Crear un proyecto

* pnpm create vuetify

# tema general

src/App.vue

cambiando entre valores como “light,dark”

<template>

  <v-app theme="light">

    <v-main>

      <router-view />

    </v-main>

  </v-app>

</template>

## Nuevo tema

Src/plugins/ vuetify.ts

export default createVuetify({

  theme: {

    defaultTheme: 'dark',

    themes:{

      test:{

        dark:true,

        colors:{

          primary:'#c84a2f',

          secundary:'#7e301f',

          otrocolor:'#5c180a'

        }

      }

    }

  },

})

# Navegación al lado (v-navigation-drawer)

src/App.vue

<template>

  <v-app theme="dark">

    <v-navigation-drawer permanent>

      mi menu de navegacion

    </v-navigation-drawer>

    <v-main>

      <router-view />

    </v-main>

  </v-app>

</template>

## Con botón de menú desplegable lateral

<v-navigation-drawer v-model="isDrawerOpen">

      mi menu de navegacion

    </v-navigation-drawer>

    <v-app-bar

      flat

      class="border-b"

    >

    <v-app-bar-nav-icon

      @click="isDrawerOpen=!isDrawerOpen"

      >

    </v-app-bar-nav-icon>

    <v-app-bar-title>El titulo</v-app-bar-title>

      Mi barra de navegacion

    </v-app-bar>

<script lang="ts" setup>

import { ref } from 'vue';

  const isDrawerOpen=ref(false);

</script>

### Con lista en el menú desplegable

<v-navigation-drawer v-model="isDrawerOpen">

      <v-list>

        <v-list-subheader>Menu</v-list-subheader>

        <v-list-item prepend-icon="mdi-home">Home</v-list-item>

        <v-list-item prepend-icon="mdi-account">Usuario</v-list-item>

        <v-list-group value="Clientes">

          <template #activator="{props}">

            <!-- <v-btn v-bind="props">Abrir</v-btn> -->

            <v-list-item

              v-bind="props"

              prepend-icon="mdi-account-circle"

            >Clientes</v-list-item>

          </template>

          <v-list-item>Uno</v-list-item>

          <v-list-item>Dos</v-list-item>

        </v-list-group>

      </v-list>

    </v-navigation-drawer>

# Nombres de componentes por defecto:

Los nombres de los componentes que ya nos trae vue suelen ser <v-nombre-componente> pero tambien están los mismos de la forma <VNombreComponente>

# Barra de Navegación superior (v-app-bar)

src/App.vue

Tipo la que traen los celulares por defecto, o las típicas barras superiores de las paginas web

<template>

  <v-app theme="dark">

   <v-app-bar flat

    class="border-b"

    title="mi barra"

    >

      Mi barra de navegacion

    </v-app-bar>

    <v-main>

      <router-view />

    </v-main>

  </v-app>

</template>

## Sin sombra

<v-app-bar flat>

      Mi barra de navegacion

    </v-app-bar>

## Titulo separado (v-app-bar-title)

<v-app-bar

      flat

      class="border-b"

    >

    <v-app-bar-title>El titulo</v-app-bar-title>

      Mi barra de navegacion

    </v-app-bar>

# Atributos comunes a la mayoría de componentes

“flat” para quitar la sombra

“color=’#FFFF’”

# Clases css por defecto

## borde

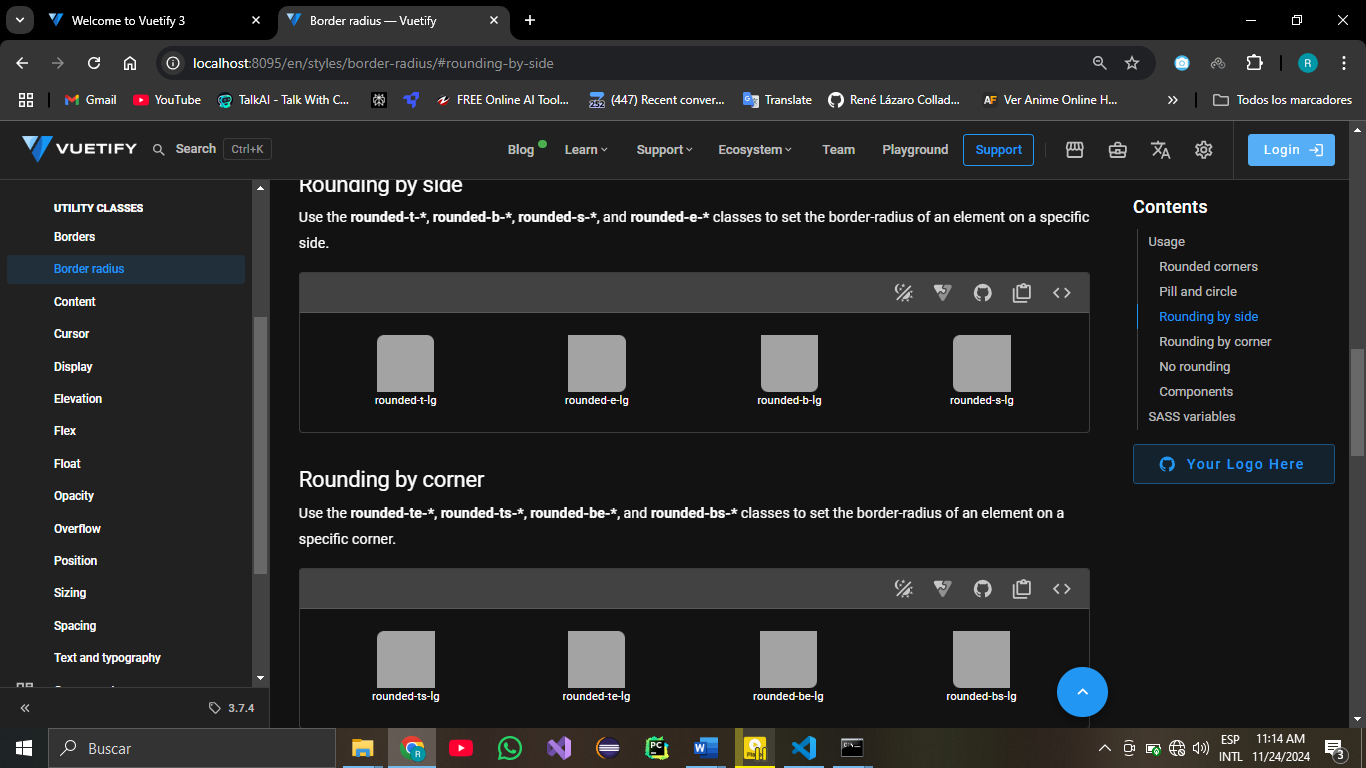
border-b -> de 1px

## radio

**rounded-lg -> radio de borde**

**rounded-pill -> ovalo**

**rounded-circle -> circulo**

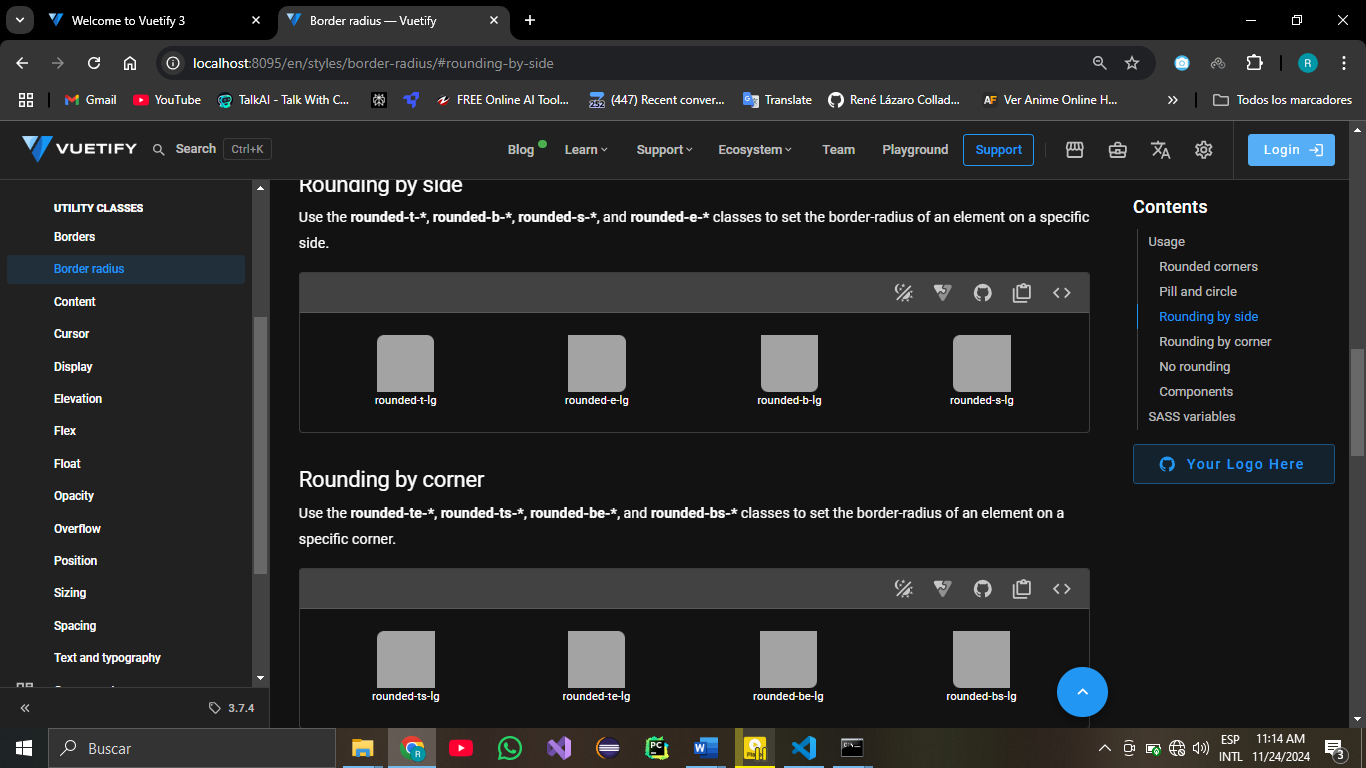


rounded-t-lg

rounded-e-lg

rounded-b-lg

rounded-s-lg



**rounded-ts-lg**

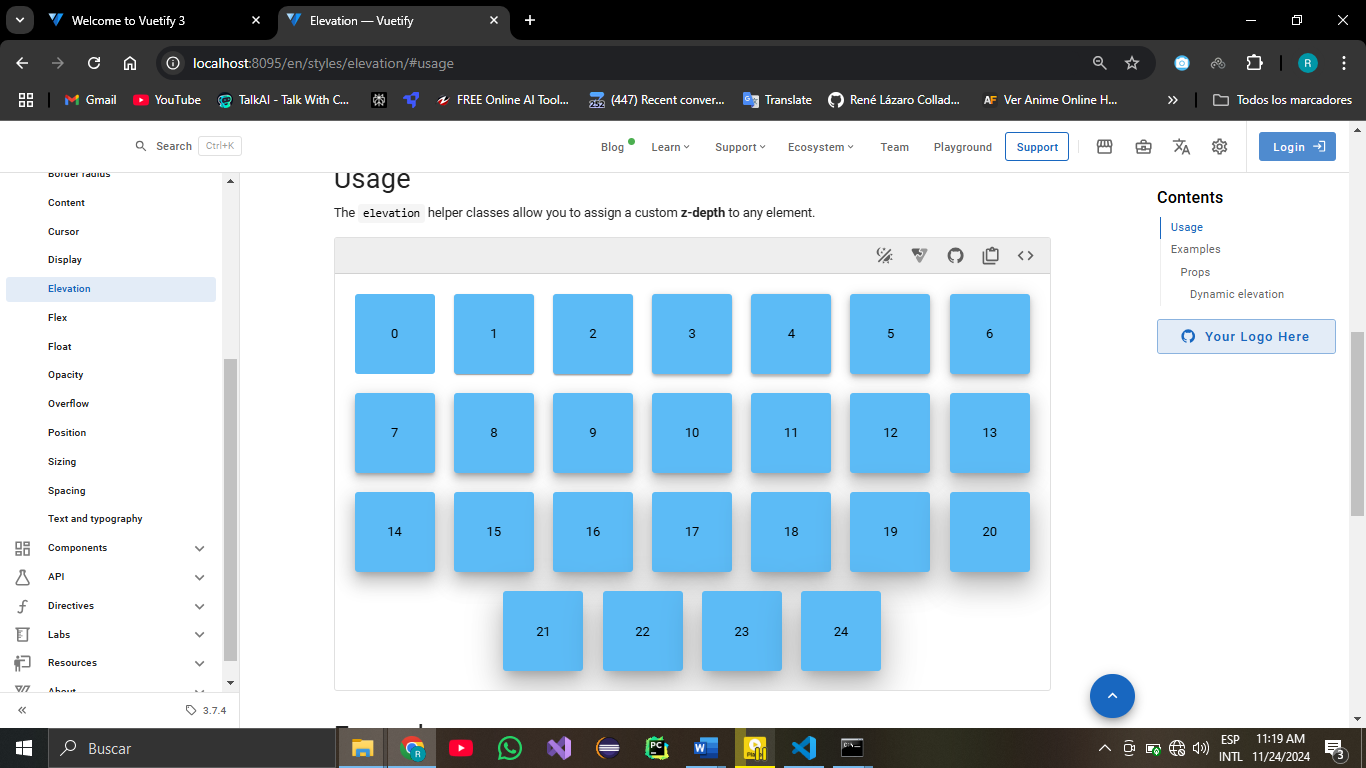
**rounded-te-lg**

**rounded-be-lg**

**rounded-bs-lg**

## sombras

**elevation-{n}**



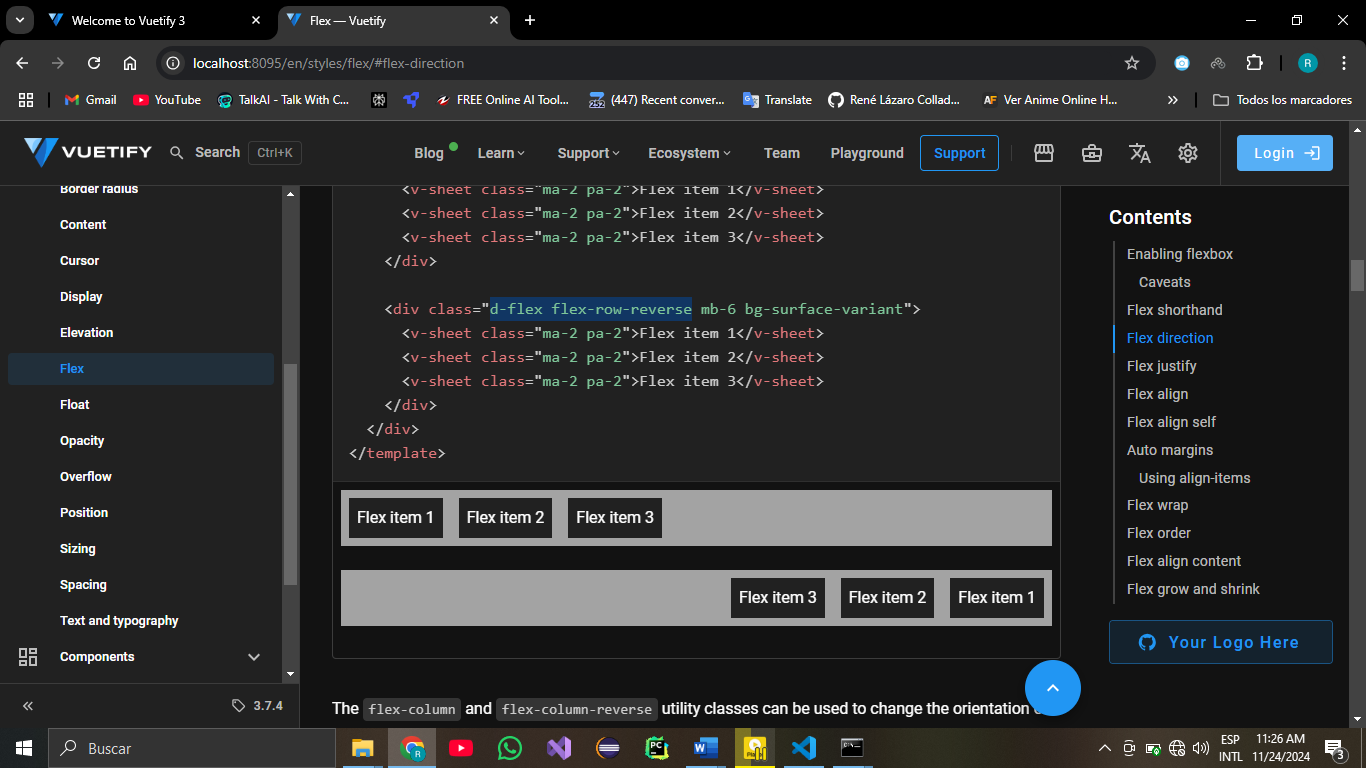
## Flex

http://localhost:8095/en/styles/flex/#flex-wrap

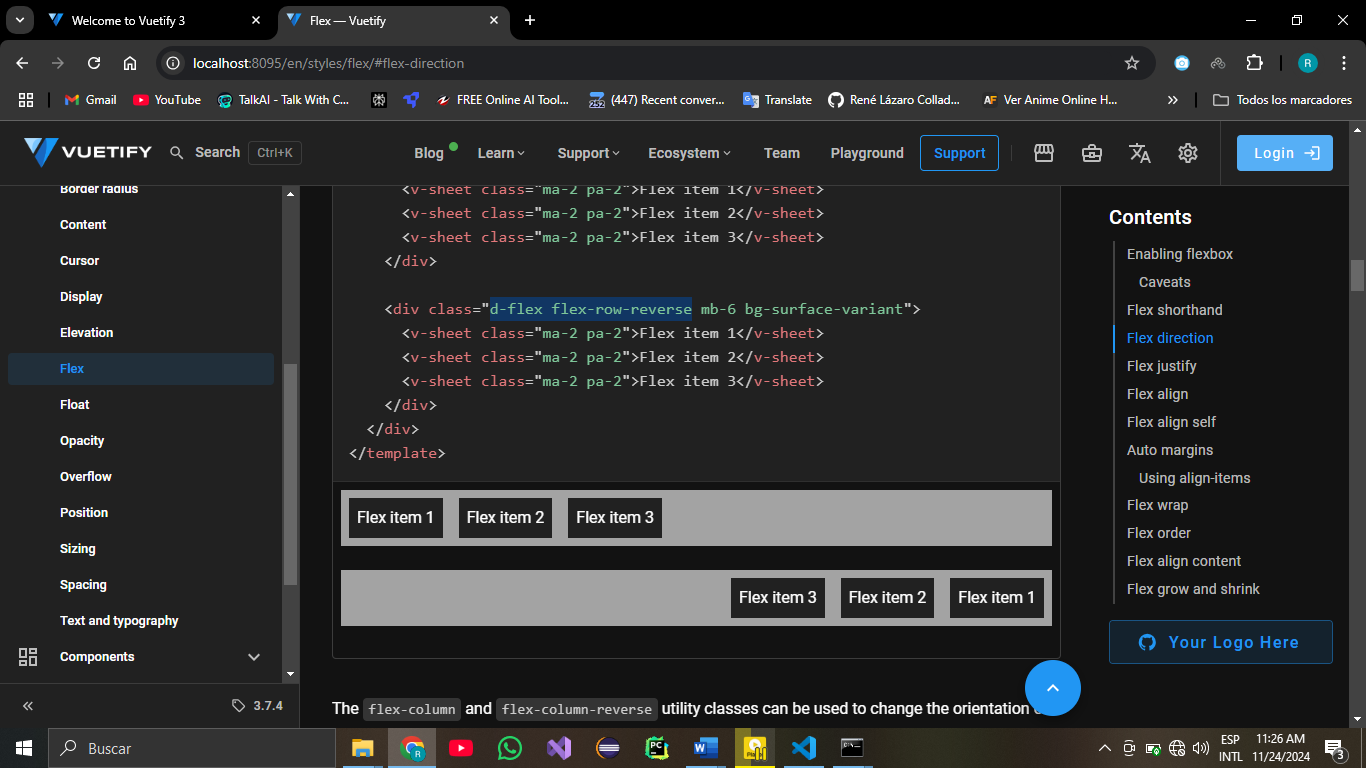
Se utiliza combinado d-flex con una clase de dirección

d-flex <+clase de dirección>

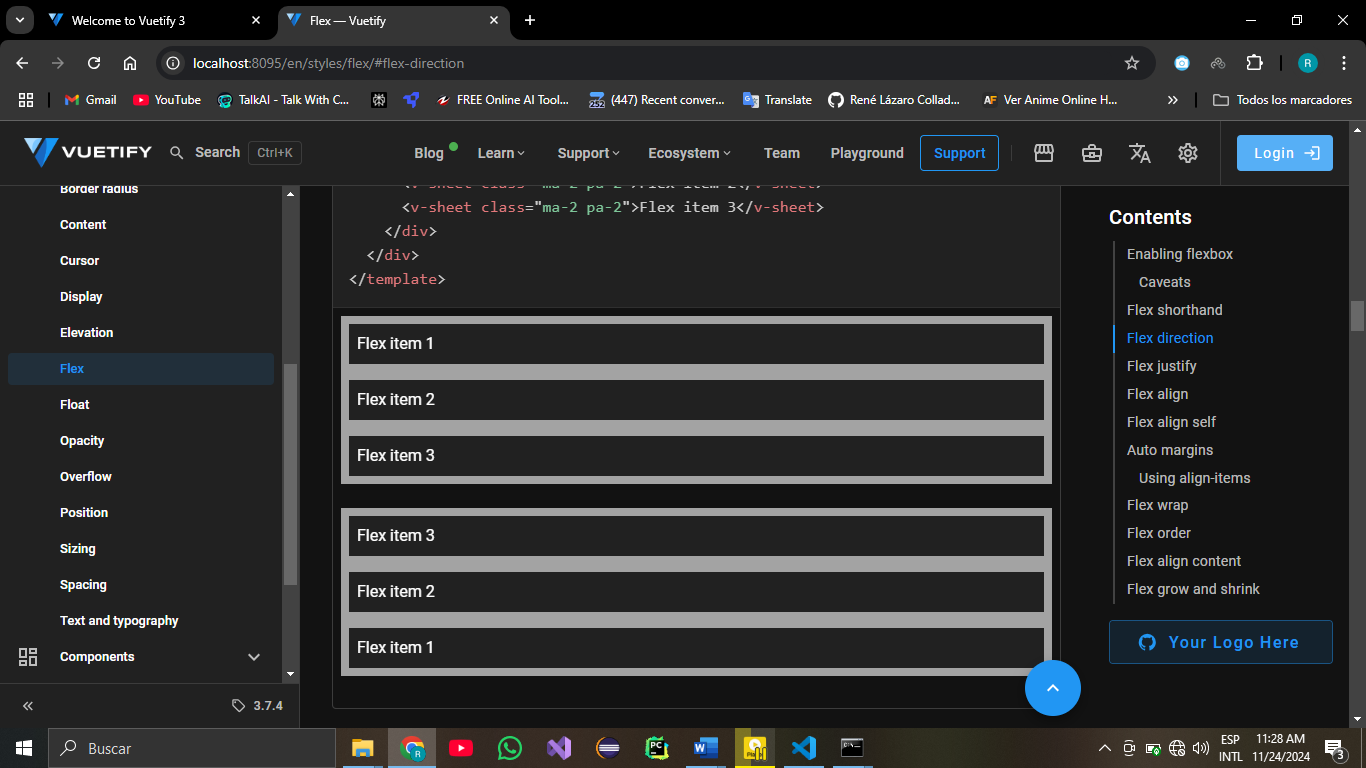
d-flex flex-row



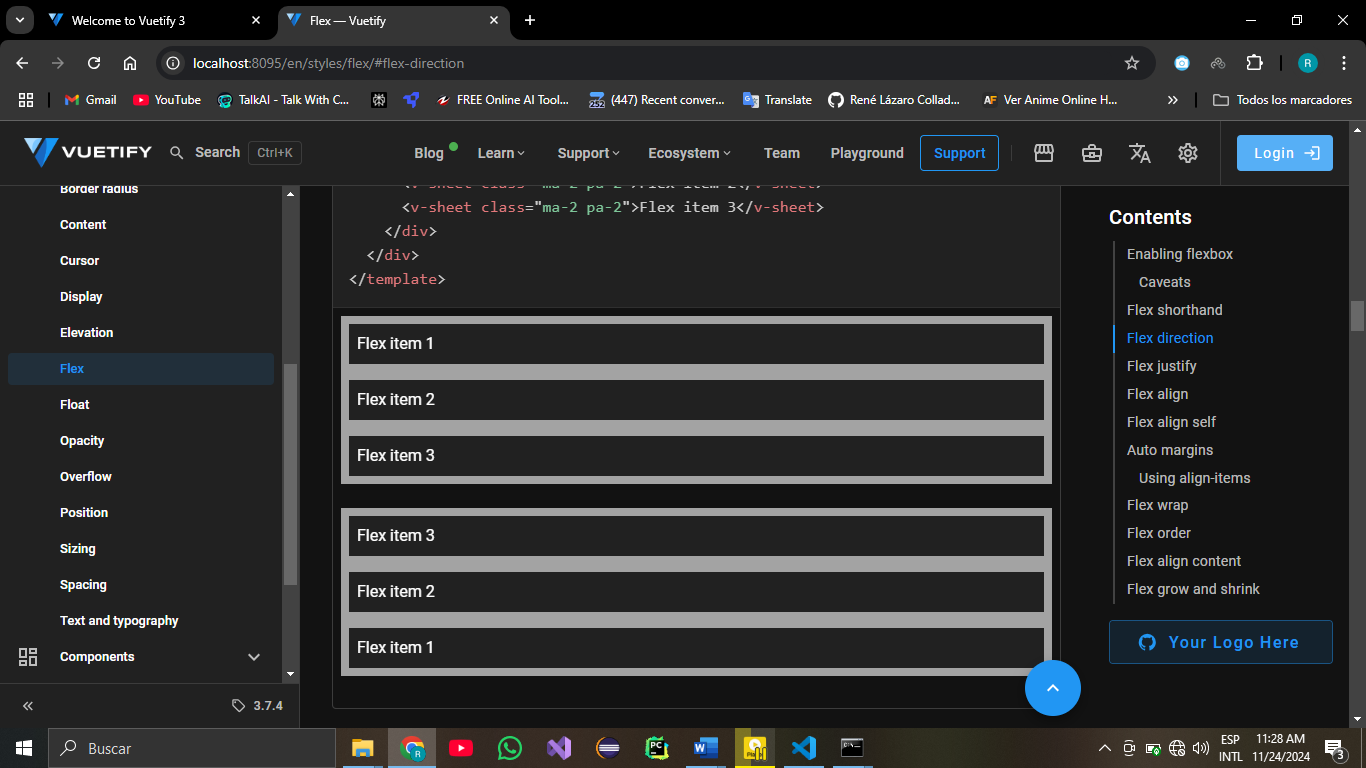
d-flex flex-row-reverse



d-flex flex-column (1-2-3)



d-flex flex-column-reverse (3-2-1)



justify-start

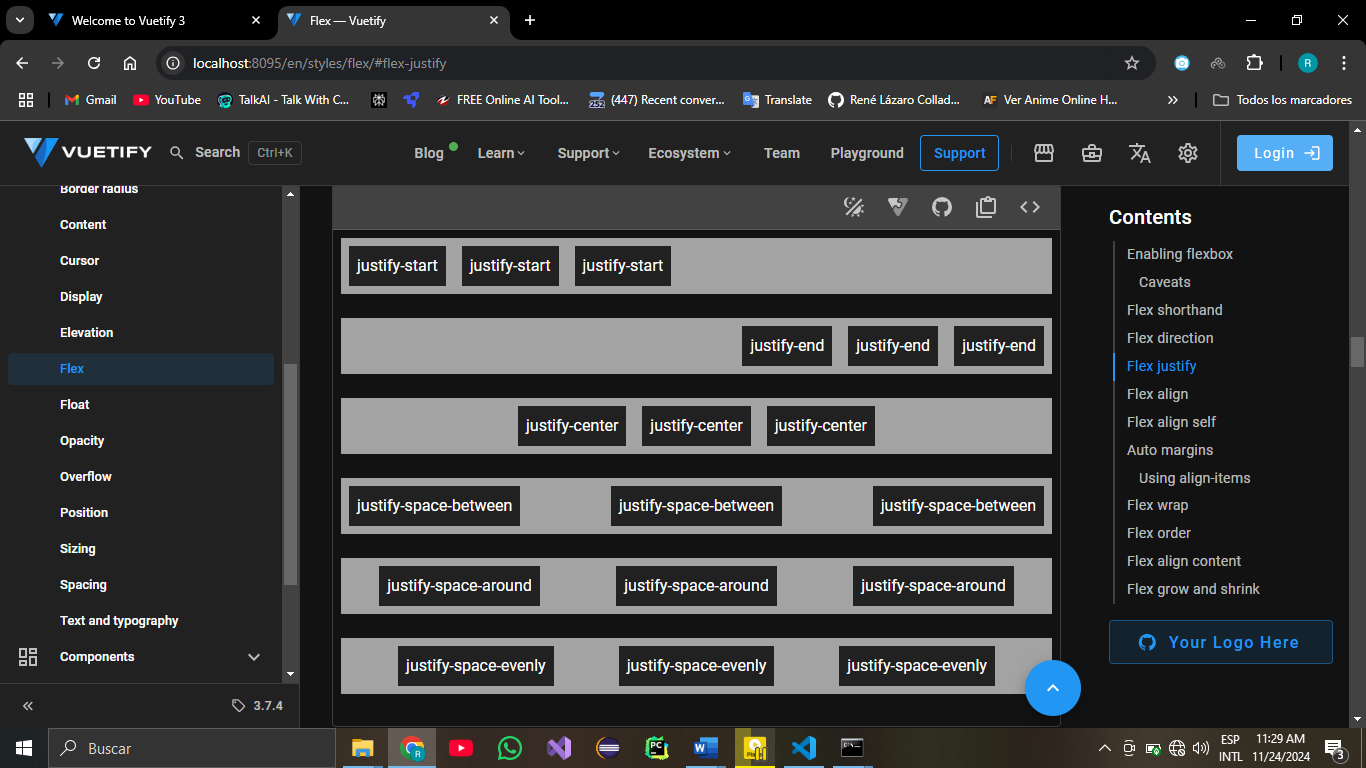
justify-end

justify-center

justify-space-between

justify-space-around

justify-space-evenly



## Sizing (h w)

|  |  |
| --- | --- |
| **h-auto** | height: auto |
| **h-screen** | height: 100vh |
| **h-0** | height: 0 |
| **h-25** | height: 25% |
| **h-50** | height: 50% |
| **h-75** | height: 75% |
| **h-100** | height: 100% |
| **fill-height** | height: 100% |
| **height-screen** | height: 100dvh |
| **w-auto** | width: auto |
| **w-0** | width: 0 |
| **w-33** | width: 33% |
| **w-25** | width: 25% |
| **w-50** | width: 50% |
| **w-66** | width: 66% |
| **w-75** | width: 75% |
| **w-100** | width: 100% |

## Pardding y margin

{property}{direction}-{size}

Ejemplo: pt-2

The **property** applies the type of spacing:

* m - applies margin
* p - applies padding
* g - applies gap

The **direction** designates the side the property applies to:

* t - applies the spacing for margin-top and padding-top
* b - applies the spacing for margin-bottom and padding-bottom
* l - applies the spacing for margin-left and padding-left
* r - applies the spacing for margin-right, padding-right, and row-gap
* s - applies the spacing for margin-left/padding-left (in LTR mode) and margin-right/padding-right (in RTL mode)
* e - applies the spacing for margin-right/padding-right (in LTR mode) and margin-left/padding-left (in RTL mode)
* x - applies the spacing for margin and padding \*-left and \*-right
* y - applies the spacing for margin and padding \*-top and \*-bottom
* a - applies the spacing for margin, padding and gap in all directions
* c - applies the spacing for column-gap

The **size** controls the increment of the property in 4px intervals:

* 0 - eliminates all margin, padding or gap by setting it to 0
* 1 - sets margin, padding or gap to 4px
* 2 - sets margin, padding or gap to 8px
* 3 - sets margin, padding or gap to 12px
* 4 - sets margin, padding or gap to 16px
* 5 - sets margin, padding or gap to 20px
* 6 - sets margin, padding or gap to 24px
* 7 - sets margin, padding or gap to 28px
* 8 - sets margin, padding or gap to 32px
* 9 - sets margin, padding or gap to 36px
* 10 - sets margin, padding or gap to 40px
* 11 - sets margin, padding or gap to 44px
* 12 - sets margin, padding or gap to 48px
* 13 - sets margin, padding or gap to 52px
* 14 - sets margin, padding or gap to 56px
* 15 - sets margin, padding or gap to 60px
* 16 - sets margin, padding or gap to 64px
* n1 - sets margin to -4px
* n2 - sets margin to -8px
* n3 - sets margin to -12px
* n4 - sets margin to -16px
* n5 - sets margin to -20px
* n6 - sets margin to -24px
* n7 - sets margin to -28px
* n8 - sets margin to -32px
* n9 - sets margin to -36px
* n10 - sets margin to -40px
* n11 - sets margin to -44px
* n12 - sets margin to -48px
* n13 - sets margin to -52px
* n14 - sets margin to -56px
* n15 - sets margin to -60px
* n16 - sets margin to -64px
* auto - sets the spacing to **auto**

## Text and typography

|  |  |
| --- | --- |
| **text-h1** | font-size: 6rem; font-weight: 300; line-height: 1; letter-spacing: -0.015625em |
| **text-h2** | font-size: 3.75rem; font-weight: 300; line-height: 1; letter-spacing: -0.0083333333em; |
| **text-h3** | font-size: 3rem; font-weight: 400; line-height: 1.05; letter-spacing: normal; |
| **text-h4** | font-size: 2.125rem; font-weight: 400; line-height: 1.175; letter-spacing: 0.0073529412em; |
| **text-h5** | font-size: 1.5rem; font-weight: 400; line-height: 1.333; line-height: normal; |
| **text-h6** | font-size: 1.25rem; font-weight: 500; line-height: 1.6; letter-spacing: 0.0125em; |
| **text-subtitle-1** | font-size: 1rem; font-weight: normal; line-height: 1.75; letter-spacing: 0.009375em; |
| **text-subtitle-2** | font-size: 0.875rem; font-weight: 500; line-height: 1.6; letter-spacing: 0.0071428571em; |
| **text-body-1** | font-size: 1rem; font-weight: 400; line-height: 1.5; letter-spacing: 0.03125em; |
| **text-body-2** | font-size: font-size: 0.875rem; font-weight: 400; line-height: 1.425; letter-spacing: 0.0178571429em; |
| **text-button** | font-size: 0.875rem; font-weight: 500; line-height: 2.6; letter-spacing: 0.0892857143em; text-transform: uppercase; |
| **text-caption** | font-size: 0.75rem; font-weight: 400; line-height: 1.667; letter-spacing: 0.0333333333em; |
| **text-overline** | font-size: 0.75rem; font-weight: 500; line-height: 2.667; letter-spacing: 0.1666666667em; text-transform: uppercase; |
| **text-high-emphasis** | color: rgba(var(–v-theme-on-background), var(–v-high-emphasis-opacity)); |
| **text-medium-emphasis** | color: rgba(var(–v-theme-on-background), var(–v-medium-emphasis-opacity)); |
| **text-disabled** | color: rgba(var(–v-theme-on-background), var(–v-disabled-opacity)); |
| **text-uppercase** | text-transform: uppercase; |
| **text-lowercase** | text-transform: lowercase; |
| **text-capitalize** | text-transform: capitalize; |
| **text-start** | text-align: start; |
| **text-center** | text-align: center; |
| **text-end** | text-align: end; |
| **text-justify** | text-align: justify; |
| **text-left** | text-align: left; |
| **text-right** | text-align: right; |
| **text-truncate** | overflow: hidden; text-overflow: ellipsis; white-space: nowrap; |
| **text-no-wrap** | white-space: nowrap; |
| **text-break** | overflow-wrap: break-word; |
| **text-decoration-none** | text-decoration: none; |
| **text-decoration-overline** | text-decoration: overline; |
| **text-decoration-underline** | text-decoration: underline; |
| **text-decoration-line-through** | text-decoration: line-through; |
| **font-weight-black** | font-weight: 900; |
| **font-weight-bold** | font-weight: 700; |
| **font-weight-medium** | font-weight: 500; |
| **font-weight-regular** | font-weight: 400; |
| **font-weight-light** | font-weight: 300; |
| **font-weight-thin** | font-weight: 100; |
| **font-italic** | font-style: italic; |
| **text-{breakpoint}-h1** | Set the font size for the specified breakpoint. |
| **text-{breakpoint}-h2** | Set the font size for the specified breakpoint. |
| **text-{breakpoint}-h3** | Set the font size for the specified breakpoint. |
| **text-{breakpoint}-h4** | Set the font size for the specified breakpoint. |
| **text-{breakpoint}-h5** | Set the font size for the specified breakpoint. |
| **text-{breakpoint}-h6** | Set the font size for the specified breakpoint. |
| **text-{breakpoint}-subtitle-1** | Set the font size for the specified breakpoint. |
| **text-{breakpoint}-subtitle-2** | Set the font size for the specified breakpoint. |
| **text-{breakpoint}-body-1** | Set the font size for the specified breakpoint. |
| **text-{breakpoint}-body-2** | Set the font size for the specified breakpoint. |
| **text-{breakpoint}-button** | Set the font size for the specified breakpoint. |
| **text-{breakpoint}-caption** | Set the font size for the specified breakpoint. |
| **text-{breakpoint}-overline** | Set the font size for the specified breakpoint. |

## Ocultar

|  |  |
| --- | --- |
| Hidden on all | .d-none |
| Hidden only on xs | .d-none .d-sm-flex |
| Hidden only on sm | .d-sm-none .d-md-flex |
| Hidden only on md | .d-md-none .d-lg-flex |
| Hidden only on lg | .d-lg-none .d-xl-flex |
| Hidden only on xl | .d-xl-none .d-xxl-flex |
| Hidden only on xxl | .d-xxl-none |
| Visible on all | .d-flex |
| Visible only on xs | .d-flex .d-sm-none |
| Visible only on sm | .d-none .d-sm-flex .d-md-none |
| Visible only on md | .d-none .d-md-flex .d-lg-none |
| Visible only on lg | .d-none .d-lg-flex .d-xl-none |
| Visible only on xl | .d-none .d-xl-flex .d-xxl-none |
| Visible only on xxl | .d-none .d-xxl-flex |

De la forma anterior están cultos por defecto, pero para que sean visibles por defecto y bajo ciertas condiciones se oculten es con

hidden-{breakpoint}-{condition?}

ejemplo:

hidden-md

### only

solo se oculta en este breakpoint

hidden-md-only

## color

{bg|text}-{color}-{lighten|darken|}-{#}

bg-red

bg-blue-2

# Breakpoints

|  |  |  |  |
| --- | --- | --- | --- |
| Extra small | **xs** | Small to large phone | < 600px |
| Small | **sm** | Small to medium tablet | 600px > < 960px |
| Medium | **md** | Large tablet to laptop | 960px > < 1280px |
| Large | **lg** | Laptop to desktop | 1280px > < 1920px |
| Extra large | **xl** | 1080p to 1440p desktop | 1920px > < 2560px |
| Extra extra large | **xxl** | 4k and ultra-wide | > 2560px |

# Grid

Es una combinación de container, row y col

<v-container>

    <v-row>

      <v-col

      justify="start"

      aling="end"

       class="red">Celda 1-1</v-col>

      <v-col class="red">Celda 1-2</v-col>

      <v-col class="red">Celda 1-3</v-col>

    </v-row>

    <v-row no-gutters>

      <v-col class="blue">Celda 2-1</v-col>

      <v-col class="blue">Celda 2-2</v-col>

    </v-row>

    <v-row>

      <v-spacer />

      <v-col class="brown">Celda 3-1</v-col>

      <v-spacer />

    </v-row>

    <v-row>

        <v-col>

            <v-expansion-panels>

                <v-expansion-panel

                    title="Titulo"

                    text="El texto"

                ></v-expansion-panel>

            </v-expansion-panels>

        </v-col>

    </v-row>

  </v-container>

Las columnas tiene una amplitude total de 12 y se puede manipular con el atributo cols="1"

<v-container>

        <v-row>

            <v-col cols="1">A</v-col>

            <v-col cols="10">B</v-col>

            <v-col cols="1">C</v-col>

        </v-row>

    </v-container>

Esta amplitud se puede manipular según el brackpoint

<v-col cols="10" md="4" lg="2">B</v-col>

# Card

| Component | Description |
| --- | --- |
| [v-card](http://localhost:8095/en/api/v-card/) | Primary Component |
| [v-card-item](http://localhost:8095/en/api/v-card-item/) | Sub-component used to wrap the Card’s v-card-title and v-card-subtitle components. |
| [v-card-title](http://localhost:8095/en/api/v-card-title/) | Sub-component used to display the Card’s title. Wraps the #title slot |
| [v-card-subtitle](http://localhost:8095/en/api/v-card-subtitle/) | Sub-component used to display the Card’s subtitle. Wraps the #subtitle slot. |
| [v-card-text](http://localhost:8095/en/api/v-card-text/) | Sub-component used to display the Card’s text. Wraps the #text slot. |
| [v-card-actions](http://localhost:8095/en/api/v-card-actions/) | Sub-component that modifies the default styling of [v-btn](http://localhost:8095/en/components/buttons/). Wraps the #actions slot |

<template>

    <v-card class="mx-auto" max-width="400">

      <v-img

        color="surface-variant"

        height="200"

        src="https://cdn.vuetifyjs.com/docs/images/cards/purple-flowers.jpg"

        cover

      >

        <v-toolbar color="transparent">

          <template v-slot:prepend>

            <v-btn icon="$menu"></v-btn>

          </template>

          <v-toolbar-title class="text-h6" text="Messages"></v-toolbar-title>

          <template v-slot:append>

            <v-btn icon="mdi-dots-vertical"></v-btn>

          </template>

        </v-toolbar>

      </v-img>

      <v-card-text>

        <div class="font-weight-bold ms-1 mb-2">Today</div>

        <v-timeline align="start" density="compact">

          <v-timeline-item

            v-for="message in messages"

            :key="message.time"

            :dot-color="message.color"

            size="x-small"

          >

            <div class="mb-4">

              <div class="font-weight-normal">

                <strong>{{ message.from }}</strong> @{{ message.time }}

              </div>

              <div>{{ message.message }}</div>

            </div>

          </v-timeline-item>

        </v-timeline>

      </v-card-text>

    </v-card>

  </template>

  <script>

    export default {

      data: () => ({

        messages: [

          {

            from: 'You',

            message: `Sure, I'll see you later.`,

            time: '10:42am',

            color: 'deep-purple-lighten-1',

          },

          {

            from: 'John Doe',

            message: 'Yeah, sure. Does 1:00pm work?',

            time: '10:37am',

            color: 'green',

          },

          {

            from: 'You',

            message: 'Did you still want to grab lunch today?',

            time: '9:47am',

            color: 'deep-purple-lighten-1',

          },

        ],

      }),

    }

  </script>

# Boton

<v-btn prepend-icon="$vuetify">

                    Button

                    </v-btn>

## Con click

<v-btn

                  color="blue-darken-1"

                  variant="text"

                  @click="save"

                >

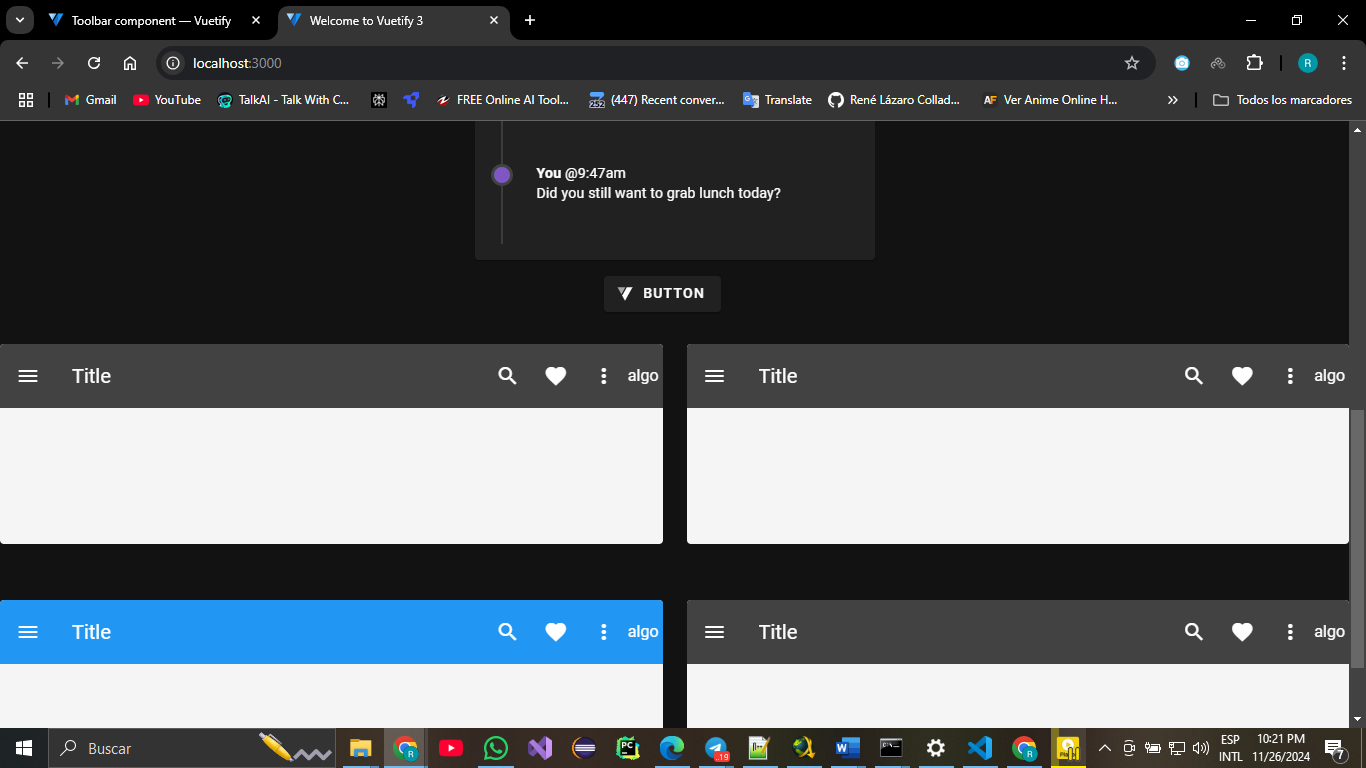
                  Save

                </v-btn>

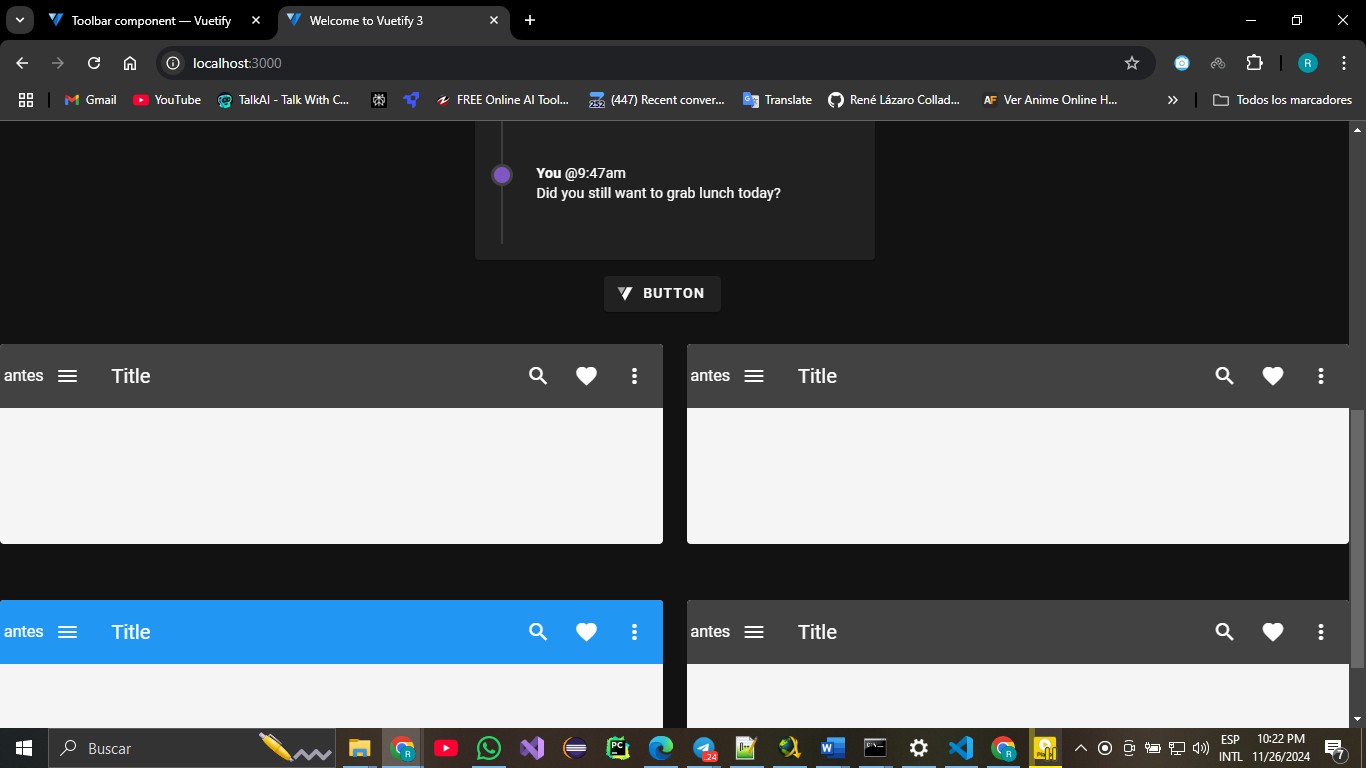
El método sabe es de tipo ()=>{}

# Barra superior

Con algo posterior a los iconos se agrega <template #append> o anterior



<template #prepend>



<v-toolbar

            :color="bar.class"

            :dark="bar.dark"

          >

            <v-app-bar-nav-icon></v-app-bar-nav-icon>

            <v-toolbar-title>Title</v-toolbar-title>

            <v-spacer></v-spacer>

            <v-btn icon>

              <v-icon>mdi-magnify</v-icon>

            </v-btn>

            <v-btn icon>

              <v-icon>mdi-heart</v-icon>

            </v-btn>

            <v-btn icon>

              <v-icon>mdi-dots-vertical</v-icon>

            </v-btn>

            <template #append>

                algo

            </template>

          </v-toolbar>

# Iconos

Depende de la librería a utilizar

Por ejemplo los de vuetify

 src/plugins/vuetify.js

import { createVuetify } from 'vuetify'

import { aliases, mdi } from 'vuetify/iconsets/mdi'

export default createVuetify({

icons: {

defaultSet: 'mdi',

aliases,

sets: {

mdi,

},

},

})

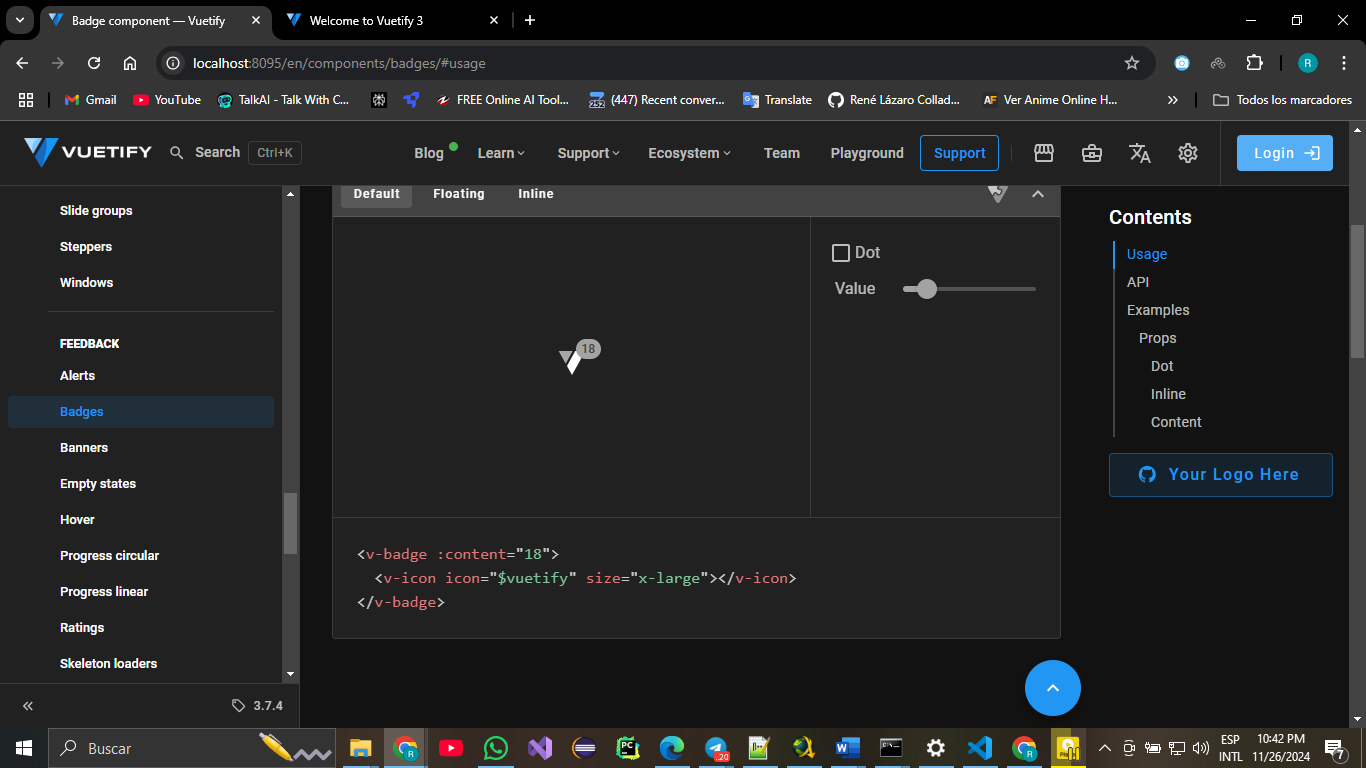
<template>

<v-icon icon="mdi-home" />

</template>

# Badge

<v-badge :content="18"> <v-icon icon="$vuetify" size="x-large"></v-icon> </v-badge>



# v-list

<v-list>

        <v-list-subheader>Menu</v-list-subheader>

        <v-list-item prepend-icon="mdi-home">Home</v-list-item>

        <v-list-item prepend-icon="mdi-account">Usuario</v-list-item>

        <v-list-group value="Clientes">

          <template #activator="{props}">

            <!-- <v-btn v-bind="props">Abrir</v-btn> -->

            <v-list-item

              v-bind="props"

              prepend-icon="mdi-account-circle"

            >Clientes</v-list-item>

          </template>

          <v-list-item>Uno</v-list-item>

          <v-list-item>Dos</v-list-item>

        </v-list-group>

      </v-list>

# Tablas

<template>

    <v-table>

      <thead>

        <tr>

          <th class="text-left">

            Name

          </th>

          <th class="text-left">

            Calories

          </th>

        </tr>

      </thead>

      <tbody>

        <tr

          v-for="item in desserts"

          :key="item.name"

        >

          <td>{{ item.name }}</td>

          <td>{{ item.calories }}</td>

        </tr>

      </tbody>

    </v-table>

  </template>

  <script lang="ts">

    export default {

      data () {

        return {

          desserts: [

            {

              name: 'Frozen Yogurt',

              calories: 159,

            },

            {

              name: 'Ice cream sandwich',

              calories: 237,

            },

            {

              name: 'Eclair',

              calories: 262,

            },

            {

              name: 'Cupcake',

              calories: 305,

            },

            {

              name: 'Gingerbread',

              calories: 356,

            },

            {

              name: 'Jelly bean',

              calories: 375,

            },

            {

              name: 'Lollipop',

              calories: 392,

            },

            {

              name: 'Honeycomb',

              calories: 408,

            },

            {

              name: 'Donut',

              calories: 452,

            },

            {

              name: 'KitKat',

              calories: 518,

            },

          ],

        }

      },

    }

  </script>

# Modales

<v-dialog width="300px">

                    <template #activator="{props}">

                        <v-btn

                            v-bind="props"

                            variant="tonal"

                            size="small">

                            Agregar

                        </v-btn>

                    </template>

                    <v-card>

                        <v-card-text>El texto</v-card-text>

                    </v-card>

                </v-dialog>

## Con ref

<v-btn @click="isDialogOpen = true" variant="tonal" size="small">

          Editar

        </v-btn>

        <v-dialog width="600px" v-model="isDialogOpen">

          <v-card>

            <v-card-text>El texto2</v-card-text>

          </v-card>

        </v-dialog>

import { ref } from "vue";

const isDialogOpen = ref(false);

# text field

<v-text-field  label="Nombre:" variant="outlined" />

# Select

<v-select

                    label="Pais"

                    variant="outlined"

                    :items="['Cuba','EEUU']"

                    />

# Formulario en dialog (sin almacenar valores)

<v-dialog v-model="isDialogOpen">

            <v-card class="border m-4">

                    <v-card-title>El Formulario</v-card-title>

                    <v-card-text>

                        <v-row>

                            <v-col>

                                <v-text-field  label="Nombre:" variant="outlined" />

                            </v-col>

                        </v-row>

                        <v-row>

                            <v-col>

                                <v-select

                                    label="Pais"

                                    variant="outlined"

                                    :items="['Cuba','EEUU']"

                                    />

                            </v-col>

                        </v-row>

                    </v-card-text>

                    <v-card-actions>

                        <v-btn variant="tonal" color="succes">Agregar</v-btn>

                    </v-card-actions>

                </v-card>

        </v-dialog>

<script setup lang="ts">

    import { ref } from "vue";

    const isDialogOpen = ref(false);

</script>

# Validaciones

Lo que se necesita es declarar métodos (reglas) que retornen verdadero si es valido o un mensaje si es invalido, luego tener por cada campo una variable que sea una lista de estas funciones (reglas), que luego un input va a recibir en el atributo :rules

<v-text-field

                                label="Correo:"

                                variant="outlined"

                                :rules="correoRules"

                                />

const correoRules=[

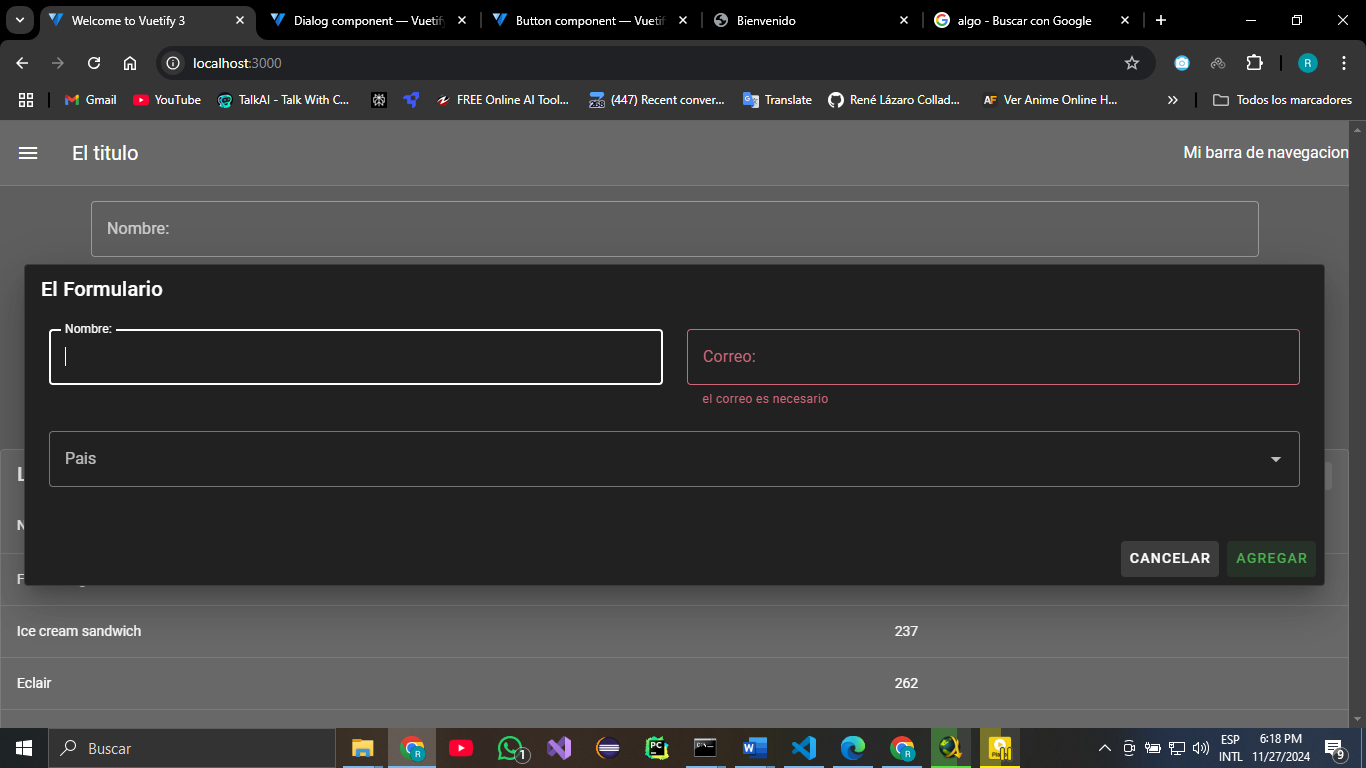
        value=>{

            if (value) return true;

            return "el correo es necesario"

        }

    ]



# Ripple

Se le puede agregar a cualquier etiqueta simplemente agregando el atributo “v-ripple”

<v-card-title v-ripple>La Tabla</v-card-title>

Tambien se puede desactivar de un componente que tal vez la tenga por defecto poniendo a false el atriburo “:ripple=’false’”

# Configuraciones globales

Src/plugins/ vuetify.ts

En el ‘createVuetify’ se puden definir propiedades que van a ser aplicadas a todos los componentes

import { createApp } from 'vue'

import { createVuetify } from 'vuetify'

export default createVuetify({

defaults: {

global: {

ripple: false,

},

VSheet: {

elevation: 4,

},

},

})

# Datatable

## Sencillo

<v-data-table :items="items"></v-data-table>

<script setup> const items = [ { name: 'African Elephant', species: 'Loxodonta africana', diet: 'Herbivore', habitat: 'Savanna, Forests', }, // ... more items ] </script>

## Tabla automatica

<template> <v-data-table-server v-model:items-per-page="itemsPerPage" :headers="headers" :items="serverItems" :items-length="totalItems" :loading="loading" item-value="name" @update:options="loadItems" ></v-data-table-server> </template>

<script> const desserts = [ { name: 'Frozen Yogurt', calories: 159, fat: 6.0, carbs: 24, protein: 4.0, iron: '1', },…]

const FakeAPI = { async fetch ({ page, itemsPerPage, sortBy }) {…}

export default {

data: () => ({ itemsPerPage: 5, headers: [ { title: 'Dessert (100g serving)', align: 'start', sortable: false, key: 'name', }, { title: 'Calories', key: 'calories', align: 'end' }, { title: 'Fat (g)', key: 'fat', align: 'end' }, { title: 'Carbs (g)', key: 'carbs', align: 'end' }, { title: 'Protein (g)', key: 'protein', align: 'end' }, { title: 'Iron (%)', key: 'iron', align: 'end' }, ],

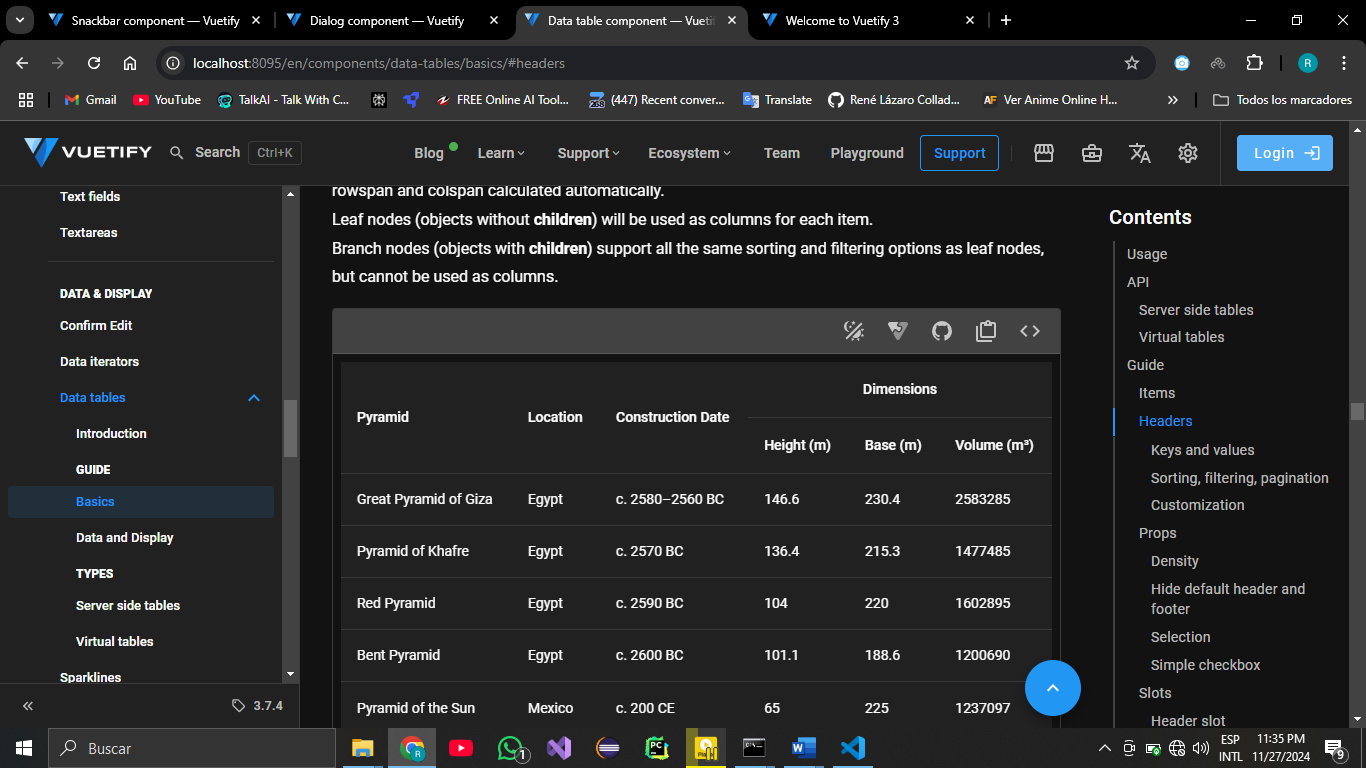
serverItems: [], loading: true, totalItems: 0, }),

methods: {

loadItems ({ page, itemsPerPage, sortBy }) { this.loading = true FakeAPI.fetch({ page, itemsPerPage, sortBy }).then(({ items, total }) => { this.serverItems = items this.totalItems = total this.loading = false }) }, }, }

</script>

## Columnas Complejas



<script> export default {

data: () => ({ headers: [ { title: 'Pyramid', value: 'name' }, { title: 'Location', value: 'location' }, { title: 'Construction Date', value: 'constructionDate' }, { title: 'Dimensions', align: 'center', children: [ { title: 'Height (m)', value: 'height' }, { title: 'Base (m)', value: 'base' }, { title: 'Volume (m³)', value: 'volume' }, ], }, ],

items: [ { name: 'Great Pyramid of Giza', location: 'Egypt', height: '146.6', base: '230.4', volume: '2583285', constructionDate: 'c. 2580–2560 BC', }, { name: 'Pyramid of Khafre', location: 'Egypt', height: '136.4', base: '215.3', volume: '1477485', constructionDate: 'c. 2570 BC', }, { name: 'Red Pyramid', location: 'Egypt', height: '104', base: '220', volume: '1602895', constructionDate: 'c. 2590 BC', }, { name: 'Bent Pyramid', location: 'Egypt', height: '101.1', base: '188.6', volume: '1200690', constructionDate: 'c. 2600 BC', }, { name: 'Pyramid of the Sun', location: 'Mexico', height: '65', base: '225', volume: '1237097', constructionDate: 'c. 200 CE', }, ], }), } </script>

## Key vs Value

Primero se lee el value (si existe se ignora el key, o para ser mas expecifico se agrega una columna al dato [item] con el nombre del key y con el valor del value ), el cual puede ser una llave del item o una funcion que reciba un item y retorne un string como valor correspondiente

items = [

{

id: 1,

name: {

first: 'John',

last: 'Doe',

},

}

]

headers = [

{ title: 'First Name', value: 'name.first' },

{ title: 'Last Name', key: 'name.last' },

{

title: 'Full Name',

key: 'fullName',

value: item => `${item.name.first} ${item.name.last}`,

},

]

## Seleccionable

<template> <v-data-table v-model="selected" :items="items" item-value="name" show-select ></v-data-table> </template>

## Columna Personalizada

Modificar la forma en que se ve una columna especifica, osea modificar el componente con el que se muestra, que por defecto es para strings

<template> <v-data-table :items="consoles"> <template v-slot:item.exclusive="{ item }"> <v-checkbox-btn v-model="item.exclusive" readonly ></v-checkbox-btn> </template> </v-data-table> </template>

### Utilizar el valor de la columna actual

<template> <v-data-table :headers="headers" :items="vegetables" > <template v-slot:item.calories="{ value }"> <v-chip :color="getColor(value)"> {{ value }} </v-chip> </template> </v-data-table> </template>

### Header Personalizado

<template> <v-data-table :items="items"> <template v-slot:header.id="{ column }"> {{ column.title.toUpperCase() }} </template> </v-data-table> </template>

### Nuevas Columnas

Se puede agregar una columna nueva, osea de un dato que no existe en la tabla, como la columna para las acciones o otra que reflege el resultado de la unión de dos datos, simplemente poniendo seguido del v-slot: un nombre que no existe entre las columnas originales de los datos

<template v-slot:nuevonombre>

### Columna de acciones

<template v-slot:item.actions="{ item }">

        <v-icon

          class="me-2"

          size="small"

          @click="editItem(item)"

        >

          mdi-pencil

        </v-icon>

        <v-icon

          size="small"

          @click="deleteItem(item)"

        >

          mdi-delete

        </v-icon>

      </template>

## Loading

Teniendo una variable booleana reactiva “loading” que indique cuando los datos se estan cargando tal vez desde un feth

<v-data-table :items="items" :loading="loading"> <template v-slot:loading> <v-skeleton-loader type="table-row@10"></v-skeleton-loader> </template> </v-data-table>

## sort-by

### orden por defecto de la tabla

<v-data-table

      :headers="headers"

      :items="desserts"

      :sort-by="[{ key: 'calories', order: 'asc' }]"

    >