

Introducción

Empezando

Componentes

Alert

Badge

Button

Detail

Dialog

Dropdown

Errors

Filters

[Input](#)

Loading

Pagination

ProgressBar

Select

Tab

Table

TextareaResize

Toast

Utilidades

useAction

useFormat

useInputMask

useKeyypress

useTable

useTheme

useTooltip

useValidator

Composables

useFilters

useLogos

Input

Documentación y ejemplos del componente [input](#). Para **crear formularios** de inserción de un registro.

Props

inputType

Type: [String](#)

Define el **tipo de input** que se necesita, ejemplo [text](#), [password](#), [email](#) etc.

```
<InputComponent  
    inputType="text"  
/>
```

html

placeHolder

Type: [String](#)

Define el **texto descriptivo** dentro del componente.

Esto es un placeholder

html

```
<InputComponent  
    inputType="text"  
    placeHolder="Esto es un placeholder"  
/>
```

maxLength

Type: [String](#)

Define el **largo máximo de la información** en el componente. Valor por defecto [50](#).

Introducción

Empezando

Componentes

Alert

Badge

Button

Detail

Dialog

Dropdown

Errors

Filters

Input

Loading

Pagination

ProgressBar

Select

Tab

Table

TextareaResize

Toast

Utilidades

useAction

useFormat

useInputMask

useKeypress

useTable

useTheme

useTooltip

useValidator

Composables

useFilters

useLogos

```
<InputComponent  
    inputType="text"  
    placeHolder="Ingresa un valor"  
    maxLength="10"  
/>
```

html

isDisabled

Type: Boolean

Esta propiedad **bloquea el componente** e impide modificar su contenido.
Valor por defecto `false`.

Ingrresa un valor

```
<InputComponent  
    inputType="text"  
    placeHolder="Ingresa un valor"  
    :isDisabled="true"  
/>
```

html

floatLabel

Type: String

Propiedad que combina el placeholder y label y lo posiciona en la parte superior al hacer focus.

Ingrresa un valor

```
<InputComponent  
    inputType="text"  
    floatLabel="Ingresa un valor"  
/>
```

html

Importante

Esta propiedad inhabilita la propiedad `placeHolder`.

floatLeft

Type: Boolean

Introducción

Empezando

Componentes

Alert

Badge

Button

Detail

Dialog

Dropdown

Errors

Filters

Input

Loading

Pagination

ProgressBar

Select

Tab

Table

TextareaResize

Toast

Utilidades

useAction

useFormat

useInputMask

useKeypress

useTable

useTheme

useTooltip

useValidator

Composables

useFilters

useLogos

Ingresa una dirección

html

```
<InputComponent  
    inputType="text"  
    floatLeft  
    :maxLength="20"  
    placeHolder="Ingresa una dirección"  
>  
<template #float-left>  
    <font-awesome-icon  
        :icon="['fas', 'location-dot']"  
        data-eit-variant="gray"  
    />  
</template>  
</InputComponent>
```

floatRight

Type: Boolean

Propiedad pensada para activar el slot llamado `#float-right` y agregar un ícono a la derecha del input. Se recomienda usar la librería [fontawesome](#)

Ingresa una dirección



html

```
<InputComponent  
    inputType="text"  
    floatRight  
    :maxLength="20"  
    placeHolder="Ingresa una dirección"  
>  
<template #float-right>  
    <font-awesome-icon  
        :icon="['fas', 'location-dot']"  
        data-eit-variant="gray"  
    />  
</template>  
</InputComponent>
```

input

Type: [String, Number]

Se define para agregar la información entrante al componente al renderizar el DOM. Principalmente usado en la edición de registros.

Introducción

Empezando

Componentes

Alert

Badge

Button

Detail

Dialog

Dropdown

Errors

Filters

Input

Loading

Pagination

ProgressBar

Select

Tab

Table

TextareaResize

Toast

Utilidades

useAction

useFormat

useInputMask

useKeyypress

useTable

useTheme

useTooltip

useValidator

Composables

useFilters

useLogos

```
vue
<script setup lang="ts">
  import { ref } from 'vue'

  const inputProps = ref('Texto ejemplo')
</script>

<template>
  <InputComponent
    inputType="text"
    :input="inputProps"
  />
</template>
```

requiredField

Type: Boolean

Define esta propiedad **si el campo debe estar lleno** al momento de realizar `submit`.

```
html
<InputComponent
  inputType="text"
  :requiredField="true"
/>
```

Importante

Esta propiedad debe usarse junto con las propiedades `error` y `validation`.

error

Type: Boolean

Controla el **error de un input requerido** al momento de realizar un `submit`. Si el input esta vacío, agrega la clase `is-invalid`.

```
html
<InputComponent
  inputType="text"
  :requiredField="true"
  :error="formError"
/>
```

Importante

[Introducción](#)[Empezando](#)

Componentes

[Alert](#)[Badge](#)[Button](#)[Detail](#)[Dialog](#)[Dropdown](#)[Errors](#)[Filters](#)

Input

[Loading](#)[Pagination](#)[ProgressBar](#)[Select](#)[Tab](#)[Table](#)[TextareaResize](#)[Toast](#)

Utilidades

[useAction](#)[useFormat](#)[useInputMask](#)[useKeypress](#)[useTable](#)[useTheme](#)[useTooltip](#)[useValidator](#)

validation

Type: [Function](#)

Valida que el componente tenga información al momento de realizar un [submit](#). Si el input esta vacío, agrega la clase [is-invalid](#).

Para más información de las validaciones, revisar la sección de utilidades [useValidator](#)

Ingresá un valor

```
<script setup lang="ts">
    import { ref } from 'vue'
    import { utils } from 'uikit-3bit-vue'

    const { validateDefault } = utils.useValidator()

    const formError = ref(true)
</script>

<template>
    <InputComponent
        inputType="text"
        :maxLength="10"
        :requiredField="true"
        :error="formError"
        placeHolder="Ingresá un valor"
        :validation="validateDefault"
    />
</template>
```

Importante

Esta propiedad debe usarse junto con las propiedades [error](#) y [requiredfield](#).

keyPress

Type: [Function](#)

Manejador de los eventos del teclado que controla los **caracteres ingresados** en el [input](#).

Para más información, revisar la sección de utilidades [useKeypress](#)

```
<InputComponent
    inputType="text"
    placeHolder="Ingresá un valor"
```

html

Composables

[useFilters](#)[useLogos](#)

Introducción

Empezando

Componentes

Alert

Badge

Button

Detail

Dialog

Dropdown

Errors

Filters

Input

Loading

Pagination

ProgressBar

Select

Tab

Table

TextareaResize

Toast

Utilidades

useAction

useFormat

useInputMask

useKeypress

useTable

useTheme

useTooltip

useValidator

Composables

useFilters

useLogos

inputMask

Type: `Function`

Manejador de máscaras para los distintos tipos de datos ingresados en el `input`.

Para más información, revisar la sección de utilidades [useInputMask](#)

```
<InputComponent  
  inputType="text"  
  placeHolder="Ingresa un valor"  
  :inputMask="inputMaskRut"  
/>
```

html

submitted

Type: `Boolean`

Propiedad que ejecuta, al pasarle un boolean `true`, una función interna llamada `clean()` que reinicia internamente el componente `input`. Usada principalmente en los formularios para limpiar todos los campos a la vez.

```
<script setup lang="ts">  
  const submittedProps = ref(false)  
</script>  
  
<template>  
  <InputComponent  
    inputType="text"  
    placeHolder="Ingresa un valor"  
    :submitted="submittedProps"  
  />  
</template>
```

vue

Emits

emitValue

Función que notifica los cambios realizados en la información del componente `input`. Se debe usar un manejador en el componente padre que reciba esta información.

Introducción

Empezando

Componentes

Alert

Badge

Button

Detail

Dialog

Dropdown

Errors

Filters

Input

Loading

Pagination

ProgressBar

Select

Tab

Table

TextareaResize

Toast

Utilidades

useAction

useFormat

useInputMask

useKeyypress

useTable

useTheme

useTooltip

useValidator

Composables

useFilters

useLogos

output →

```
vue
<script setup lang="ts">
  import { ref, computed } from 'vue'

  const inputEmit = ref('')

  const handleEmitInput = (value) => {
    inputEmit.value = value
  }
</script>

<template>
  <InputComponent
    inputType="text"
    placeHolder="Ingresa un valor"
    @emitValue="handleEmitInput"
  />
</template>
```

emitPressEnter

Función que se ejecuta cada vez que se presiona la tecla `enter`, siempre y cuando el focus este en el componente `input`.

Haz clic aquí y presiona ENTER

ENTER → 0

```
vue
<script setup lang="ts">
  import { ref, computed } from 'vue'

  const inputEnterCount = ref(0)

  const handlePressEnter = () => {
    inputEnterCount.value = inputEnterCount.value + 1
  }
</script>

<template>
  <InputComponent
    inputType="text"
    placeHolder="Haz clic aquí y presiona ENTER"
    @emitPressEnter="handlePressEnter"
  />
</template>
```

Introducción

Empezando

Componentes

Alert

Badge

Button

Detail

Dialog

Dropdown

Errors

Filters

Input

Loading

Pagination

ProgressBar

Select

Tab

Table

TextareaResize

Toast

Utilidades

useAction

useFormat

useInputMask

useKeypress

useTable

useTheme

useTooltip

useValidator

Composables

useFilters

useLogos

input password

Para este ejemplo se necesitan las propiedades `floatRight` y `inputType` como mínimo. Se recomienda usar la librería [fontawesome](#)

Contraseña



```
vue
<script setup lang="ts">
  import { ref, computed } from 'vue'

  const showPassword = ref(false)
  const inputPassword = ref('')

  const controlTypeInputPassword = computed(() => {
    return showPassword.value ? 'text' : 'password'
  })

  const handleVisiblePass = () => {
    showPassword.value = !showPassword.value
  }

  const handlePasswordValue = (value) => {
    inputPassword.value = value
  }
</script>

<template>
  <InputComponent
    :inputType="controlTypeInputPassword"
    placeHolder="Contraseña"
    floatRight
    @emitValue="handlePasswordValue"
  >
    <template #float-right>
      <a
        @click="handleVisiblePass()"
        href="javascript:"
        data-eit-color="text-soft"
        data-eit-link
      >
        <font-awesome-icon
          v-if="!showPassword"
          icon="fa-regular fa-eye-slash"
        />
        <font-awesome-icon
          v-if="showPassword"
          icon="fa-regular fa-eye"
        />
      </a>
    </template>
  </InputComponent>
</template>
```

Introducción

Empezando

Componentes

Alert

Badge

Button

Detail

Dialog

Dropdown

Errors

Filters

Input

Loading

Pagination

ProgressBar

Select

Tab

Table

TextareaResize

Toast

Utilidades

useAction

useFormat

useInputMask

useKeypress

useTable

useTheme

useTooltip

useValidator

Composables

useFilters

useLogos

input datepicker

Para este ejemplo se necesita instalar la librería [Vue Datepicker](#)



Fecha

```
<script setup lang="ts">
  import { ref } from 'vue'
  import { utils } from 'uikit-3it-vue'

  const { formatDate } = utils.useFormat()

  const inputDate = ref(null)
  const inputDateRef = ref(null)

  const handleClearDate = () => {
    inputDateRef.value.clean()
  }
</script>

<template>
  <VueDatePicker
    auto-apply
    locale="es"
    :enable-time-picker="false"
    :month-change-on-scroll="false"
    :format="formatDate"
    v-model="inputDate"
  >
    <template #dp-input="{ value }">
      <InputComponent
        ref="inputDateRef"
        inputType="text"
        floatLeft
        :input="value"
      >
        <template #float-left>
          <font-awesome-icon
            icon="fa-regular fa-calendar"
          />
        </template>
      </InputComponent>
    </template>
    <template #clear-icon="{ clear }">
      <font-awesome-icon
        :icon="['fas', 'xmark']"
        data-eit-pe="3"
        @click="clear(), handleClearDate()"
      />
    </template>
  </VueDatePicker>
</template>
```

vue

Introducción

Empezando

Componentes

Alert

Badge

Button

Detail

Dialog

Dropdown

Errors

Filters

Input

Loading

Pagination

ProgressBar

Select

Tab

Table

TextareaResize

Toast

Utilidades

useAction

useFormat

useInputMask

useKeypress

useTable

useTheme

useTooltip

useValidator

Previous page

Filters

Next page

Loading

Composables

useFilters

useLogos