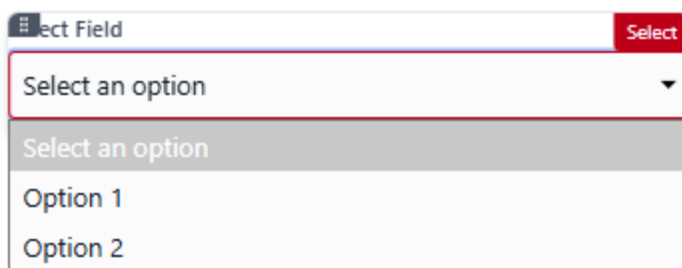


Nexus – Elemento: Select



¿Qué es?

Select es un componente de tipo **Form**. Es un menú desplegable de selección de opciones. Al arrastrarlo al lienzo viene con dos opciones de ejemplo precargadas (Option 1, Option 2).



Propiedades

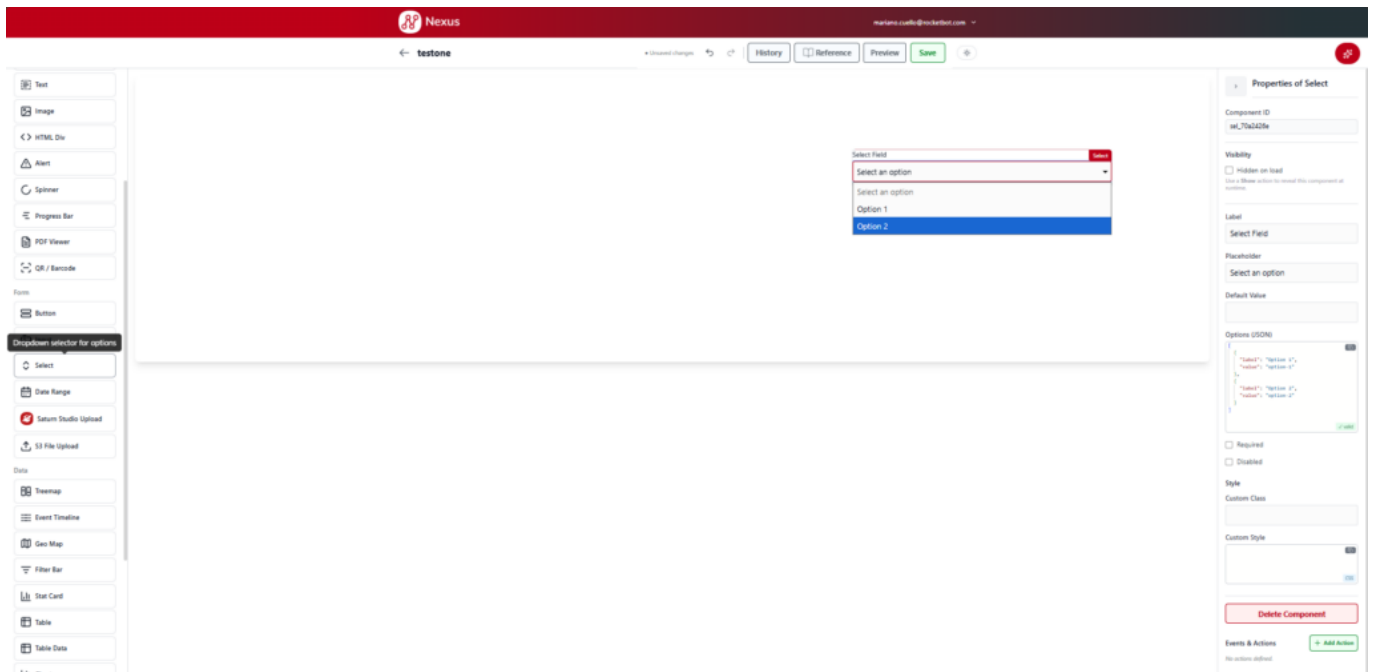
Al seleccionar Select en el lienzo, el panel derecho muestra “**Properties of Select**” con las siguientes opciones:

- **Component ID**: identificador único del componente, autogenerado (ej. sel_70a2426e).
- **Visibility → Hidden on load**: casilla para que el componente esté oculto al cargar la pantalla. Puede mostrarse luego mediante una acción de tipo **Show** (ver Events & Actions).
- **Label**: etiqueta que se muestra encima del campo (ej. “Select Field”).
- **Placeholder**: texto que aparece como opción inicial antes de elegir algo (ej. “Select an option”).
- **Default Value**: opción seleccionada por defecto al cargar la pantalla.
- **Options (JSON)**: campo de código donde se definen las opciones del desplegable en formato JSON. Cada opción se compone de:
 - **label**: texto visible de la opción.
 - **value**: valor interno asociado a esa opción.

El editor valida el JSON en tiempo real (marca “✓ valid” cuando el formato es correcto).

- **Required**: casilla que marca el campo como obligatorio.
- **Disabled**: casilla que deshabilita el campo (queda inactivo, no seleccionable).
- **Style → Custom Class**: campo para asignar una clase CSS personalizada.

- **Style → Custom Style:** campo de texto para CSS personalizado aplicado directamente a este componente.
- **Delete Component:** elimina el componente del lienzo.
- **Events & Actions:** sistema para definir comportamientos interactivos (por ejemplo, disparar una acción al cambiar la opción seleccionada, con trigger **On Change**). Ver documentación aparte: [Nexus – Events & Actions](#).



Ejemplo de uso

Para agregar una nueva opción, hay que ubicarse en la última llave `}` del bloque anterior dentro de **Options (JSON)**, agregarle una coma, y pegar el nuevo bloque a continuación:

```
[
  {
    "label": "Option 2",
    "value": "option-2"
  },
  {
    "label": "Option 3",
    "value": "option-3"
  }
]
```

⚠ Importante: la coma va pegada a la llave `}` del bloque anterior, no al principio del bloque nuevo.

Select Field Select

Select an option ▼

Select an option

Option 1

Option 2

Option 3

Visibility

☐ Hidden on load

Use a **Show** action to reveal this component at runtime.

Label

Select Field

Placeholder

Select an option

Default Value

Options (JSON)

```
[
  {
    "label": "Option 1",
    "value": "option-1"
  },
  {
    "label": "Option 2",
    "value": "option-2"
  },
  {
    "label": "Option 3",
    "value": "option-3"
  }
]
```

✓ valid

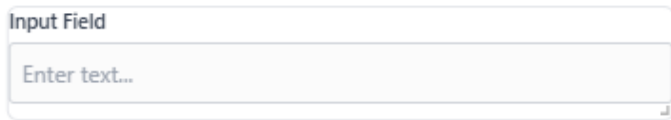
El valor seleccionado por el usuario (value de la opción elegida) puede referenciarse luego desde otras partes de la app (queries, acciones) usando el **Component ID** del Select.

Nexus – Elemento: Input



¿Qué es?

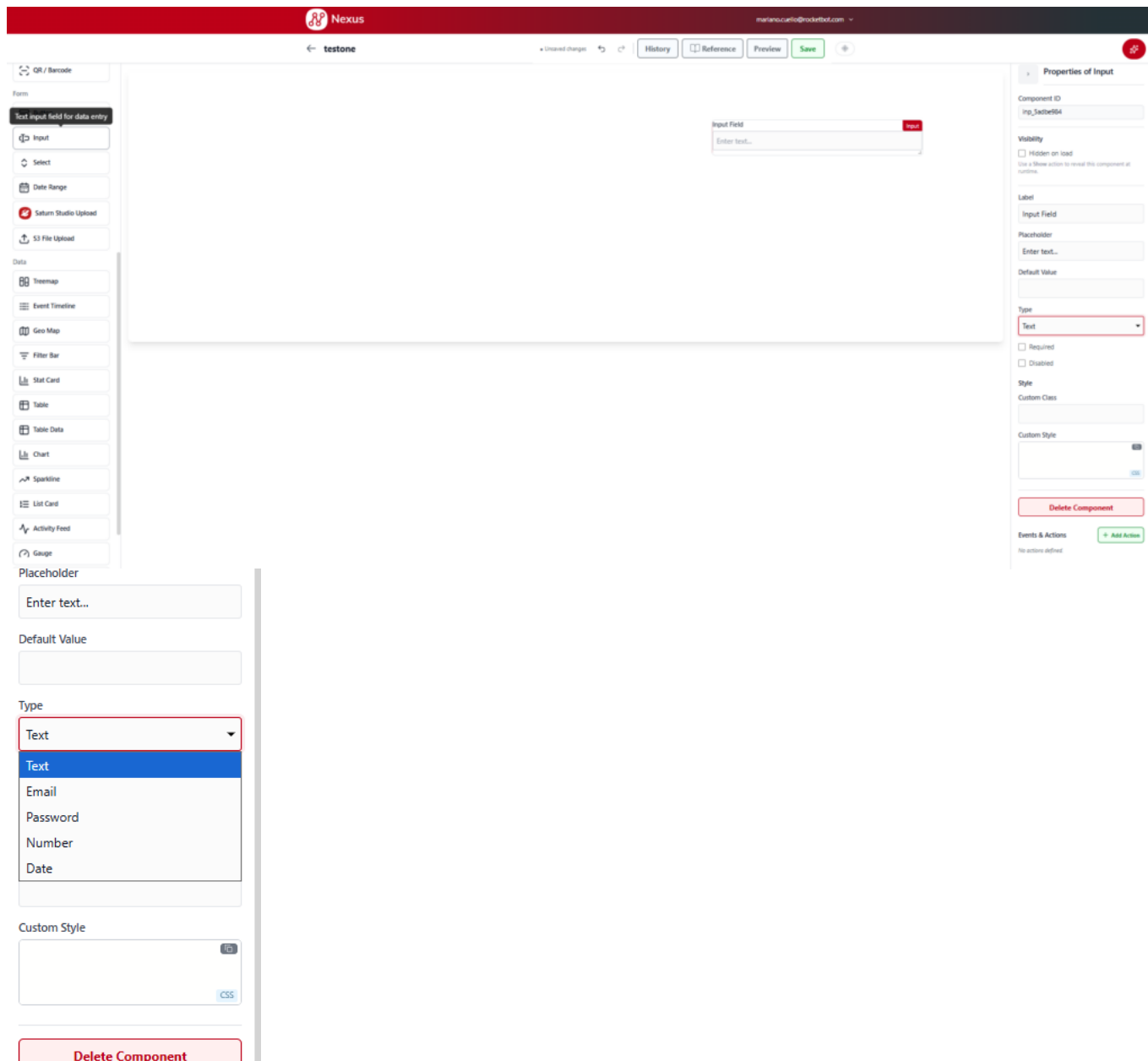
Input es un componente de tipo **Form**. Es un campo de entrada de texto para captura de datos del usuario, con distintos tipos de validación según el dato que se espera recibir. Al arrastrarlo al lienzo viene con un ejemplo precargado (“Input Field” con placeholder “Enter text...”).

A screenshot of a web component labeled "Input Field". It consists of a light gray rectangular box with a thin border. Inside the box, the text "Enter text..." is displayed in a light gray font, serving as a placeholder. The box has a subtle drop shadow.

Propiedades

Al seleccionar Input en el lienzo, el panel derecho muestra **“Properties of Input”** con las siguientes opciones:

- **Component ID:** identificador único del componente, autogenerado (ej. inp_5adbe984).
- **Visibility → Hidden on load:** casilla para que el componente esté oculto al cargar la pantalla. Puede mostrarse luego mediante una acción de tipo **Show** (ver Events & Actions).
- **Label:** etiqueta que se muestra encima del campo (ej. “Input Field”).
- **Placeholder:** texto de guía que aparece dentro del campo vacío (ej. “Enter text...”).
- **Default Value:** valor que trae el campo cargado por defecto.
- **Type:** menú desplegable que define el tipo de dato que acepta el campo, cambiando su validación y/o teclado. Opciones: **Text, Email, Password, Number, Date**.
- **Required:** casilla que marca el campo como obligatorio. Al activarla, se agrega un asterisco rojo (*) junto al Label, tanto en el editor como en el modo Preview, indicando visualmente que el campo es requerido.
- **Disabled:** casilla que deshabilita el campo (queda inactivo, no editable).
- **Style → Custom Class:** campo para asignar una clase CSS personalizada.
- **Style → Custom Style:** campo de texto para CSS personalizado aplicado directamente a este componente.
- **Delete Component:** elimina el componente del lienzo.
- **Events & Actions:** sistema para definir comportamientos interactivos. Ver documentación aparte: [Nexus – Events & Actions](#).



Ejemplo de uso

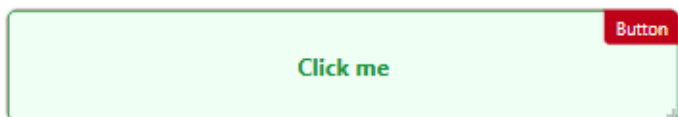
Con **Type** se adapta el campo al dato que se necesita capturar: por ejemplo, **Email** valida que el texto tenga formato de correo, **Password** oculta los caracteres ingresados, **Number** solo acepta valores numéricos, y **Date** muestra un selector de fecha. Activando **Required**, el campo pasa a ser obligatorio (probablemente impidiendo enviar un formulario si queda vacío). El valor ingresado por el usuario puede referenciarse luego desde otras partes de la app (queries, acciones) usando el **Component ID** del Input.

Nexus – Elemento: Button



¿Qué es?

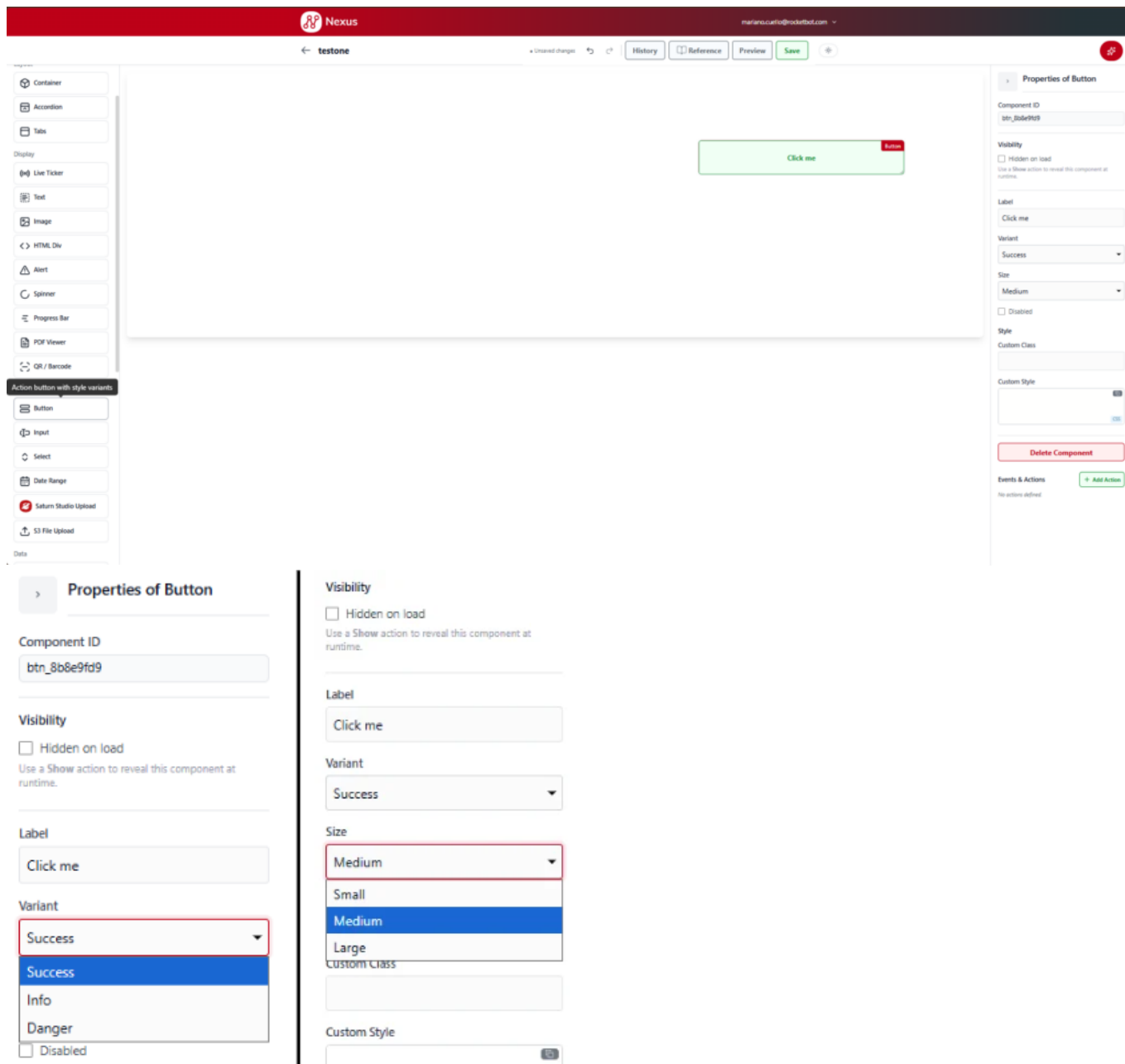
Button es un componente de tipo **Form**. Es un botón de acción con distintas variantes de estilo, pensado para disparar acciones (navegar, ejecutar queries, mostrar/ocultar componentes, etc.) mediante **Events & Actions**. Al arrastrarlo al lienzo viene con un texto de ejemplo precargado (“Click me”).



Propiedades

Al seleccionar Button en el lienzo, el panel derecho muestra “**Properties of Button**” con las siguientes opciones:

- **Component ID**: identificador único del componente, autogenerado (ej. btn_8b8e9fd9).
- **Visibility → Hidden on load**: casilla para que el componente esté oculto al cargar la pantalla. Puede mostrarse luego mediante una acción de tipo **Show** (ver Events & Actions).
- **Label**: texto que se muestra dentro del botón (ej. “Click me”).
- **Variant**: menú desplegable que define el color/estilo del botón. Opciones: **Success, Info, Danger**.
- **Size**: menú desplegable que define el tamaño del botón. Opciones: **Small, Medium, Large**.
- **Disabled**: casilla que deshabilita el botón (queda inactivo, sin poder hacer clic).
- **Style → Custom Class**: campo para asignar una clase CSS personalizada.
- **Style → Custom Style**: campo de texto para CSS personalizado aplicado directamente a este componente.
- **Delete Component**: elimina el componente del lienzo.
- **Events & Actions**: sistema para definir comportamientos interactivos (qué hace el botón al hacer clic). Ver documentación aparte: [Nexus – Events & Actions](#).



Ejemplo de uso

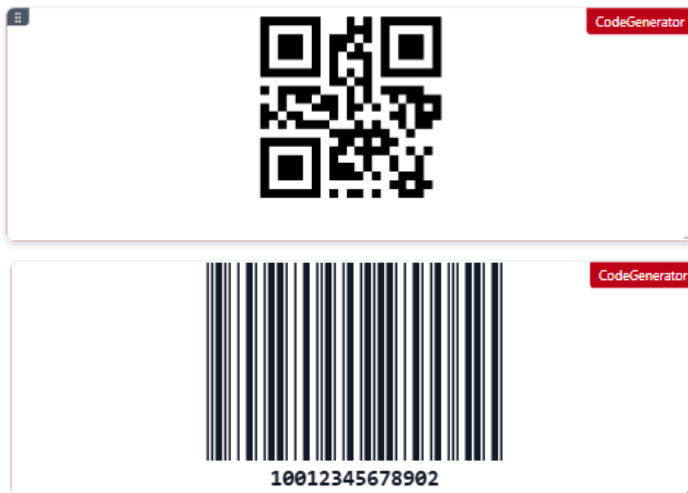
El uso típico de Button es combinarlo con **Events & Actions**, configurando un trigger **On Click** junto a la acción deseada (por ejemplo, **Navigate** para llevar a otra pantalla, o **Execute Query** para disparar una consulta de datos). Con **Variant** se puede dar significado visual a la acción: por ejemplo, **Success** para una acción de confirmación, o **Danger** para una acción destructiva como eliminar algo. Activando **Disabled**, el botón queda inhabilitado – útil para bloquearlo hasta que se cumpla alguna condición (por ejemplo, hasta que un formulario esté completo).

Nexus – Elemento: QR / Barcode



¿Qué es?

QR / Barcode es un componente de tipo **Display**. Genera códigos QR o códigos de barras a partir de un valor de texto, con distintos formatos de codificación. Al arrastrarlo al lienzo viene con un QR de ejemplo precargado, apuntando a una URL.

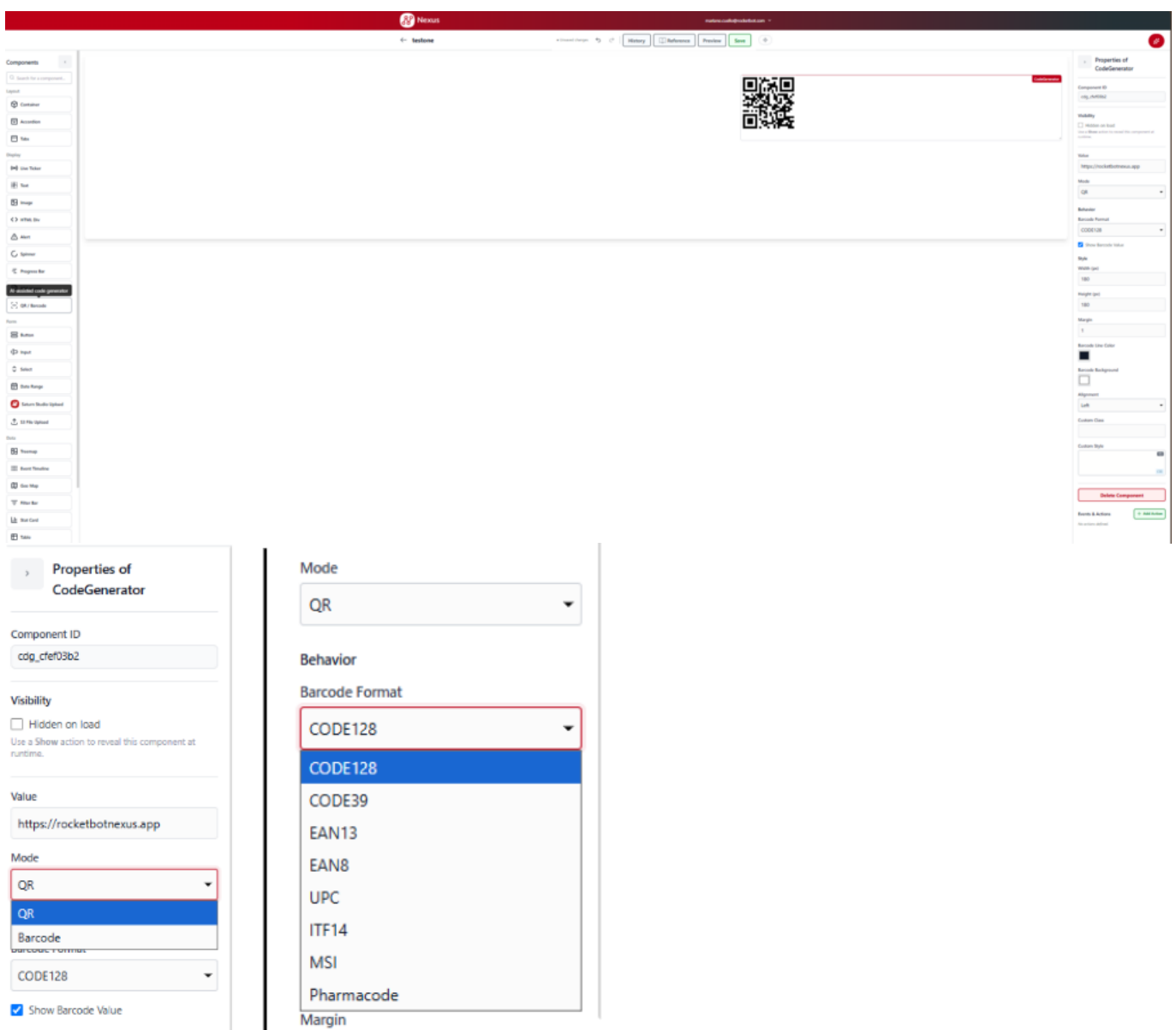


Propiedades

Al seleccionar el componente en el lienzo, el panel derecho muestra “**Properties of CodeGenerator**” con las siguientes opciones:

- **Component ID**: identificador único del componente, autogenerado (ej. `cdg_cfef03b2`).
- **Visibility → Hidden on load**: casilla para que el componente esté oculto al cargar la pantalla. Puede mostrarse luego mediante una acción de tipo **Show** (ver Events & Actions).
- **Value**: el texto o dato que se va a codificar (ej. una URL, un número, un código).
- **Mode**: menú desplegable que define si el componente genera un **QR** o un **Barcode** (código de barras lineal).
- **Behavior → Barcode Format** (*solo visible en modo Barcode*): define el estándar de codificación del código de barras. Opciones: **CODE128**, **CODE39**, **EAN13**, **EAN8**, **UPC**, **ITF14**, **MSI**, **Pharmacode**.
- **Show Barcode Value**: casilla que muestra el valor en texto debajo del código de barras.

- **Style → Width (px) / Height (px):** dimensiones del código generado (valor por defecto: 180×180).
- **Style → Margin:** margen alrededor del código.
- **Style → Barcode Line Color:** color de las líneas/módulos del código.
- **Style → Barcode Background:** color de fondo del código.
- **Style → Alignment:** alineación del componente. Opciones: **Left, Center, Right.**
- **Style → Custom Class:** clase CSS personalizada.
- **Style → Custom Style:** CSS personalizado aplicado directamente a este componente.
- **Delete Component:** elimina el componente del lienzo.
- **Events & Actions:** sistema para definir comportamientos interactivos. Ver documentación aparte: [Nexus – Events & Actions](#) .



Ejemplo de uso – formatos de Barcode

Cada formato de código de barras tiene reglas propias sobre qué valores acepta (algunos piden solo números, otros una cantidad exacta de dígitos). Se

probó cada formato con un valor de ejemplo y funcionaron correctamente:

Formato	Uso típico	Valor de prueba usado
CODE128	Genérico, acepta letras, números y símbolos. Muy usado en logística.	Nexus-2026
CODE39	Alfanumérico (mayúsculas, números y algunos símbolos). Usado en industria/gobierno.	NEXUS123
EAN13	Código de producto de retail (13 dígitos numéricos exactos).	4006381333931
EAN8	Versión corta de EAN, para productos pequeños (8 dígitos numéricos).	96385074
UPC	Estándar de retail en EE.UU./Canadá (12 dígitos numéricos).	036000291452
ITF14	Usado en cajas/embalajes de logística (14 dígitos numéricos).	10012345678902
MSI	Usado en inventario y estanterías (solo números).	1234567
Pharmacode	Usado en packaging farmacéutico (número simple, sin chequeo).	1234

Para probar cualquier formato: cambiar **Mode** a Barcode, elegir el formato deseado en **Barcode Format**, y pegar en **Value** un dato acorde a ese estándar (por ejemplo, para **EAN13** debe ser un número de 13 dígitos, para **UPC** de 12 dígitos, y para **Pharmacode** simplemente un número simple sin formato especial).

Nexus – Elemento: PDF Viewer



¿Qué es?

PDF Viewer es un componente de tipo **Display**. Muestra un visor embebido de archivos PDF, con navegación entre páginas. Al arrastrarlo al lienzo viene con un PDF de ejemplo precargado (documento de 14 páginas).

Trace-based Just-in-Time Type Specialization for Dynamic Languages

Andreas Gal¹, Brendan Eich², Mike Shaver³, David Anderson⁴, David Mandelin⁵,
 Mohammad R. Haghighat⁶, Blake Kaplan⁷, Graydon Hoare⁸, Boris Zharsky⁹, Jason Orendorff¹,
 Jesse Ruderman¹, Edwin Smith¹⁰, Rick Reitmaier¹¹, Michael Hebenitz¹, Mason Chang¹, Michael Franz¹

Mozilla Corporation^{*}
 {gal, brendan, shaver, danderson, dmandelin, mrbkap, graydon, bz, jorendorff, jruderman}@mozilla.com

Adobe Corporation²
 {edsmith, rreitmai}@adobe.com

Intel Corporation³
 {mohammad.r.haghighat}@intel.com

University of California, Irvine⁴
 {mhebenitz, changm, franz}@uci.edu

Abstract

Dynamic languages such as JavaScript are more difficult to compile than statically typed ones. Since no concrete type information is available, traditional compilers need to emit generic code that can handle all possible type combinations at runtime. We present an alternative compilation technique for dynamically-typed languages that identifies frequently executed loop traces at run time and then generates machine code on the fly that is specialized for the actual dynamic types occurring on each path through the loop. Our method provides cheap in-line procedural type specialization, and an elegant and efficient way of incrementally compiling heavily discovered alternative paths through nested loops. We have implemented a dynamic compiler for JavaScript based on our technique and we have measured speedups of 10x and more for certain benchmark programs.

Categories and Subject Descriptors: D.3.4 [Programming Languages]: Processors — Incremental compilers, code generation.
General Terms: Design, Experimentation, Measurement, Performance.

Keywords: JavaScript, just-in-time compilation, trace trace.

1. Introduction

Dynamic languages such as JavaScript, Python, and Ruby, are popular since they are expressive, accessible to non-experts, and make deployment as easy as distributing a source file. They are used for small scripts as well as for complex applications. JavaScript, for example, is the de facto standard for client-side web programming

and is used for the application logic of browser-based productivity applications such as Google Mail, Google Docs and Zimbra Collaboration Suite. In this domain, in order to provide a fluid user experience and enable a new generation of applications, virtual machines must provide a low startup time and high performance.

Compilers for statically typed languages rely on type information to generate efficient machine code. In a dynamically typed programming language such as JavaScript, the types of expressions may vary at runtime. This means that the compiler can no longer easily transform operations into machine instructions that operate on one specific type. Without exact type information, the compiler must emit slower generalized machine code that can deal with all potential type combinations. While compile-time static type inference might be able to gather type information to generate optimized machine code, traditional static analysis is very expensive and hence not well suited for the highly interactive environments of a web browser.

We present a trace-based compilation technique for dynamic languages that reconciles speed of compilation with excellent performance of the generated machine code. Our system uses a mixed-mode execution approach: the system starts running JavaScript in a fast-starting bytecode interpreter. As the program runs, the system identifies hot (frequently executed) bytecode sequences, records them, and compiles them to fast native code. We call such a sequence of instructions a *trace*.

Unlike method-based dynamic compilers, our dynamic compiler operates at the granularity of individual loops. This design choice is based on the expectation that programs spend most of their time in hot loops. Even in dynamically typed languages, we expect hot loops to be mostly type stable, meaning that the types of values are invariant. (12) For example, we would expect loop counters that start as integers to remain integers for all iterations. When both of these expectations hold, a trace-based compiler can cover the program execution with a small number of type-specialized, efficiently compiled traces.

Each compiled trace covers one path through the program with one mapping of values to types. When the VM executes a compiled trace, it cannot guarantee that the same path will be followed or that the same types will occur in subsequent loop iterations.

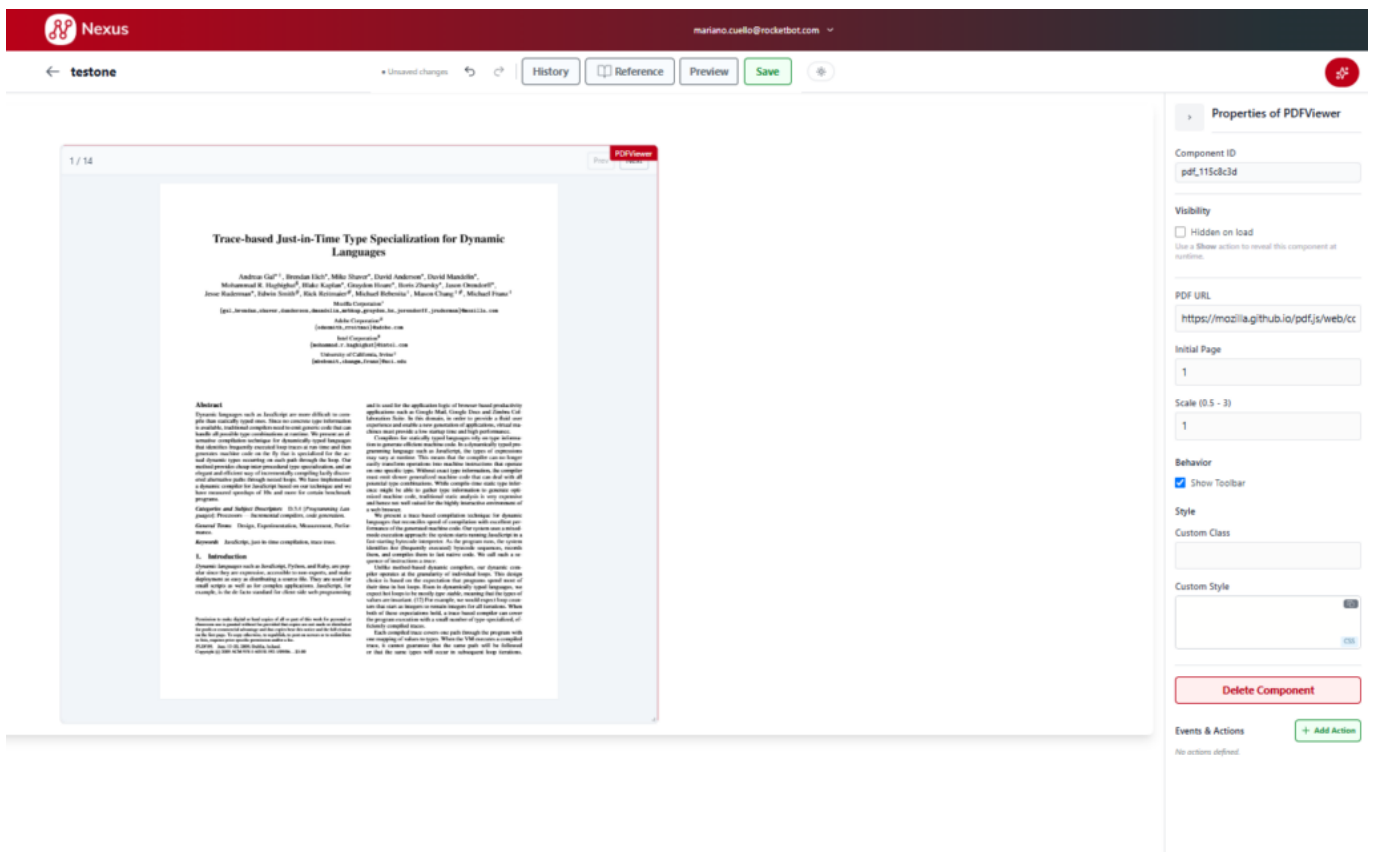
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or fee.
 PLDI'09, June 15–20, 2009, Dublin, Ireland.
 Copyright © 2009 ACM 978-1-60558-190-0...\$5.00

Propiedades

Al seleccionar PDF Viewer en el lienzo, el panel derecho muestra “**Properties of PDFViewer**” con las siguientes opciones:

- **Component ID:** identificador único del componente, autogenerado (ej. pdf_115c8c3d).
- **Visibility → Hidden on load:** casilla para que el componente esté oculto al cargar la pantalla. Puede mostrarse luego mediante una acción de tipo **Show** (ver Events & Actions).
- **PDF URL:** campo donde se coloca la URL del archivo PDF a mostrar.
- **Initial Page:** número de página con la que se abre el documento por defecto (ej. 1).
- **Scale (0.5 – 3):** nivel de zoom/escala con el que se muestra el PDF, en un rango de 0.5 a 3 (valor por defecto: 1).

- **Behavior → Show Toolbar:** casilla que muestra u oculta la barra de herramientas del visor (indicador de página, botones Prev/Next).
- **Style → Custom Class:** campo para asignar una clase CSS personalizada.
- **Style → Custom Style:** campo de texto para CSS personalizado aplicado directamente a este componente.
- **Delete Component:** elimina el componente del lienzo.
- **Events & Actions:** sistema para definir comportamientos interactivos. Ver documentación aparte: [Nexus – Events & Actions](#).



Ejemplo de uso

Para mostrar un PDF propio, se reemplaza **PDF URL** por la dirección del archivo (debe ser una URL pública accesible, terminada en .pdf). Con **Initial Page** se puede hacer que el documento se abra directamente en una página específica en vez de la primera, y con **Scale** se ajusta el zoom inicial del documento.

[Nexus – Elemento: Progress Bar](#)



¿Qué es?

Progress Bar es un componente de tipo **Display**. Muestra una barra de progreso con porcentaje, útil para representar el avance de un proceso. Al arrastrarlo al lienzo viene con un valor de ejemplo precargado (60 de 100).



Propiedades

Al seleccionar Progress Bar en el lienzo, el panel derecho muestra “**Properties of ProgressBar**” con las siguientes opciones:

- **Component ID**: identificador único del componente, autogenerado (ej. prg_b9ce0397).
- **Visibility → Hidden on load**: casilla para que el componente esté oculto al cargar la pantalla. Puede mostrarse luego mediante una acción de tipo **Show** (ver Events & Actions).
- **Value**: valor actual de la barra (ej. 60).
- **Max**: valor máximo que representa el 100% de la barra (ej. 100).
- **Label**: texto opcional para acompañar la barra (vacío por defecto).
- **Show Value**: casilla que muestra u oculta el porcentaje/valor numérico sobre la barra.
- **Variant**: menú desplegable que define el color de la barra. Opciones: **Primary, Success, Warning, Danger, Info**.
- **Size**: menú desplegable que define el grosor de la barra. Opciones: **Small, Medium, Large**.
- **Animated**: casilla que hace que la barra se rellene con una animación progresiva hasta llegar al valor definido en **Value**, en vez de mostrarse directamente ya completa.
- **Style → Custom Class**: campo para asignar una clase CSS personalizada.
- **Style → Custom Style**: campo de texto para CSS personalizado aplicado directamente a este componente.
- **Delete Component**: elimina el componente del lienzo.
- **Events & Actions**: sistema para definir comportamientos interactivos. Ver documentación aparte: [Nexus – Events & Actions](#) .

Properties of ProgressBar

Component ID

prg_7f9ed5af

Visibility

☐ Hidden on load

Use a **Show** action to reveal this component at runtime.

Value

60

Max

100

Label

☒ Show Value

Variant

Primary

Size

Medium

☐ Animated

Style

Custom Class

Custom Style

Delete Component

Events & Actions

No actions defined.

+ Add Action

• Unsaved changes

History

Reference

Preview

Save

ProgressBar

Properties of ProgressBar

Component ID
prg_7f9ad5af

Visibility
☐ Hidden on load
Use a Show action to reveal this component at runtime.

Value
60

Max
100

Label

☒ Show Value

Variant
Warning
Primary
Success
Warning
Danger
Info
Custom Class

Custom Style
CSS

Delete Component

Events & Actions
No actions defined.
+ Add Action

Ejemplo de uso

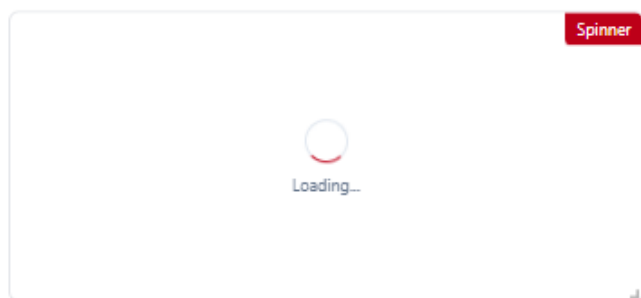
El progreso se controla ajustando **Value** en relación a **Max** (por ejemplo, Value: 60 y Max: 100 representa un 60%). Con **Variant** se puede dar significado visual al progreso – por ejemplo, usar **Success** cuando el proceso llega al 100%, o **Danger** si representa un límite excedido. Activando **Animated**, la barra se llena de forma progresiva (efecto de carga) hasta alcanzar el valor definido, en lugar de mostrarse ya completa desde el inicio.

Nexus – Elemento: Spinner



¿Qué es?

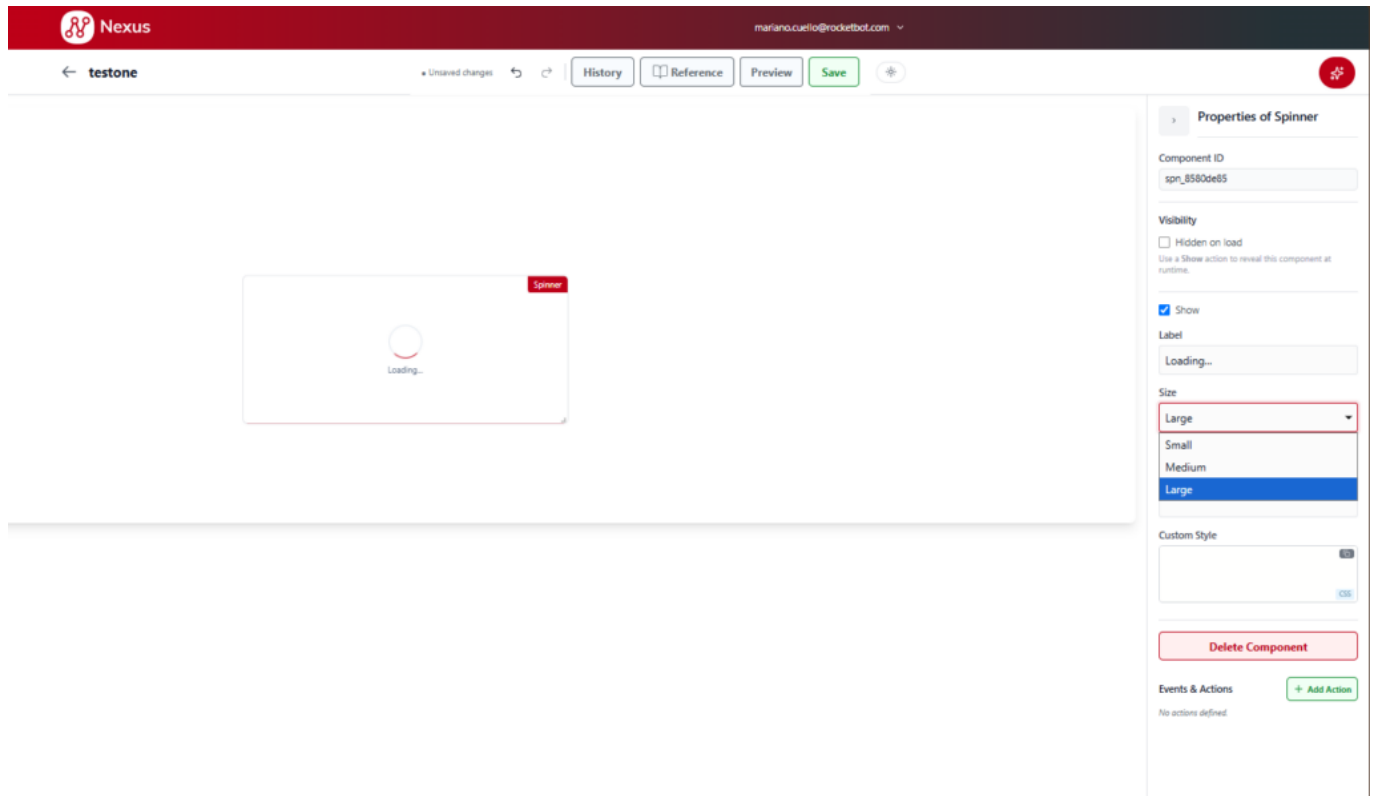
Spinner es un componente de tipo **Display**. Muestra un indicador de carga animado, útil para representar visualmente que un proceso está en curso. Al arrastrarlo al lienzo viene con un texto de ejemplo precargado (“Loading…”).



Propiedades

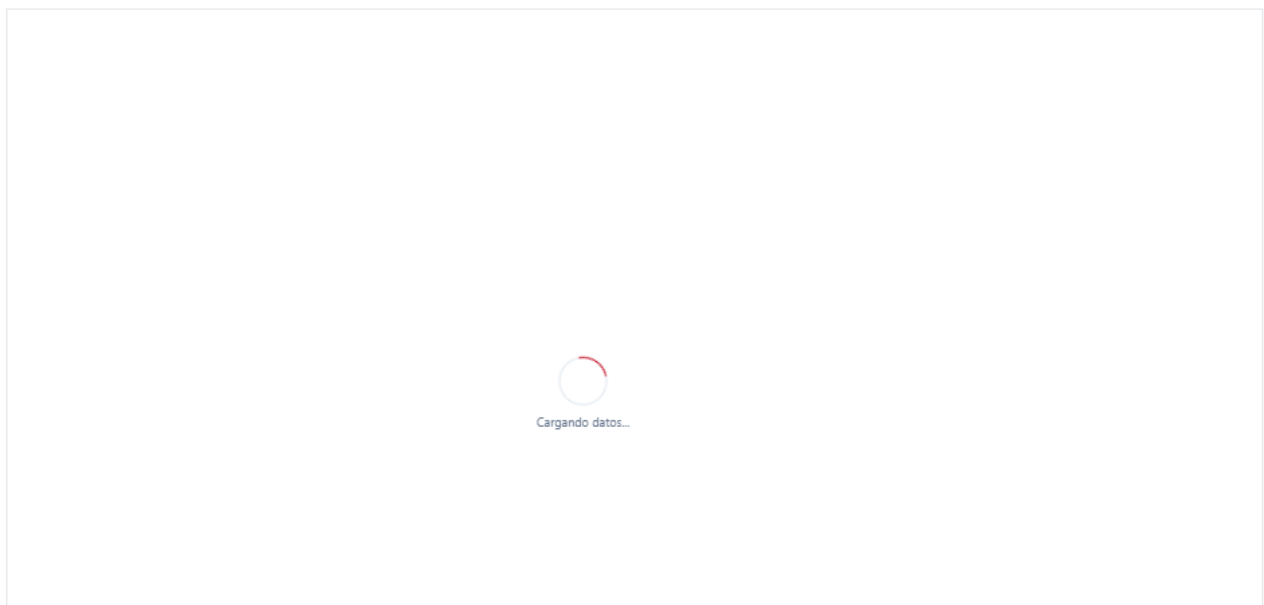
Al seleccionar Spinner en el lienzo, el panel derecho muestra **“Properties of Spinner”** con las siguientes opciones:

- **Component ID**: identificador único del componente, autogenerado (ej. spn_8580de85).
- **Visibility → Hidden on load**: casilla para que el componente esté oculto al cargar la pantalla. Puede mostrarse luego mediante una acción de tipo **Show** (ver Events & Actions).
- **Show**: casilla que controla si el spinner se muestra o no.
- **Label**: texto que acompaña a la animación de carga (ej. “Loading…”).
- **Size**: menú desplegable que define el tamaño de la animación. Opciones: **Small, Medium, Large**.
- **Style → Custom Class**: campo para asignar una clase CSS personalizada.
- **Style → Custom Style**: campo de texto para CSS personalizado aplicado directamente a este componente.
- **Delete Component**: elimina el componente del lienzo.
- **Events & Actions**: sistema para definir comportamientos interactivos. Ver documentación aparte: [Nexus – Events & Actions](#) .



Ejemplo de uso

El Spinner suele usarse en combinación con **Events & Actions** para mostrarse mientras se ejecuta un proceso (por ejemplo, mientras se espera la respuesta de una API o query) y ocultarse una vez que ese proceso termina, usando acciones de tipo **Show Components** / **Hide Components** sobre otros componentes. Con **Label** se puede personalizar el texto que acompaña la animación (ej. "Cargando datos...", "Procesando...").

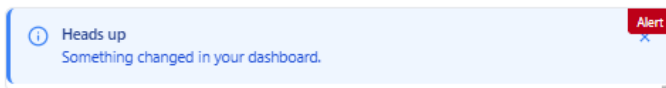


Nexus – Elemento: Alert



¿Qué es?

Alert es un componente de tipo **Display**. Muestra un mensaje de aviso o notificación con distintos niveles de severidad, cada uno con su propio color e ícono. Al arrastrarlo al lienzo viene con un mensaje de ejemplo precargado (“Heads up / Something changed in your dashboard”).

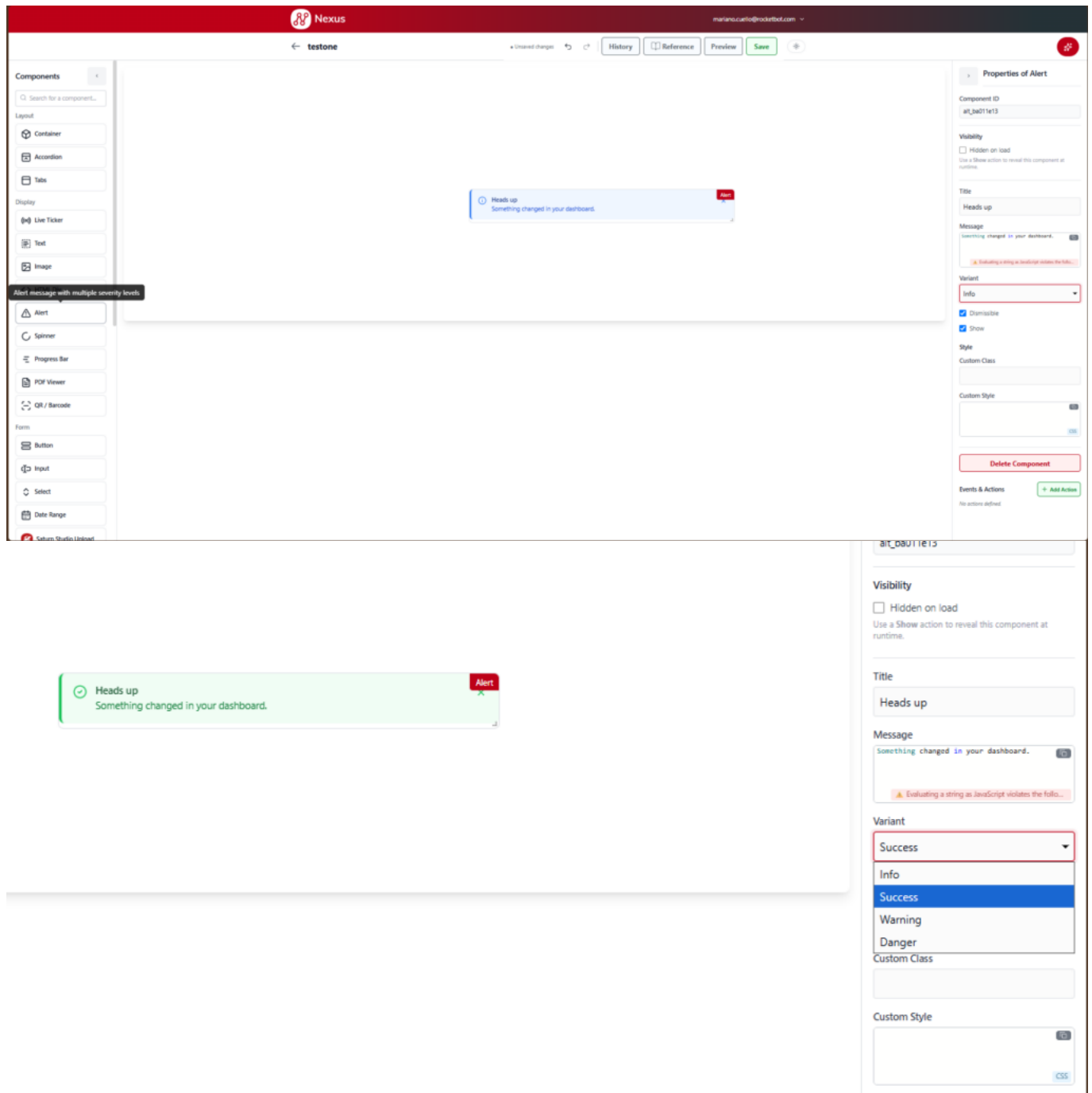


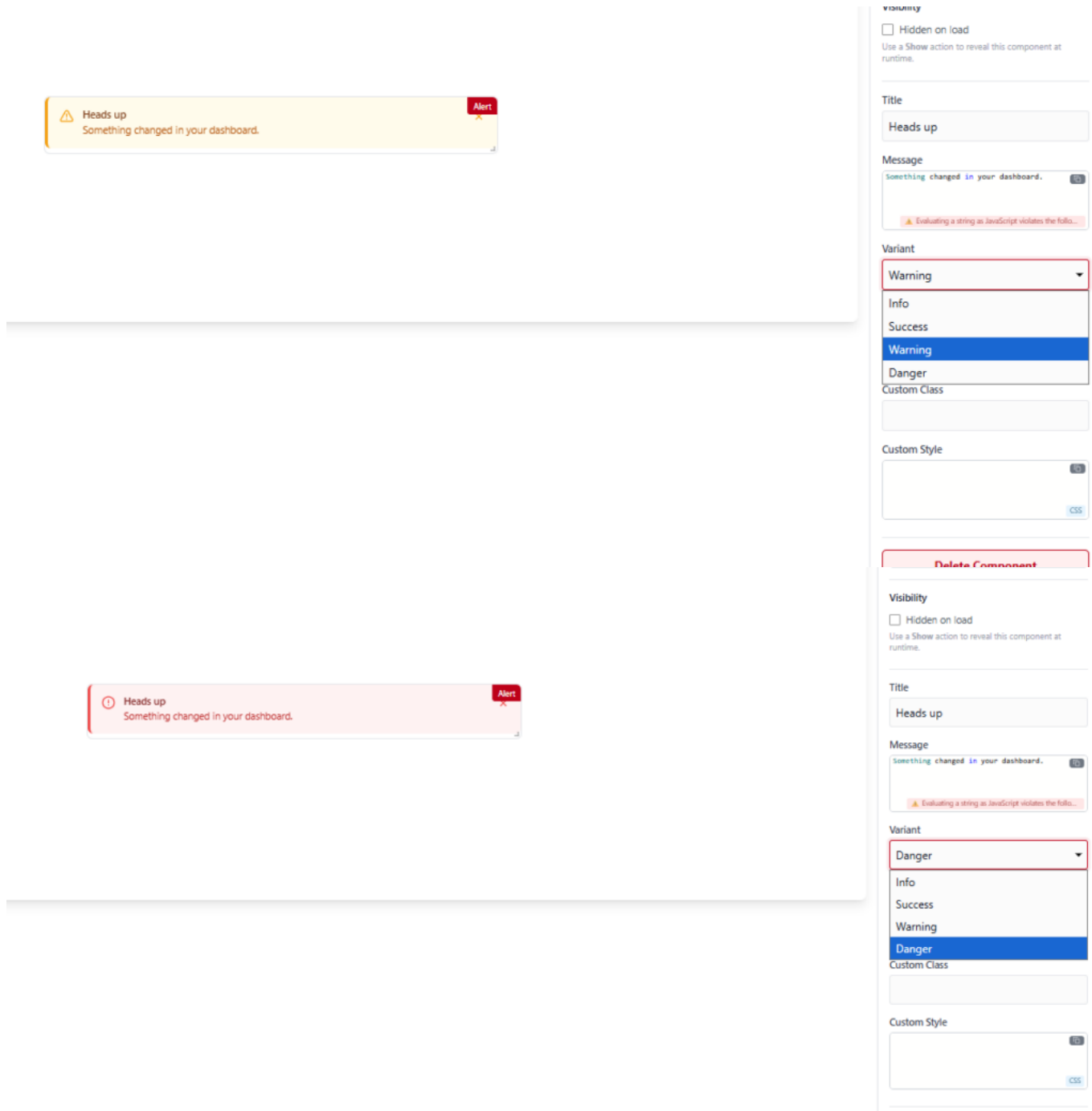
Propiedades

Al seleccionar Alert en el lienzo, el panel derecho muestra “**Properties of Alert**” con las siguientes opciones:

- **Component ID**: identificador único del componente, autogenerado (ej. alt_ba011e13).
- **Visibility → Hidden on load**: casilla para que el componente esté oculto al cargar la pantalla. Puede mostrarse luego mediante una acción de tipo **Show** (ver Events & Actions).
- **Title**: título del mensaje de alerta (ej. “Heads up”).
- **Message**: texto del cuerpo del mensaje (ej. “Something changed in your dashboard.”).
- **Variant**: menú desplegable que define el tipo/severidad de la alerta, cambiando color e ícono. Opciones:
 - **Info**: alerta informativa, en color azul.
 - **Success**: alerta de éxito, en color verde.
 - **Warning**: alerta de advertencia, en color amarillo/naranja.
 - **Danger**: alerta de error/peligro, en color rojo.
- **Dismissible**: casilla que habilita un botón (ícono x) para que el usuario pueda cerrar la alerta manualmente.
- **Show**: casilla que controla si la alerta se muestra o no.
- **Style → Custom Class**: campo para asignar una clase CSS personalizada.
- **Style → Custom Style**: campo de texto para CSS personalizado aplicado directamente a este componente.

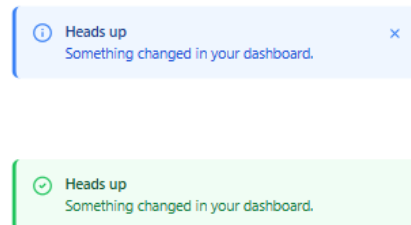
- **Delete Component:** elimina el componente del lienzo.
- **Events & Actions:** sistema para definir comportamientos interactivos. Ver documentación aparte: [Nexus – Events & Actions](#) .





Ejemplo de uso

Se puede usar **Variant** para comunicar distintos tipos de mensajes según el contexto: por ejemplo, **Success** al confirmar que una acción se completó correctamente, **Warning** para advertir sobre algo que requiere atención, o **Danger** para señalar un error. Con **Dismissible** activado, el usuario puede cerrar la alerta haciendo clic en el ícono ×; si se desactiva, la alerta permanece visible de forma fija mientras **Show** esté activo.



Nexus – Elemento: HTML Div



¿Qué es?

HTML Div es un componente de tipo **Display**. Es un contenedor genérico de HTML, que permite insertar código HTML personalizado directamente dentro de la pantalla. Al arrastrarlo al lienzo viene con un contenido de ejemplo precargado (“HTML content here”).

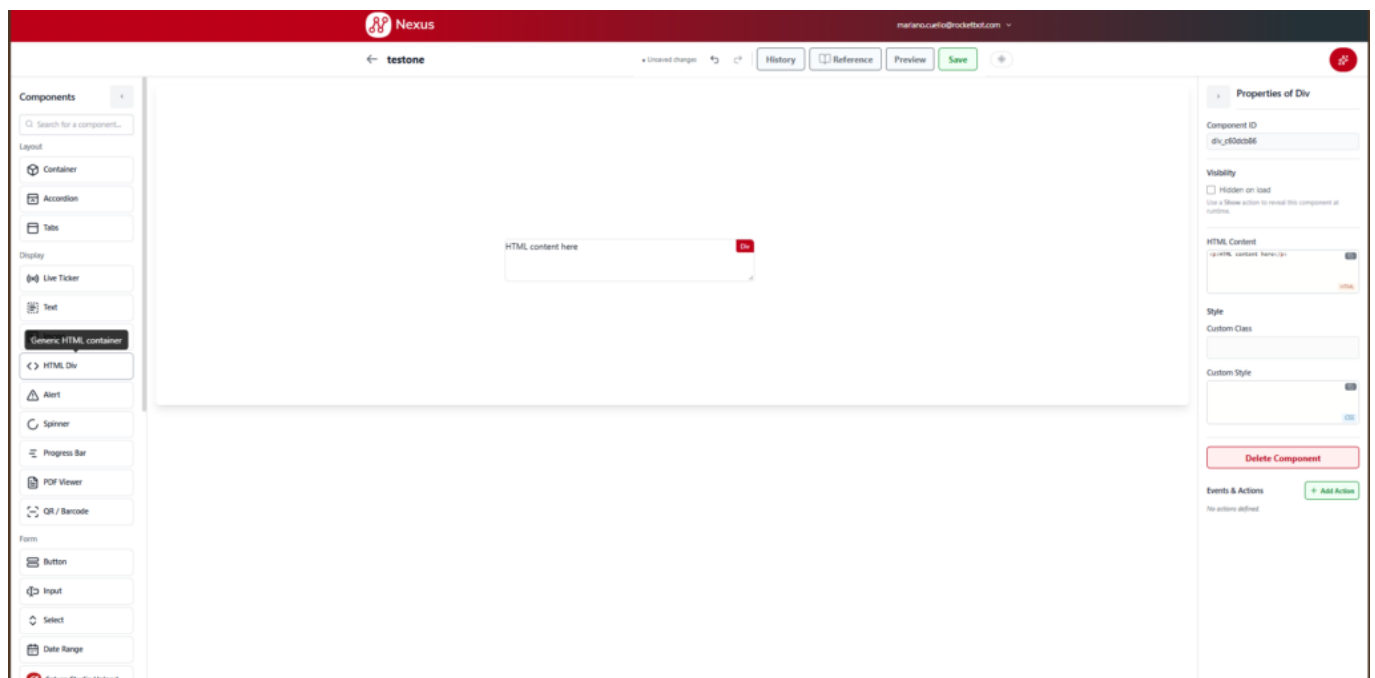
HTML content here

Div

Propiedades

Al seleccionar HTML Div en el lienzo, el panel derecho muestra “**Properties of Div**” con las siguientes opciones:

- **Component ID**: identificador único del componente, autogenerado (ej. div_c60dcb86).
- **Visibility → Hidden on load**: casilla para que el componente esté oculto al cargar la pantalla. Puede mostrarse luego mediante una acción de tipo **Show** (ver Events & Actions).
- **HTML Content**: campo de código donde se escribe el contenido en HTML directamente (ej. <p>HTML content here</p>).
- **Style → Custom Class**: campo para asignar una clase CSS personalizada.
- **Style → Custom Style**: campo de texto para CSS personalizado aplicado directamente a este componente.
- **Delete Component**: elimina el componente del lienzo.
- **Events & Actions**: sistema para definir comportamientos interactivos. Ver documentación aparte: [Nexus – Events & Actions](#).



Ejemplo de uso

El contenido se escribe directamente en HTML dentro de **HTML Content**. Por ejemplo:

```
html
```

```
<p>Este es un párrafo con <strong>texto en negrita</strong> y un <a href="https://www.rocketbot.com">link</a>.</p>
```

A diferencia del componente **Text** (que usa Markdown), HTML Div permite

escribir HTML puro, útil para estructuras más complejas o personalizadas que no se logran fácilmente con Markdown.

Unsaved changes ↶ ↷

History

Reference

Preview

Save

⚙️

Div

Este es un párrafo con **texto en negrita** y un [link](#).

Properties of Div

Component ID

div_cf0dcb86

Visibility

☐ Hidden on load

Use a Show action to reveal this component at runtime.

HTML Content

```
<p>Este es un párrafo con <strong>texto en negrita</strong> y un <a href="https://www.rocketbot.com">link</a>.</p>
```

Style

Custom Class

Custom Style

Delete Component

Events & Actions

+ Add Action

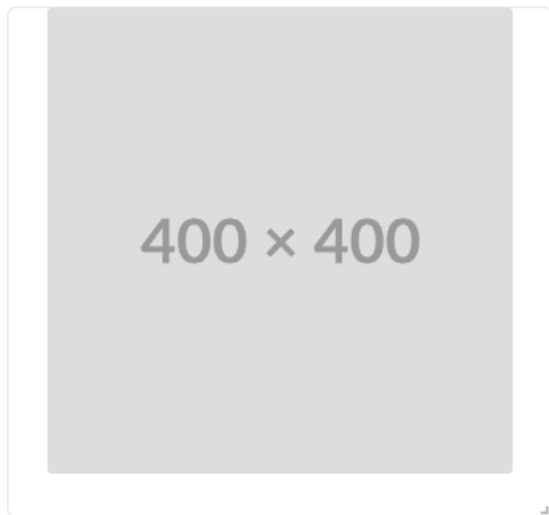
No actions defined.

Nexus – Elemento: Image



¿Qué es?

Image es un componente de tipo **Display**. Permite insertar una imagen con URL configurable. Al arrastrarlo al lienzo viene con una imagen de ejemplo tipo placeholder (400×400).



Propiedades

Al seleccionar Image en el lienzo, el panel derecho muestra **“Properties of Image”** con las siguientes opciones:

- **Component ID:** identificador único del componente, autogenerado (ej. `img_ba0309b3`).
- **Visibility → Hidden on load:** casilla para que el componente esté oculto al cargar la pantalla. Puede mostrarse luego mediante una acción de tipo **Show** (ver Events & Actions).
- **Image URL:** campo donde se coloca la URL de la imagen a mostrar (ej. `https://placeholder.co/400`).
- **Alt Text:** texto alternativo de la imagen (accesibilidad / se muestra si la imagen no carga).
- **Variant:** menú desplegable que define el tamaño con el que se muestra la imagen dentro de su espacio disponible. Opciones:
 - **Thumbnail:** la imagen se muestra en tamaño reducido, tipo miniatura.
 - **Body:** la imagen se muestra en un tamaño intermedio, como parte del contenido normal de la pantalla.
 - **Header:** la imagen se muestra ocupando el ancho completo disponible, como una imagen de cabecera.
- **Align:** menú desplegable que define la alineación de la imagen dentro de su espacio. Opciones: **Left, Center, Right**.
- **Style → Custom Class:** campo para asignar una clase CSS personalizada.
- **Style → Custom Style:** campo de texto para CSS personalizado aplicado directamente a este componente.
- **Delete Component:** elimina el componente del lienzo.
- **Events & Actions:** sistema para definir comportamientos interactivos. Ver documentación aparte: [Nexus – Events & Actions](#) .

testone

Unsaved changes

History

Reference

Preview

Save

Properties of Image

Component ID
img_ba3309b3

Visibility
☐ Hidden on load
Use a Show action to reveal this component at runtime.

Image URL
https://placeholder.co/400

Alt Text
Image

Variant
Thumbnail

Align
Center

Style
Custom Class

Custom Style

Delete Component

Events & Actions
[+ Add Action](#)
No actions defined.

Nexus

mariano.cuello@rocketbot.com

testone

Unsaved changes

History

Reference

Preview

Save

Properties of Image

Component ID
img_ba3309b3

Visibility
☐ Hidden on load
Use a Show action to reveal this component at runtime.

Image URL
https://placeholder.co/400

Alt Text
Image

Variant
Body

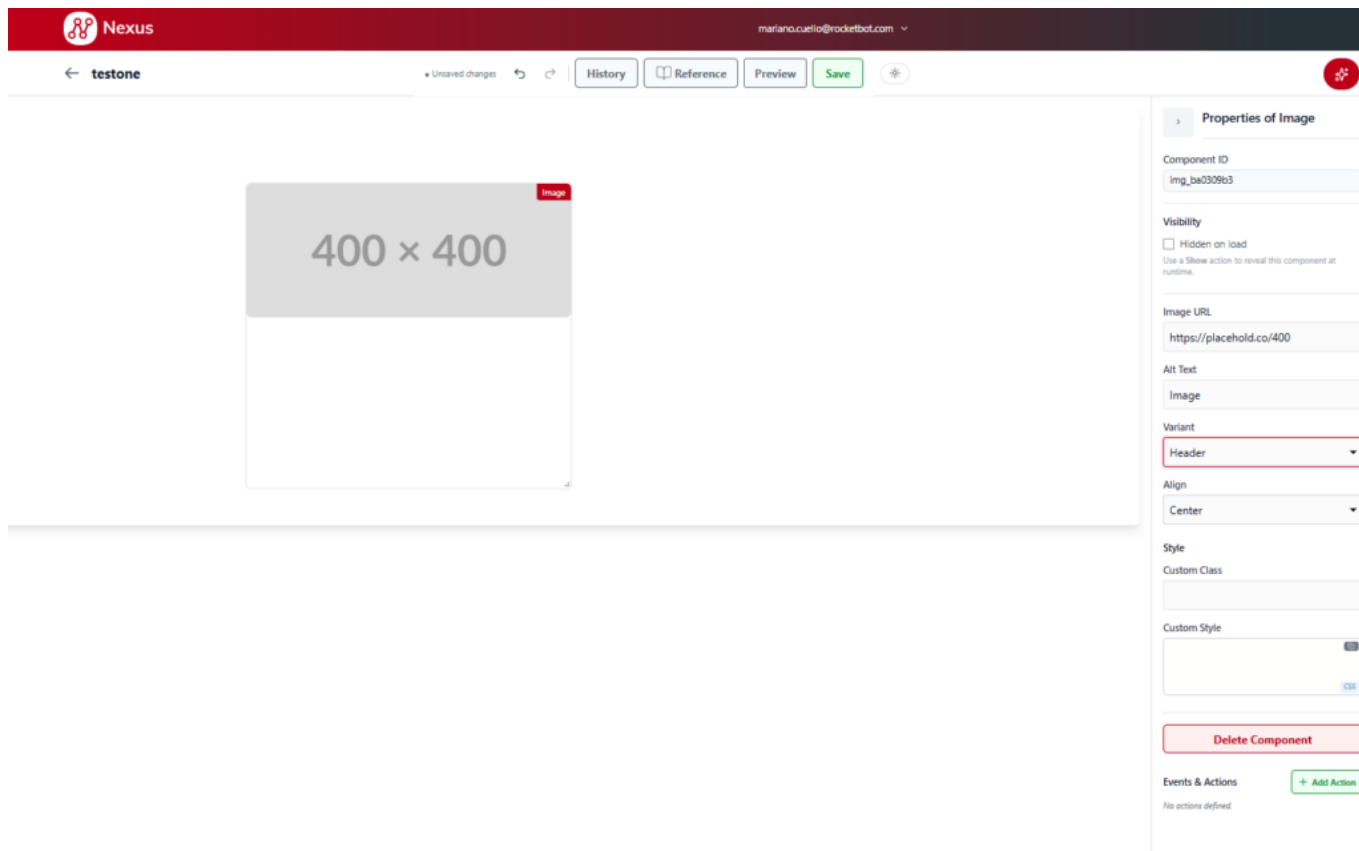
Align
Center

Style
Custom Class

Custom Style

Delete Component

Events & Actions
[+ Add Action](#)
No actions defined.

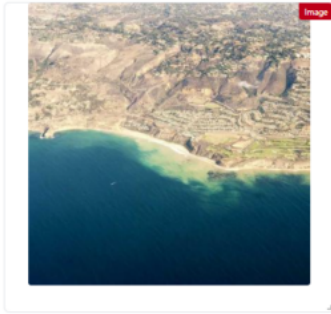


Ejemplo de uso

Para mostrar una imagen propia, se reemplaza el valor de **Image URL** por la dirección de la imagen deseada (debe ser una URL pública accesible). Por ejemplo, se puede usar una imagen libre de copyright de prueba:

<https://picsum.photos/400/400>

Con **Variant** se ajusta el tamaño según el uso: **Thumbnail** para íconos o miniaturas, **Body** para imágenes dentro del contenido, y **Header** para imágenes destacadas de ancho completo. Con **Align** se controla si queda alineada a la izquierda, centrada o a la derecha.



Properties of Image

Component ID

img_ba0309b3

Visibility

☐ Hidden on load

Use a Show action to reveal this component at runtime.

Image URL

<https://picsum.photos/400/400>

Alt Text

Image

Variant

Body

Align

Center

Style

Custom Class

Custom Style

Delete Component

Events & Actions

+ Add Action

No actions defined.