

07/01/2025

MENÚ DE EVENTOS DESPLIEGABLE

**Hernández Trejo Wendy Belén
Ramírez López Karmi Yajseel**

Grupo:5801

**Bases de datos para
dispositivos móviles**

Base de datos: Script y modelo E-R

Esta base de datos se hizo con el propósito de crear un menú con submenús, donde cada elemento tiene eventos y cada evento tiene actividades:

1. **Tabla de elementos:** Cada fila en la tabla `elementos` representa un elemento principal del menú, con un nombre único. Esto permite organizar los diferentes apartados del menú principal.
2. **Tabla de eventos:** Cada evento está asociado a un elemento del menú, a través de la columna `elemento_id`, que actúa como llave foránea referenciando la tabla `elementos`. Esto te permite tener submenús (eventos) para cada elemento del menú principal.
3. **Tabla de actividades:** Cada actividad está asociada a un evento, mediante la columna `evento_id`, que es la llave foránea que hace referencia a la tabla `eventos`. Esto te permite tener submenús adicionales (actividades) dentro de cada evento.

Las llaves foráneas con la opción `ON DELETE CASCADE` garantizan que, si se elimina un elemento o evento, automáticamente se eliminarán los eventos o actividades asociados.

Script

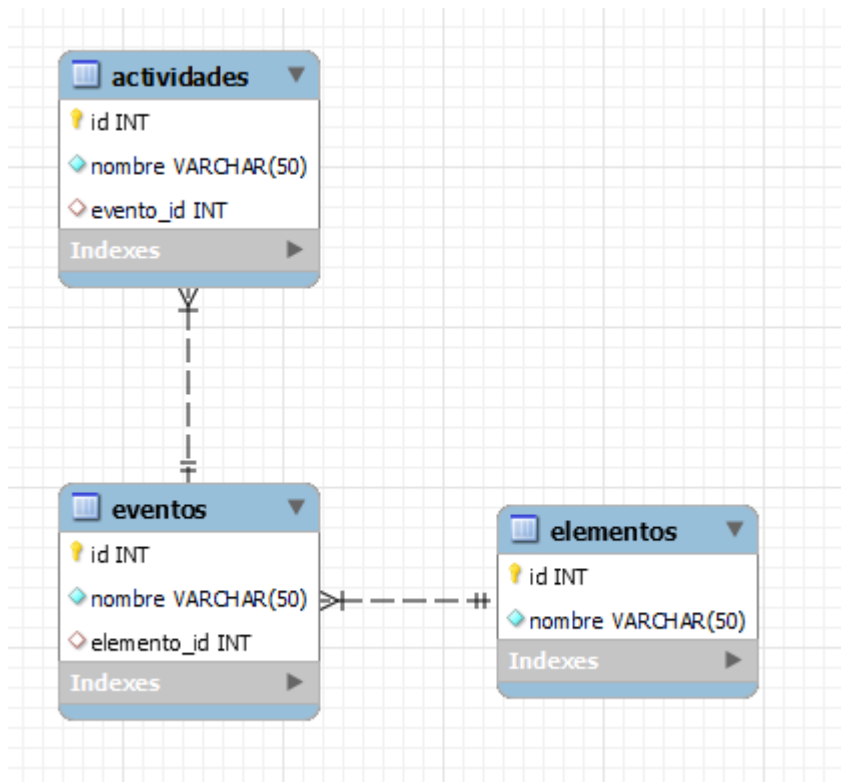
```
1 • CREATE DATABASE IF NOT EXISTS menu_eventos_db;
2 • USE menu_eventos_db;
3 • CREATE TABLE elementos (
4     id INT AUTO_INCREMENT PRIMARY KEY,
5     nombre VARCHAR(50) NOT NULL UNIQUE
6 );
7 • CREATE TABLE eventos (
8     id INT AUTO_INCREMENT PRIMARY KEY,
9     nombre VARCHAR(50) NOT NULL,
10    elemento_id INT,
11    FOREIGN KEY (elemento_id) REFERENCES elementos(id) ON DELETE CASCADE
12 );
13 • CREATE TABLE actividades (
14     id INT AUTO_INCREMENT PRIMARY KEY,
15     nombre VARCHAR(50) NOT NULL,
16     evento_id INT,
17     FOREIGN KEY (evento_id) REFERENCES eventos(id) ON DELETE CASCADE
18 );
```

Inserciones

```
INSERT INTO elementos (nombre) VALUES ('elemento n'), ('elemento m'),('eventos');
INSERT INTO eventos (nombre, elemento_id) VALUES ('Concurso de robotica', 8), ('Hackaton', 8), ('Congreso', 8);
INSERT INTO eventos (nombre, elemento_id) VALUES ('Reunion', 9), ('Titulacion', 9), ('Concurso programacion', 9);
INSERT INTO actividades (nombre, evento_id) VALUES
('actividad 1 evento 2', 2),
('actividad 2 evento 2', 2);
INSERT INTO actividades (nombre, evento_id) VALUES
('actividad 1 evento 3', 3);
INSERT INTO actividades (nombre, evento_id) VALUES
('actividad 1 evento 1', 1),
('actividad 2 evento 1', 1),
('actividad 3 evento 1', 1);
INSERT INTO actividades (nombre, evento_id) VALUES
('actividad 1 Concurso de robotica', 13),
('actividad 2 Concurso de robotica', 13),
('actividad 3 Concurso de robotica', 13);
INSERT INTO actividades (nombre, evento_id) VALUES
('actividad 1 Hackaton', 14),
('actividad 2 Hackaton', 14),
('actividad 3 Hackaton', 14);

INSERT INTO actividades (nombre, evento_id) VALUES
('actividad 1 Congreso', 15),
('actividad 2 Congreso', 15),
('actividad 3 Congreso', 15);
INSERT INTO actividades (nombre, evento_id) VALUES
('actividad 1 Reunion', 16),
('actividad 2 Reunion', 16),
('actividad 3 Reunion', 16);
INSERT INTO actividades (nombre, evento_id) VALUES
('actividad 1 Titulacion', 17),
('actividad 2 Titulacion', 17),
('actividad 3 Titulacion', 17);
INSERT INTO actividades (nombre, evento_id) VALUES
('actividad 1 Concurso programacion', 18),
('actividad 2 Concurso programacion', 18),
('actividad 3 Concurso programacion', 18);
```

Modelo Entidad-Relación



Relación entre elementos y eventos:

La tabla eventos tiene una llave foránea (elemento_id) que hace referencia a la llave primaria (id) de la tabla elementos.

Esto indica que cada evento está asociado a un único elemento del menú, creando una relación uno a muchos (un elemento puede tener muchos eventos, pero un evento pertenece a un solo elemento).

Relación entre eventos y actividades:

La tabla actividades tiene una llave foránea (evento_id) que hace referencia a la llave primaria (id) de la tabla eventos.

Esto establece una relación uno a muchos entre eventos y actividades (un evento puede tener muchas actividades, pero una actividad pertenece a un solo evento).

Nivel 1: elementos (elementos del menú principal).

Nivel 2: eventos (submenús asociados a los elementos del menú).

Nivel 3: actividades (submenús adicionales dentro de los eventos).

Backend y Frontend: Manual de usuario y Manual técnico

Dentro del **backend** tenemos:

Primeramente, debemos tener instaladas las dependencias necesarias con los comandos:

```
npm init -y
```

```
npm install express mysql2 cors
```

```
C:\Users\Usuario\buenomenu>npm install express mysql2 cors  
  
changed 1 package, and audited 1246 packages in 21s  
  
259 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities
```

En un archivo llamado server.js tenemos la **configuración a la BD** donde le indicamos el host, usuario, contraseña y nombre de la base de datos

```
const db = mysql.createConnection({  
  host: 'localhost',  
  user: 'root',  
  password: 'Ween551640',  
  database: 'menu_eventos_db'  
});
```

Ruta para obtener los elementos del menú

```
app.get('/elementos', (req, res) => {  
  db.query('SELECT * FROM elementos', (err, results) => {  
    if (err) throw err;  
    res.json(results);  
  });  
});
```

Ruta para obtener eventos de un elemento específico

```
app.get('/eventos/:elementoId', (req, res) => {  
  const { elementoId } = req.params;  
  db.query('SELECT * FROM eventos WHERE elemento_id= ?', [elementoId], (err, results) => {  
    if (err) throw err;  
    res.json(results);  
  });  
});
```

Ruta para obtener actividades de un evento específico

```
app.get('/actividades/:eventoId', (req, res) => {  
  const { eventoId } = req.params;  
  db.query('SELECT * FROM actividades WHERE evento_id = ?', [eventoId], (err, results) => {  
    if (err) throw err;  
    res.json(results);  
  });  
});
```

Y la iniciación del servidor

```
app.listen(port, () => {  
  console.log(`Servidor corriendo en http://localhost:\${port}`);  
});
```

Código completo:

```
JS server.js > ...
1  const express = require('express');
2  const mysql = require('mysql2');
3  const cors = require('cors');
4  const app = express();
5  const port = 3000;
6  app.use(cors());
7  app.use(express.json());
8  const db = mysql.createConnection({
9    host: 'localhost',
10   user: 'root',
11   password: 'Ween551640',
12   database: 'menu_eventos_db'
13 });
14 db.connect(err => {
15   if (err) throw err;
16   console.log('Conexión exitosa a MySQL');
17 });
18 app.get('/elementos', (req, res) => {
19   db.query('SELECT * FROM elementos', (err, results) => {
20     if (err) throw err;
21     res.json(results);
22   });
23 });
24 app.get('/eventos/:elementoId', (req, res) => {
25   const { elementoId } = req.params;
26   db.query('SELECT * FROM eventos WHERE elemento_id= ?', [elementoId], (err, results) => {
27     if (err) throw err;
28     res.json(results);
29   });
30 });
31 app.get('/actividades/:eventoId', (req, res) => {
32   const { eventoId } = req.params;
33   db.query('SELECT * FROM actividades WHERE evento_id = ?', [eventoId], (err, results) => {
34     if (err) throw err;
35     res.json(results);
36   });
37 });
38 app.listen(port, () => {
39   console.log(`Servidor corriendo en http://localhost:${port}`);
40 });
41
```

Iniciamos el servidor con “**node server.js**” para corroborar que no haya errores y corra bien

```
C:\Users\Usuario\buenomenu>node server.js
Servidor corriendo en http://localhost:3000
Conexión exitosa a MySQL
```

En la parte del **Frontend** se encuentran los archivos:

Html:

Este código crea el menú principal. En el encabezado, muestra el título "Menú Principal". Luego, genera dinámicamente un menú horizontal con elementos obtenidos de la variable `elementos`. Al hacer clic en un elemento, se cargan sus eventos relacionados, que aparecen en un menú vertical. Si se selecciona un evento, muestra sus actividades asociadas en otro menú vertical.

```
<ion-header>
  <ion-toolbar>
    <ion-title>Menú Principal</ion-title>
  </ion-toolbar>
</ion-header>
<div class="menu-horizontal">
  <div
    class="menu-item"
    *ngFor="let elemento of elementos"
    (click)="mostrarEventos(elemento.id)">
    {{ elemento.nombre }}
  </div>
</div>
<div *ngIf="eventos.length > 0" class="submenu-vertical">
  <h3>Eventos</h3>
  <div
    class="submenu-item"
    *ngFor="let evento of eventos"
    (click)="mostrarActividades(evento.id)">
    {{ evento.nombre }}
  </div>
</div>
<div *ngIf="actividades.length > 0" class="submenu-vertical-actividades">
  <h3>Actividades</h3>
  <div class="submenu-item-actividad" *ngFor="let actividad of actividades">
    {{ actividad.nombre }}
  </div>
</div>
```


Ts: Este código maneja la carga de elementos, eventos y actividades desde el servicio MenuService. Al iniciar, carga los elementos mediante la función cargarElementos(). Cuando se selecciona un elemento, se obtienen y muestran sus eventos relacionados con mostrarEventos(). Luego, al seleccionar un evento, se cargan y muestran las actividades asociadas a ese evento con la funciónmostrarActividades().

```
1  import { Component, OnInit } from '@angular/core';
2  import { MenuService } from '../menu.service';
3
4  @Component({
5    selector: 'app-home',
6    templateUrl: 'home.page.html',
7    styleUrls: ['home.page.scss'],
8  })
9  export class HomePage implements OnInit {
10    elementos: any[] = [];
11    eventos: any[] = [];
12    actividades: any[] = [];
13    constructor(private menuService: MenuService) {}
14    ngOnInit() {
15      this.cargarElementos();
16    }
17    cargarElementos() {
18      this.menuService.getElementos().subscribe(
19        (data) => {
20          console.log('Datos recibidos del servicio:', data);
21          this.elementos = data;
22        },
23        (error) => {
24          console.error('Error al obtener los elementos:', error);
25        }
26      );
27    }
28    mostrarEventos(elementoId: number) {
29      this.eventos = [];
30      this.actividades = [];
31      this.menuService.getEventos(elementoId).subscribe((data) => {
32        this.eventos = data;
33      });
34    }
35    mostrarActividades(eventoId: number) {
36      this.actividades = [];
37      this.menuService.getActividades(eventoId).subscribe((data) => {
38        this.actividades = data;
39      });
40    }
41  }
```

Se creo un servicio llamado menú para manejar lógica, compartir datos y funcionalidades entre el front y el back, **menú.service.ts**:

```
C:\Users\Usuario\buenomenu>ionic generate service menu
> ng.cmd generate service menu --project=app
CREATE src/app/menu.service.spec.ts (363 bytes)
CREATE src/app/menu.service.ts (142 bytes)
[OK] Generated service!
```

Este código se encarga de realizar solicitudes HTTP para obtener datos desde el servidor. Tiene tres métodos principales: `getElementos()` para obtener los elementos desde la ruta `/elementos`, `getEventos()` para obtener los eventos de un elemento específico mediante su ID en la ruta `/eventos/{elementId}`, y `getActividades()` para obtener las actividades de un evento específico a través de la ruta `/actividades/{eventId}`. Utiliza el cliente HTTP (`HttpClient`) para hacer estas peticiones y devolver los resultados como observables.

```
src > app > TS menu.service.ts > ...
1  import { Injectable } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3  import { Observable } from 'rxjs';
4
5  @Injectable({
6    providedIn: 'root',
7  })
8  export class MenuService {
9    private apiUrl = 'http://localhost:3000';
10
11    constructor(private http: HttpClient) {}
12
13    Tabnine | Edit | Test | Explain | Document | Ask
14    getElementos(): Observable<any> {
15      return this.http.get(`${this.apiUrl}/elementos`);
16    }
17
18    Tabnine | Edit | Test | Explain | Document | Ask
19    getEventos(elementId: number): Observable<any> {
20      return this.http.get(`${this.apiUrl}/eventos/${elementId}`);
21    }
22
23    Tabnine | Edit | Test | Explain | Document | Ask
24    getActividades(eventId: number): Observable<any> {
25      return this.http.get(`${this.apiUrl}/actividades/${eventId}`);
26    }
27  }
```

Importar HttpClientModule

Para poder hacer solicitudes HTTP en Angular utilizando el servicio. Este módulo es esencial para habilitar el uso del cliente HTTP (HttpClient) que se utiliza en los servicios para interactuar con el backend.

```
@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule, HttpClientModule],
  providers: [{ provide: RouteReuseStrategy, useClass: IonicRouteStrategy }, MenuService],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

Ejecutar frontend con “ionic serve”:

```
C:\Users\Usuario\buenomenu>ionic serve
> ng.cmd run app:serve --host=localhost --port=8100
[ng] - Generating browser application bundles (phase: setup)...
[ng] ✓ Browser application bundle generation complete.
[ng]
[ng] Initial chunk files
[ng] | Names | Raw size |
[ng] vendor.js | vendor | 4.23 MB |
[ng] polyfills.js | polyfills | 352.20 kB |
[ng] styles.css, styles.js | styles | 273.06 kB |
[ng] main.js | main | 17.48 kB |
[ng] runtime.js | runtime | 14.45 kB |
[ng] Build at: 2025-01-07T22:41:20.602Z - Hash: 1e348bf60199cafb - Time: 14645ms
[ng] ✓ Compiled successfully.
```

Menú principal:

Menú Principal

elemento m

elemento n

eventos

Menú mostrando elementos y eventos (eventos del elemento m):

elemento m

elemento n

eventos

Eventos

- Reunion
- Titulacion
- Concurso programacion

Menú mostrando elementos, eventos y actividades (eventos del elemento m y actividades del evento reunión):

elemento m

elemento n

eventos

Eventos

- Reunion
- Titulacion
- Concurso programacion

Actividades

- actividad 1 Reunion
- actividad 2 Reunion
- actividad 3 Reunion

Menú mostrando elementos y eventos (eventos del elemento n):

elemento m

elemento n

eventos

Eventos

Concurso de robotica

Hackaton

Congreso

Menú mostrando elementos, eventos y actividades (eventos del elemento n y actividades del evento Hackaton):

elemento m

elemento n

eventos

Eventos

Concurso de robotica

Hackaton

Congreso

Actividades

actividad 1 Hackaton

actividad 2 Hackaton

actividad 3 Hackaton

Menú mostrando elementos y eventos (eventos del elemento eventos):

elemento m

elemento n

eventos

Eventos

evento 1

evento 2

evento 3

Menú mostrando elementos, eventos y actividades (eventos del elemento eventos y actividades del evento 1):

elemento m

elemento n

eventos

Eventos

evento 1

evento 2

evento 3

Actividades

actividad 1 evento 1

actividad 2 evento 1

actividad 3 evento 1

Subir en Github en cada rama: Proyecto completo (bd, front y back) y documentación



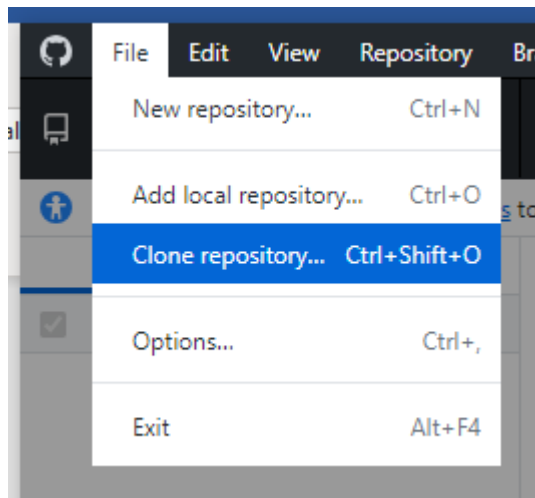
[JohannHG](#) invited you to collaborate on
JohannHG/Proyecto2ParcialBDDM

Accept invitation

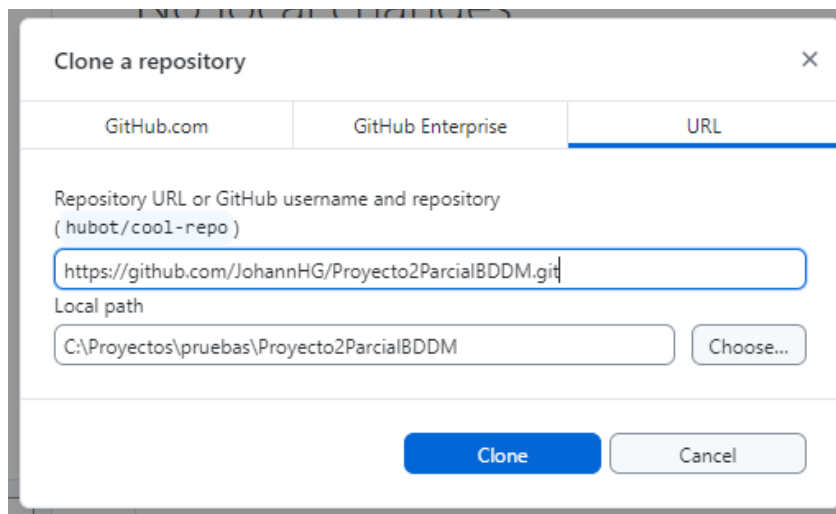
Decline invitation

Clonar el repositorio en tu computadora:

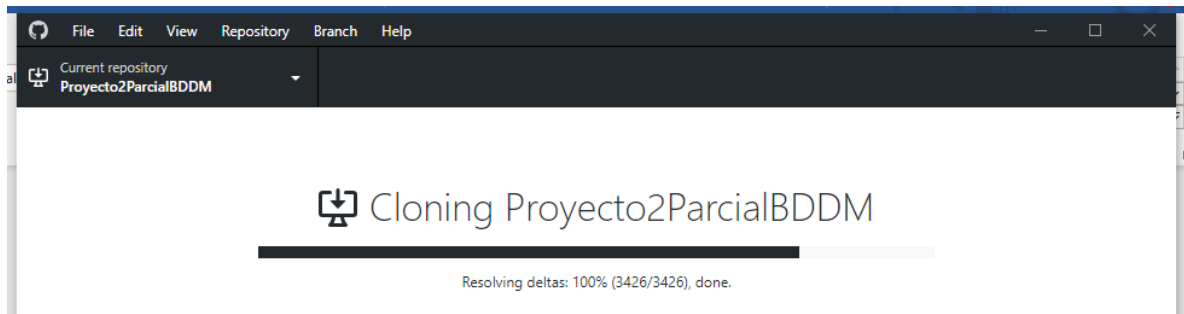
- Abrir GitHub Desktop y seleccionar **"File > Clone repository"**.



- Eliger la opción **"URL"** y pega la URL del repositorio



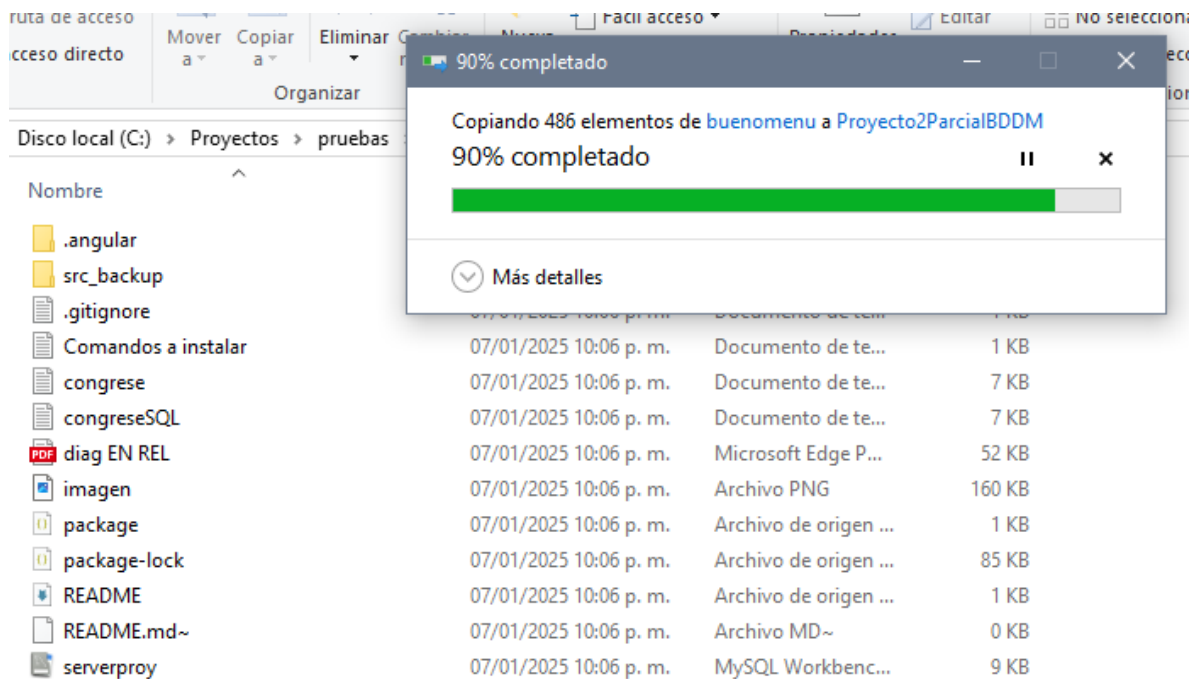
- Selecciona una carpeta local donde se descargará el repositorio.



Copiar tus archivos al repositorio local:

En la carpeta donde se clono el repositorio.

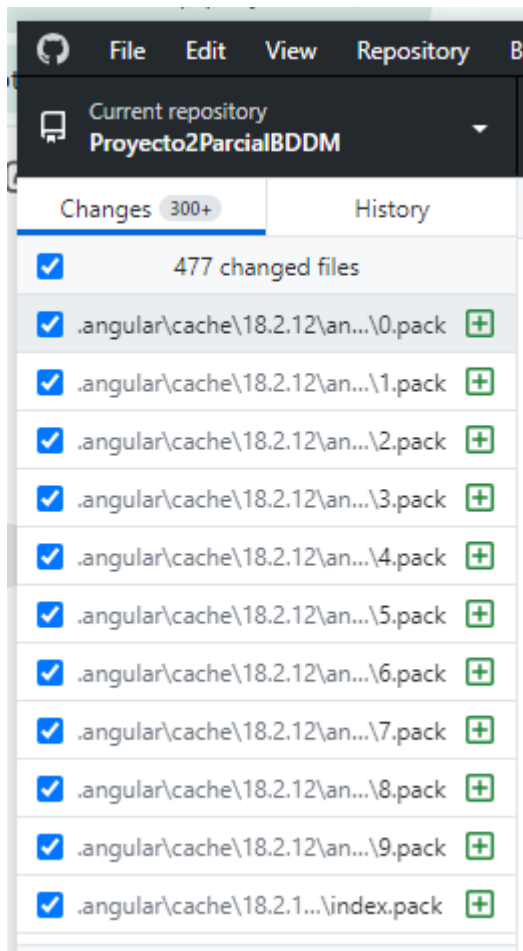
Copiar los archivos del proyecto Ionic dentro de esta carpeta



Verifica los cambios en GitHub Desktop:

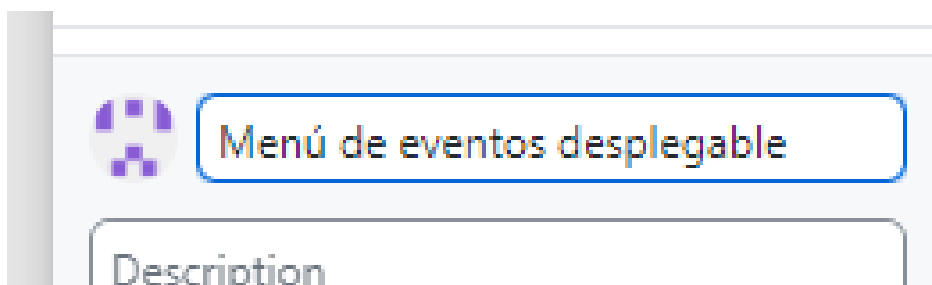
Abrir GitHub Desktop.

En la pestaña "Changes", se deben ver todos los archivos nuevos y modificados.

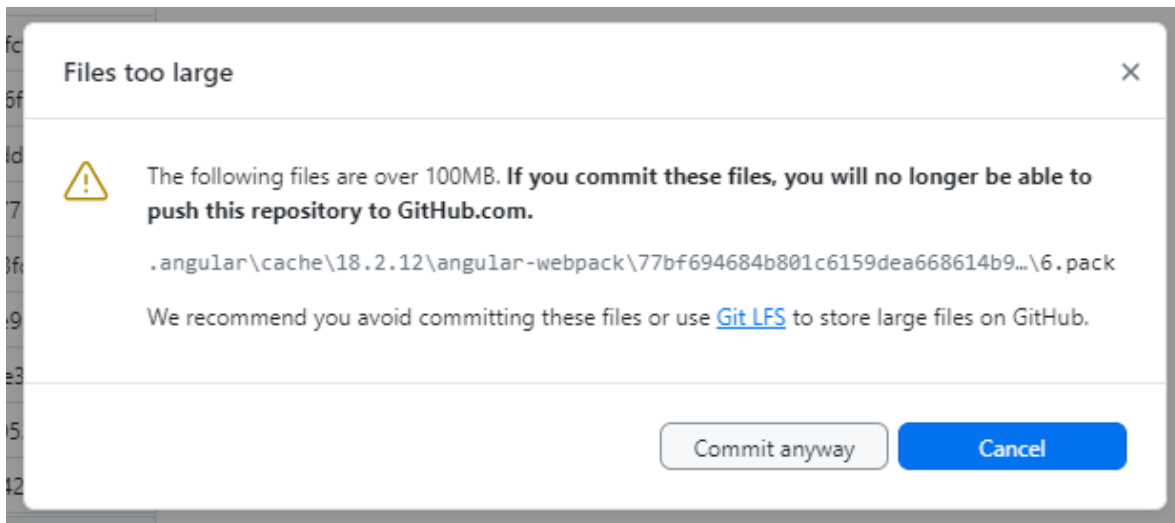
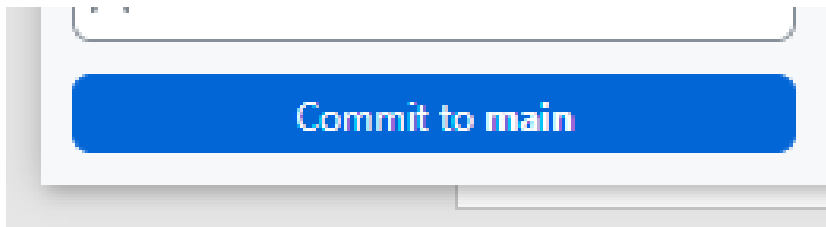


Hacer un commit:

En GitHub Desktop, escribir un mensaje descriptivo para los cambios en la sección "Summary" (en nuestro caso: "Menú de eventos desplegable").



Hacer clic en "Commit to main".



No quería perjudicar a los compañeros y ese mensaje me dio miedo):entonces hice uno nuevo y le mande solicitud

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *

 Wendybelen30

Repository name *

Menu desplegable

✓ Your new repository will be created as Menu-desplegable-.
The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

Great repository names are short and memorable. Need inspiration? How about **turbo-potato** ?

Description (optional)

Proyecto tercer parcial



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)







Choose a license

License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

<input type="checkbox"/>	 Gcortes7 Awaiting Gcortes7's response	Pending Invite 	
<input type="checkbox"/>	 Karmi Yajseel Ramírez López Awaiting Karmi-Yajseel's response	Pending Invite 	

Clone a repository

GitHub.com

GitHub Enterprise

URL

Repository URL or GitHub username and repository
(hubot/cool-repo)

https://github.com/Wendybel30/Menu-desplegable-.git

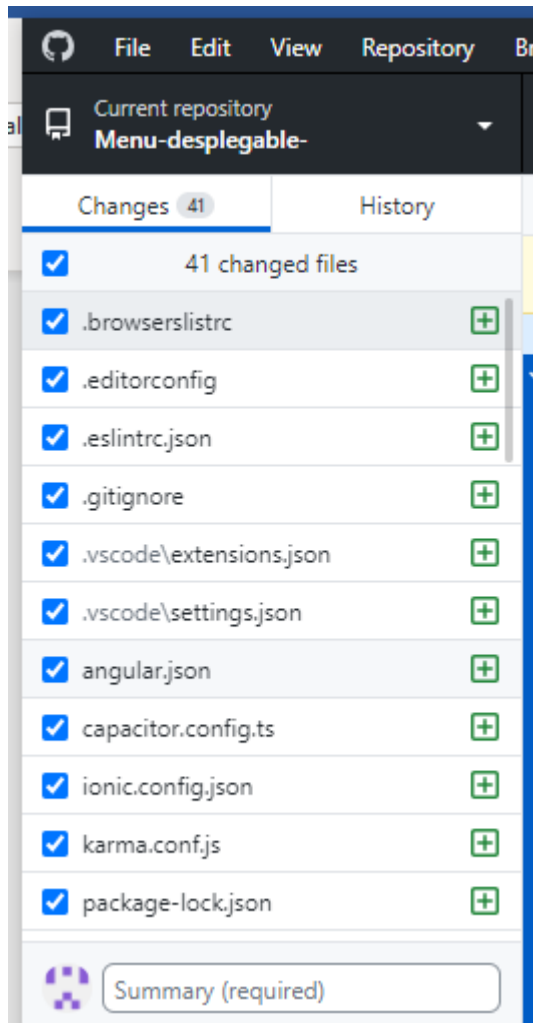
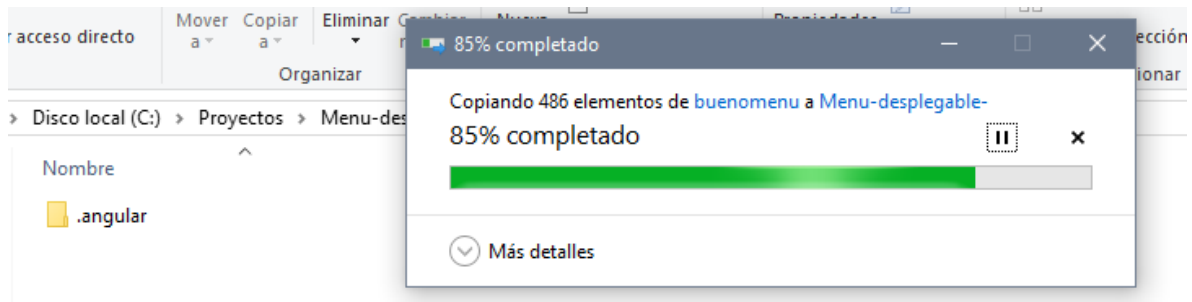
Local path

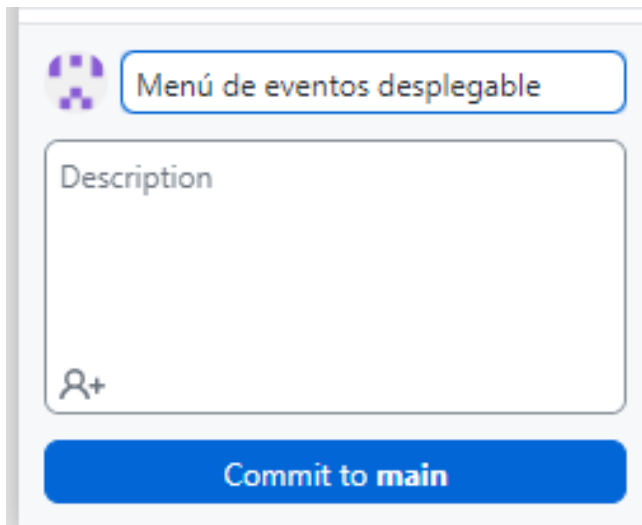
C:\Proyectos\Menu-desplegable-

Choose...

Clone

Cancel





Y ya están los archivos en github

