargar una página o descargar un archivo. El ***Router*** recibe la solicitud y se la envía a un procedimiento, el cual se encargará de procesar la información y enviar la respuesta correspondiente de vuelta a través del ***Router***.

Nota: Es importante declarar el Router siempre antes que el Servidor. Ya que el servidor requiere como parámetro obligatorio el Router dónde va a procesar todas las solicitudes del navegador.



Para declarar el Router debemos definir una variable de tipo ***TERouter*** la cual debemos inicializar y asignar al Servidor posteriormente. Y al liberar los objetos siempre hay que liberar primero el Router.



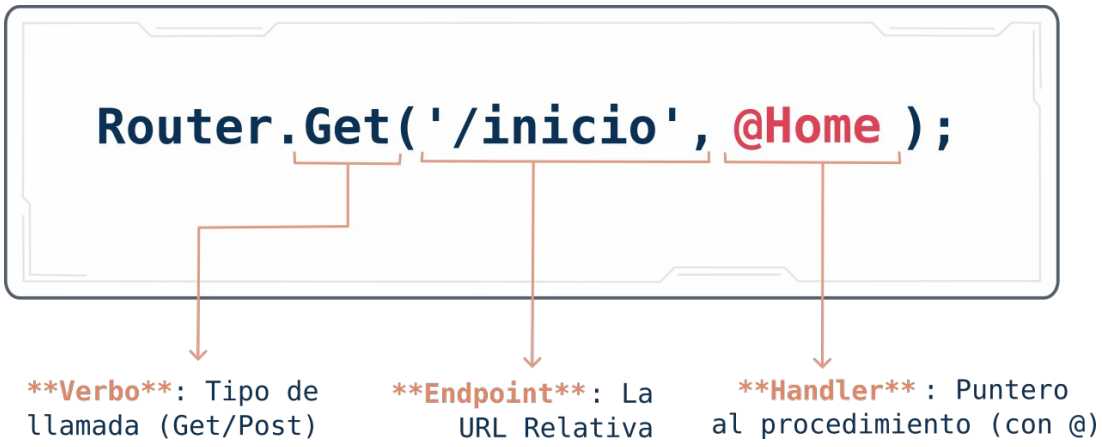
Definiendo Rutas

Es muy importante recordar que el Servidor trabaja directamente con las rutas. Cuando hablamos de rutas, estamos hablando de una llamada con un tipo y nombre

específico que recibe el servidor. Mediante la cual ese nombre o ruta le permite al Router poder ejecutar un procedimiento para interactuar las solicitudes y respuestas del navegador con nuestro código Free Pascal para realizar operaciones.

La estructurara de una Ruta se divide en 3 partes:

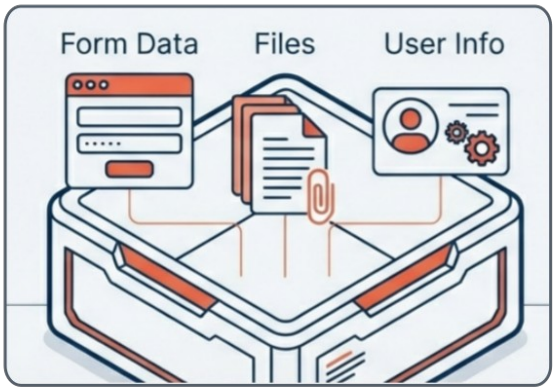
1. ***Tipo***: aquí definimos el tipo de llamada a la ***Router***. Puede ser de tipo ***GET*** y ***POST*** (por el momento).
2. ***URL***: pasamos como primer parámetro el nombre que tendrá nuestra URL después o seguido al ***Localhost*** o ***URL*** base de nuestro sitio web.
3. ***Handler***: el segundo parámetros es un puntero o enlace al ***procedimiento*** que se encargará de procesar y responder nuestra solicitud. El nombre del procedimiento siempre debe anteponer un arroba (@).



El Contexto Web

El contexto web es un contenedor que almacena toda la información recibida desde el navegador capturada en la ruta.

Al solicitar una ruta el Contexto Web se encarga de almacenar todo lo que el navegador solicita (qué página necesita, qué datos envía, si es un formulario, si solicita una descarga, etc). Todo lo que está dentro de esta solicitud es empaquetado y enviado dentro del Contexto Web.



EWeb tiene acceso a este contexto y mediante él puede acceder a la información y procesar una respuesta de forma ordenada y sencilla. Todo irá empaquetado dentro del Contexto Web dentro de la solicitud actual. De esta forma solo debemos enfocarnos en qué cosas podemos o queremos hacer con este paquete y nada más.

El Handler (Procedimiento)

Podríamos decir que aquí es dónde ocurre la magia. El Handler es un procedimiento con una estructura y parámetros determinados que nos permitirá interactuar con el contexto recibido y usarlo para brindar una respuesta al servidor.

La forma de trabajo es bastante sencilla. Recibimos el Contexto Web como un parámetro obligatorio (Puntero). Dentro del procedimiento debemos convertir este puntero y asignarlo al TEContext para así poder acceder a todo el contenido recibido y poder procesar una respuesta mediante el Contexto Web hacia el Servidor y mediante el Router.

La estructura o forma de trabajar con un Handler es bastante sencilla.

```
procedure  Home(HCtx: Pointer);
var
  ctx:TEContext;
begin
   ctx:= NewContext(HCtx);
   try
     ctx.Text('Hola EWeb');
   finally
     ctx.Free;
   end;
end;
```

Requisito Obligatorio

Extracción de Contexto

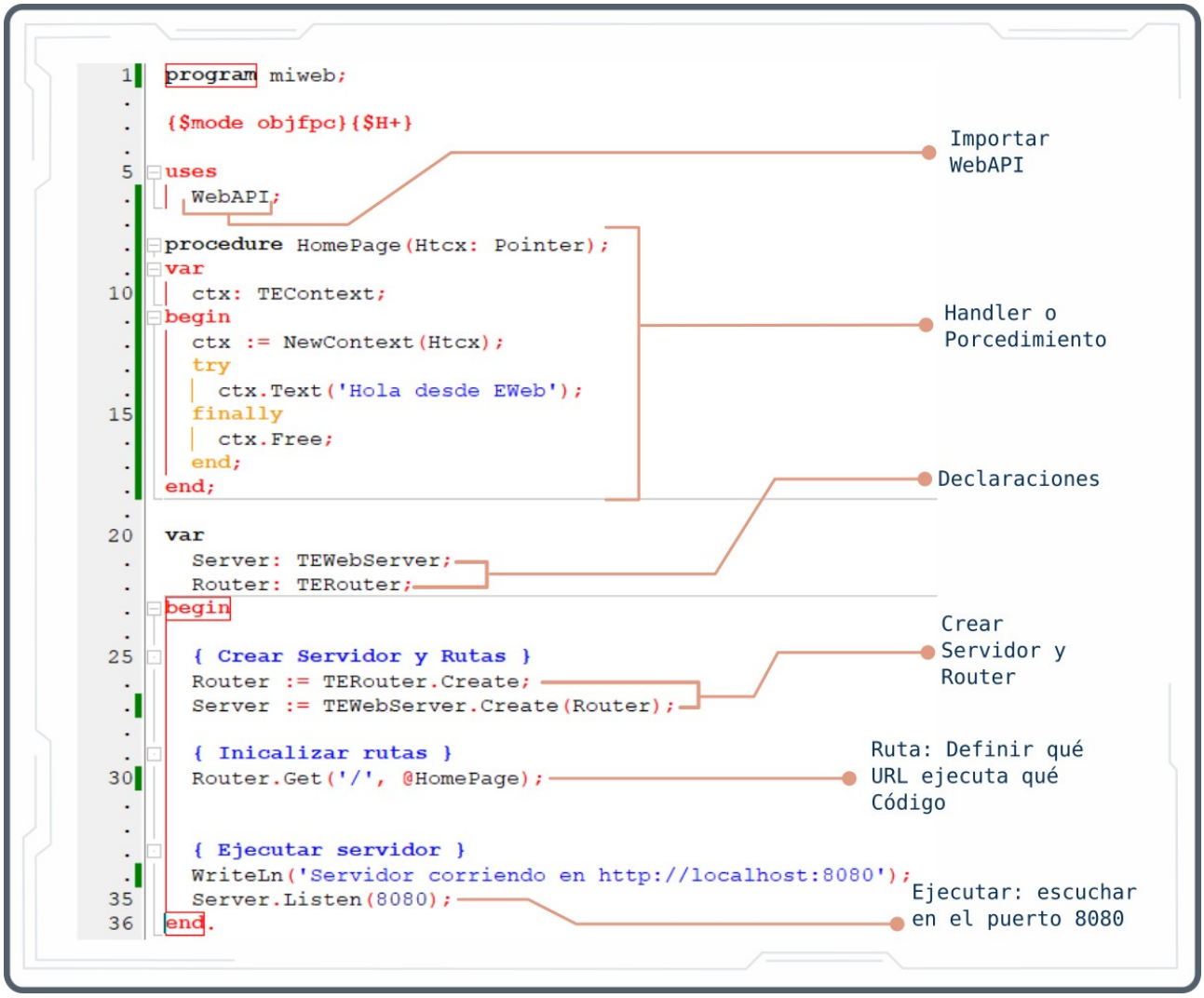
Gestión de Memoria Segura

Crear tu primer servidor

Para la creación de una aplicación simple debes declarar 2 objetos obligatorios:

1. ***TERouter***: el manejador de rutas y todo lo relacionado a la interacción entre páginas del proyecto.
2. ***TEWebServer***: el servidor que maneja, interactúa con todo, no solo las rutas y sirve las páginas procesando las solicitudes del navegado.

Veamos un ejemplo de servidor mínimo con una ruta:



Ejecutar tu primer servidor

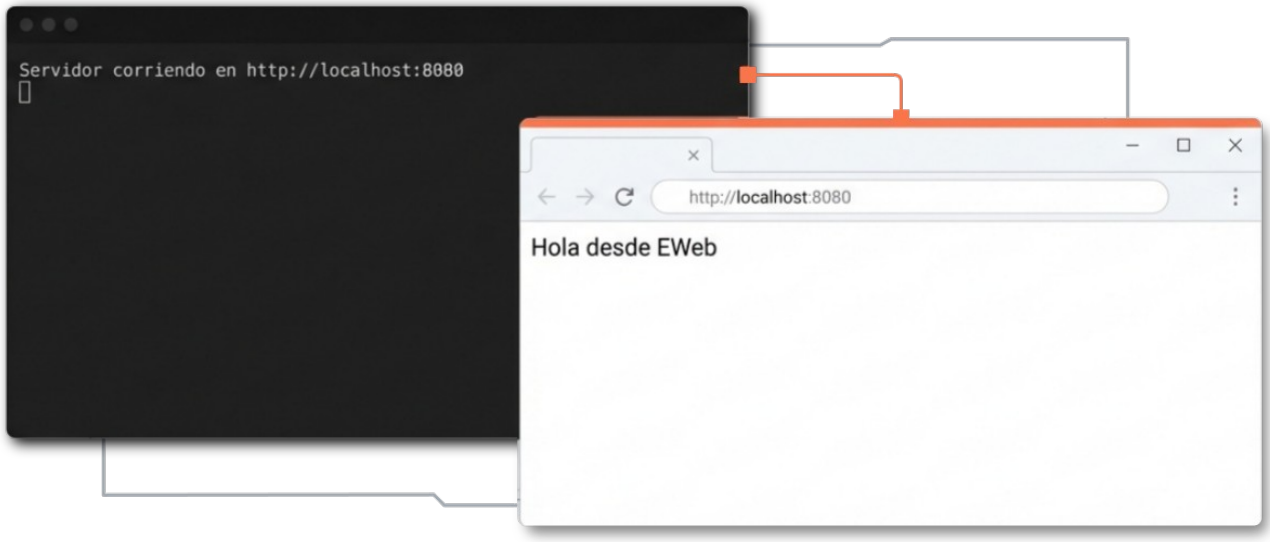
Parar ejecutar el servidor por primera vez debemos usar el comando Compilar del menú ejecutar

Pro Tip

Usa la función **Construir (BUILD)** para asegurar la **INTEGRACIÓN** de EWEB

Si todo construyó correctamente puedes ejecutar la aplicación desde una terminal o presiona **Run → Run (o F9)** para ejecutar el archivo compilado.

Si construyó bien deberás ver en consola el mensaje de ejecución (***si lo haz definido como mensaje con un WriteLn()***) y en el navegador podrás ver la respuesta del servidor.



Respuestas básicas del TEContext

El **TEContext** permite, no solo procesar un Contexto Web recibido. Sino que también permite enviar una respuesta al navegador a través del servidor.

Los tipos de respuesta básicos son

1. ***Texto:*** Permite enviar como respuesta un texto plano sin formato.
2. ***HTML:*** permite enviar como respuesta texto con interpretación de etiquetas e interpretarlas como HTML en el navegador.
3. ***JSON:*** Permite enviar una estructura JSON con todas sus propiedades. Es ideal para en un futuro implementar más funcionalidades para trabajo con API Rest.
4. ***Render:*** Esta función permite renderizar un archivo .html, procesar variables y condicionales como lo hacen otros lenguajes. Es muy útil para separar la lógica de la aplicación de la interfaz web que por lo general se hace con HTML, CSS y JS.