INSTITUTO TECNOLOGICO
DEL VALLE DE OAXACA

TECNOLÓGICO
NACIONAL DE MÉXICO

INGENIERIA EN TECNOLOGIAS DE LA
INFORMACION Y COMUNICACIONES

TALLER DE BASE DE DATOS

GRUPO: TIC'S 4'A

NOMBRE:CARLOS EDUARDO BALTAZAR RAMIREZ

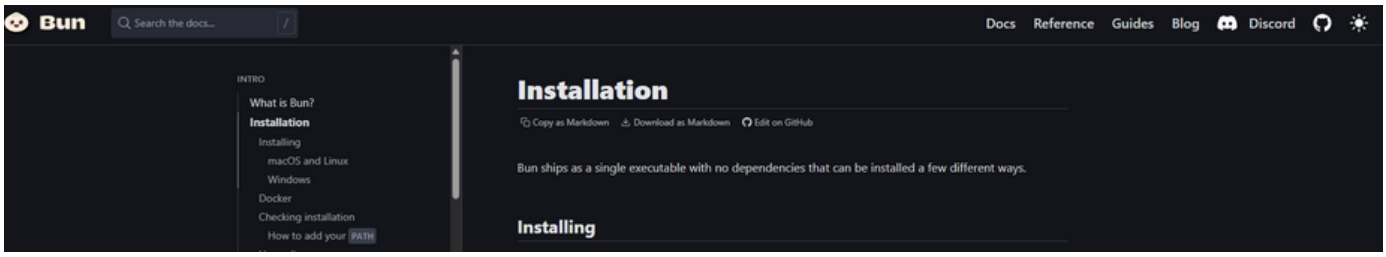
Aplicación con conexión a Base de Datos

30 de mayo DEL 2025

JAVASCRIPT POSTGREST-MAIN

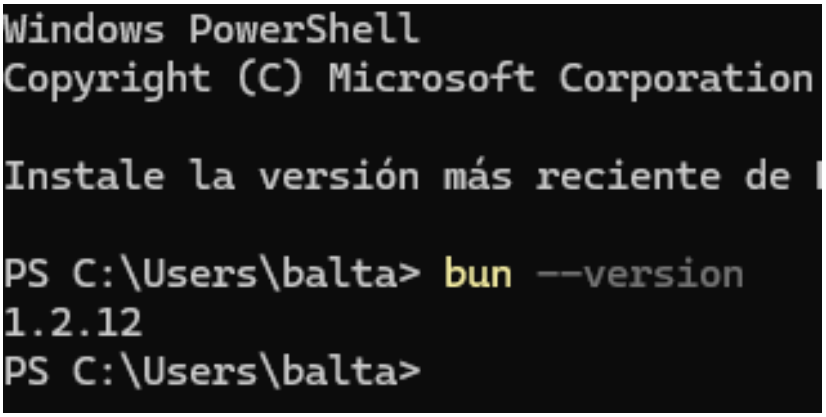
instalacion del bun

1.Nos dirigimos hacia la pagina de instalacion "Bun" : https://bun.sh/docs/installation



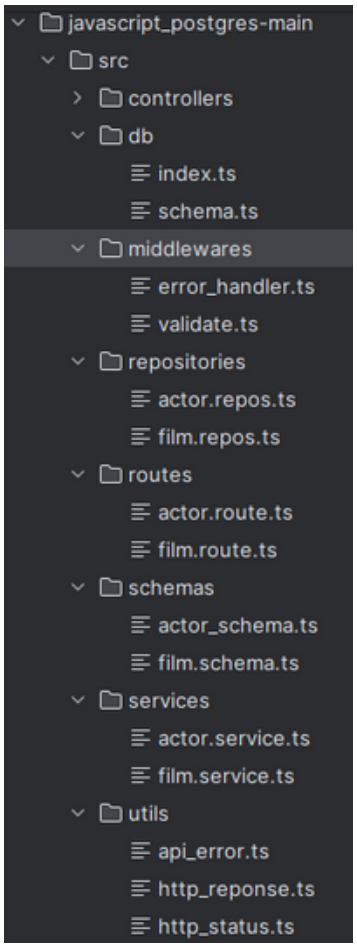
2. Una vez ya dentro, escogemos el sistema operativo en mi caso es Windows, nos dirigiremos a los codigos que nos proporciona para su instalacion desde Powershell o CMD.

3. Copiaremos el codigo de powershell abriremos una nueva ventana de este mismo programa y pegaremos el cdigo una vez hecho, para revisar si se instalo la ultima version metemos el siguiente comando



CONEXION DE BASE DE DATOS

Estructura del proyecto:



En esta parte creamos 2 archivos de “TS” en el cual vamos a hacer la exportacion de 2 tablas

schema.ts

```ts
import { pgTable, serial,  varchar, integer } from 'drizzle-orm/pg-core';

export const actors = pgTable('actor', {
    actor_id: serial('actor_id').primaryKey(),
    first_name: varchar('first_name', { length: 100 }),
    last_name: varchar('last_name', { length: 100 })
});

export const films = pgTable('film', {
    film_id: serial('film_id').primaryKey(),
    title: varchar('title', { length: 255 }),
    description: varchar('description', { length: 500 }),
    release_year: integer('release_year'),
    language_id:integer('language_id'),
    rental_duration: integer ('rental_duration'),
    length: integer('length'),
});
```

Tabla film

actor.repos.ts | film.repos.ts

```ts
import { db } from '../db/connection.ts';
import { actors } from '../db/schema.ts';
import { eq } from "drizzle-orm/sql/expressions/conditions";

export const ActorRepository = {
    findAll: async () => db.select().from(actors),
    findById: async (id: number) => {
        const [actor] = await db
            .select()
            .from(actors)
            .where(eq(actors.actor_id, id));
        return actor;
    },
    add: async (data: { first_name: string; last_name: string }) =>
        db.insert(actors).values(data).returning(),
};
```

En esta carpeta, se hace las consutas SQL dependiendo del ID que quiera colocar el usuario, en los archivos de TS de actor y film

actor.repos.ts | film.repos.ts

```ts
import { db } from '../db';
import { films } from '../db/schema.ts';
import { eq } from "drizzle-orm/sql/expressions/conditions";

export const filmRepository = {
    findAll: async () => db.select().from(films),
    findById: async (id: number) => {
        const [film] = await db
            .select()
            .from(films)
            .where(eq(films.film_id, id));
        return film;
    },
    add: async (data: { title: string; description: string, rental_duration: number }) =>
        db.insert(films).values(data).returning(),
};
```

En esta carpeta, se define las validaciones de cada atributo que asignamos anteriormente,

actor_schema.ts | film.schema.ts

```ts
import { z } from 'zod';

export const actorSchema = z.object({
    first_name: z.string().min(1, "El nombre es obligatorio"),
    last_name: z.string().min(1, "El apellido es obligatorio"),
});
```

actor_schema.ts | film.schema.ts

```ts
import { z } from 'zod';

export const FilmSchema = z.object({
    title: z.string(),
    description: z.string(),
    rental_duration: z.number()
});

export type FilmInput = z.infer<typeof FilmSchema>;
```

Aqui deben de editarse como se debe de ver los resultados, en este caso como tipo JSON y como queremos que los visualice el usuario

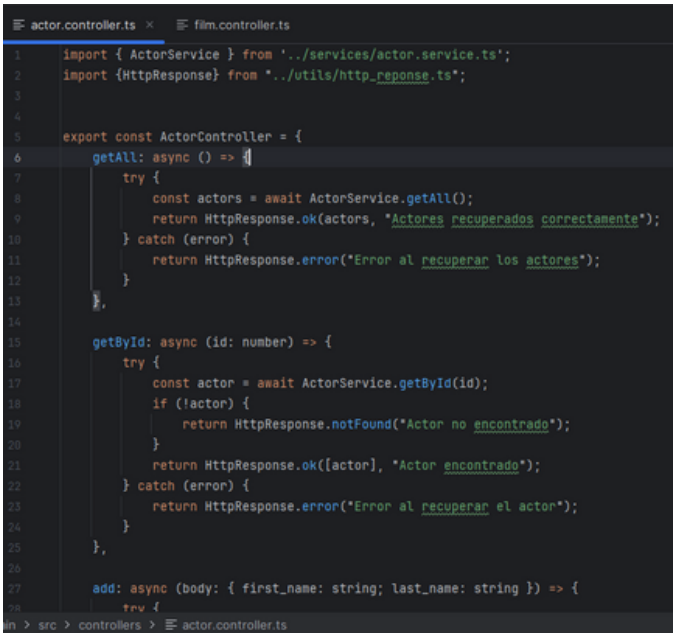
actor.service.ts | film.service.ts

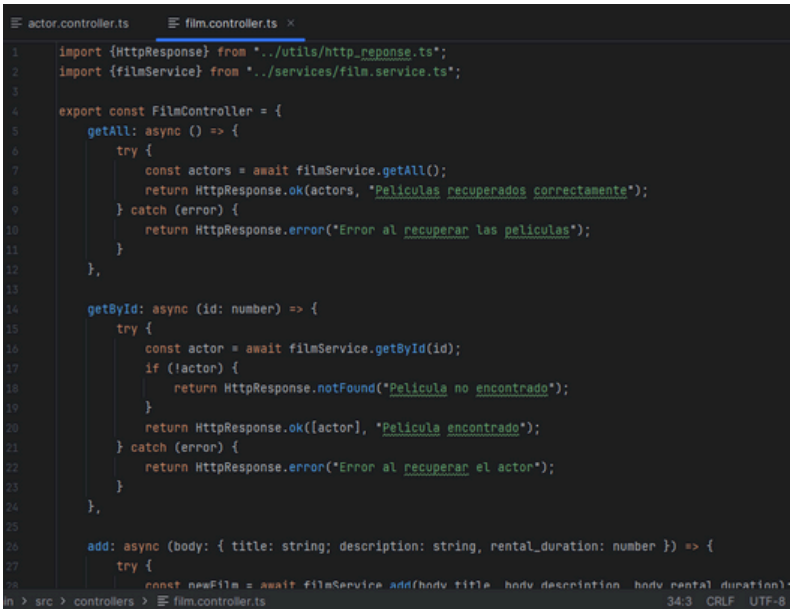
```ts
import { ActorRepository } from '../repositories/actor.repos.ts';

export const ActorService = {
    getAll: () => ActorRepository.findAll(),
    getById: (id: number) => ActorRepository.findById(id),
    add: (first_name: string, last_name: string) =>
        ActorRepository.add({ first_name, last_name }),
};
```

actor.service.ts | film.service.ts

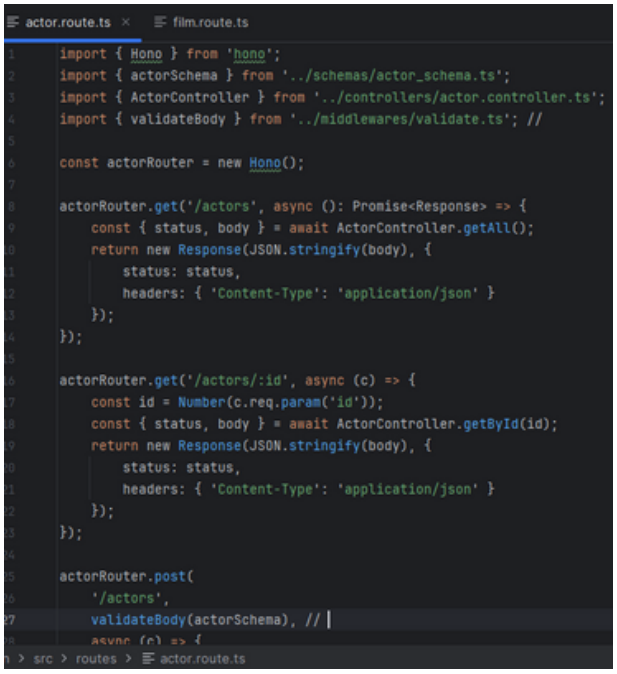
```ts
import { filmRepository } from '../repositories/film.repos.ts';
export const filmService = {
    getAll: () => filmRepository.findAll(),
    getById: (id: number) => filmRepository.findById(id),
    add: (title: string, description: string, rental_duration:number) =>
        filmRepository.add({ title, description, rental_duration }),
};
```





En esta carpeta se crean los getters y los metodos de añadir para actores y para peliculas, ademas de que se añade al mismo BD cuando desea el usuario añadir peliculas o actores

En esta carpeta define las rutas y asocia el controlador.



app.ts

```ts
import { Hono } from 'hono';
import { db } from './src/db/index';
import { actors } from './src/db/schema';

const app = new Hono();

// Ruta GET /actors
app.get('/actors', async (c) => {
  try {
    const result = await db.select().from(actors).execute();
    return c.json(result);
  } catch (error) {
    console.error('Database error:', error);
    return c.json({ error: 'Error al obtener actores' }, 500);
  }
});

// Ruta GET /films
app.get('/films', async (c) => {
  try {
    const result = await db.select().from(films).execute();
    return c.json(result);
  } catch (error) {
    console.error('Database error:', error);
    return c.json({ error: 'Error al obtener películas' }, 500);
  }
});
```

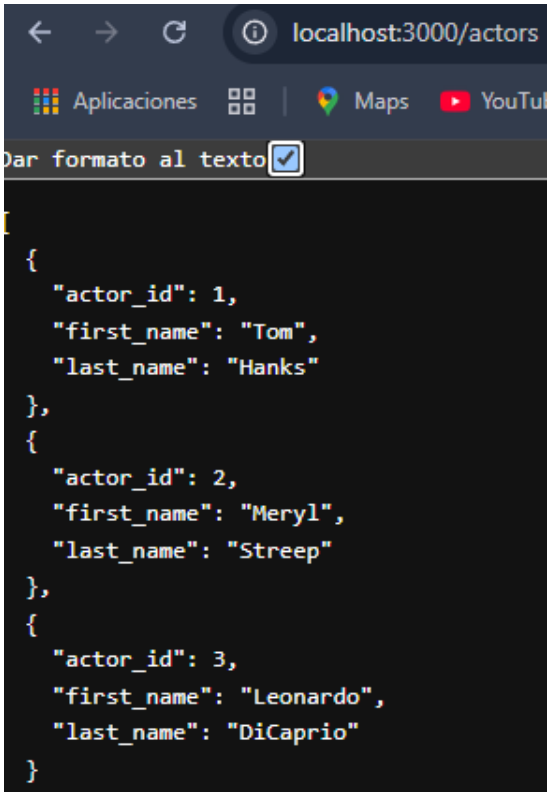
n > app.ts

En esta carpeta define las rutas para su ejecutable y se pueda hacer en el navegador.

C:\Windows\System32\cmd.e

```
Microsoft Windows [Versión 10.0.26100.4061]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\balta\Downloads\javascript_postgres-main (1)\javascript_postgres-main>bun server.ts
 Servidor escuchando en http://localhost:3000
```




```
[
  {
    "actor_id": 1,
    "first_name": "Tom",
    "last_name": "Hanks"
  },
  {
    "actor_id": 2,
    "first_name": "Meryl",
    "last_name": "Streep"
  },
  {
    "actor_id": 3,
    "first_name": "Leonardo",
    "last_name": "DiCaprio"
  }
```


En cmd de la carpeta donde viene todo, ejecutamos con “Bun” server.ts y en el navegador se puede ver por medio del localHost colocando el simbolo “\” y despues escribiendo ya sea “actors” o “films”

EVALUACION DOCENTE CON PHP

Descargar PHP completo con extensiones
https://windows.php.net/download/

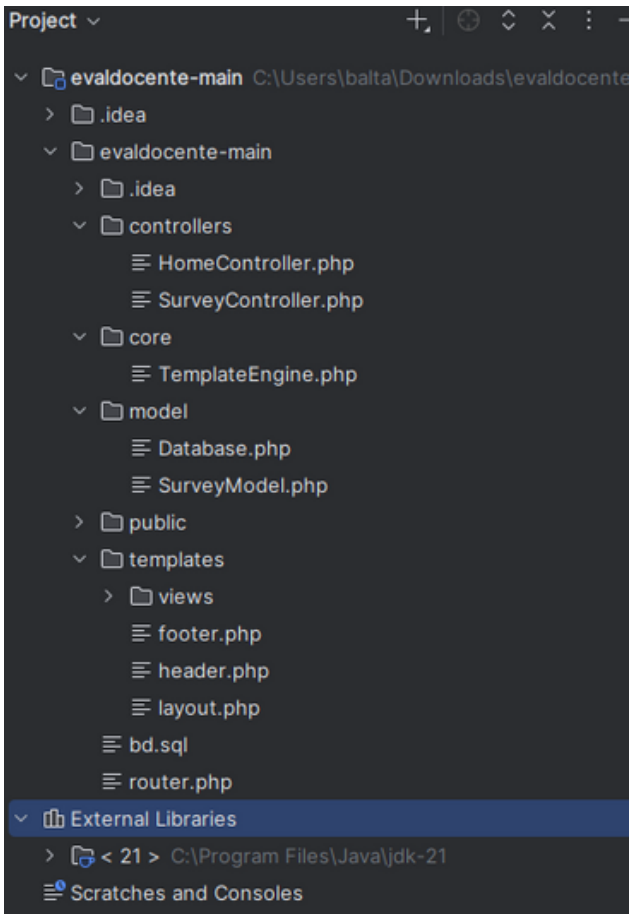
VS17 x64 Thread Safe (2025-May-06 14:18:08)

- Zip [32.37MB]
  sha256: b2cce3ca2ae599ef1b13ae0b84b17aa25f71a94e3098d52c7407238375c1056a
- Debug Pack [37.13MB]
  sha256: 7d4d38328e9766a22548602696dc82f7f76e79ffe32d3570c5fb067fbea621ab
- Development package (SDK to develop PHP extensions) [1.35MB]
  sha256: 1402f45e0910b600482e173821deda0de4987b54f956300cee9109e54e1594b1

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\balta> php --version
PHP 8.4.7 (cli) (built: May  6 2025 14:12:45) (ZTS Visual C++ 2022 x64)
Copyright (c) The PHP Group
```



Estructura del proyecto:

TemplateEngine.php

```php
<?php

namespace core;

class TemplateEngine
{
    public static function render($view, $data = [])
    {
        extract($data);
        ob_start();
        include __DIR__ . "/../templates/views/$view.php";
        $content = ob_get_clean();
        include __DIR__ . "/../templates/layout.php";
    }

    public static function renderPartial($view, $data = [])
    {
        extract($data);
        include __DIR__ . "/../templates/views/$view.php";
    }
}
```

Motor de plantillas para renderizar vistas.

Conexión a maria db

Database.php SurveyModel.php

```php
<?php

namespace model;

class Database {
    private static $conn;


    private function __construct() {}

    public static function getConnection() {
        if (self::$conn === null) {
            self::$conn = new \mysqli("127.0.0.1", "root", "123987", "sample", 3306);
            if (self::$conn->connect_error) {
                throw new \Exception("Error de conexión: " . self::$conn->connect_error);
            }
        }
        return self::$conn;
    }
}
```

Database.php SurveyModel.php

```php
<?php

namespace model;

use http\Params;

class SurveyModel {
    private $conn;

    public function __construct($conn) {
        $this->conn = $conn;
    }

    public function save($docente, $p1, $p2, $p3, $comentarios) {
        $stmt = $this->conn->prepare(
            "INSERT INTO respuestas (docente, pregunta1, pregunta2, pregunta3, comentarios)
             VALUES (?, ?, ?, ?, ?)"
        );

        $stmt->bind_param("siiis", $docente, $p1, $p2, $p3, $comentarios);

        if ($stmt->execute()) {
            return [true, "Gracias por enviar tu evaluación."];
        } else {
            return [false, "Error al guardar los datos: " . $stmt->error];
        }
    }
```

.htaccess index.php

```php
<?php

require_once dirname(__DIR__) . '/core/TemplateEngine.php';
const BASE_URL = '/';

spl_autoload_register(function ($class) {
    $baseDir = dirname(__DIR__);
    $path = $baseDir . '/' . str_replace('\\', '/', $class) . '.php';

    if (file_exists($path)) {
        require_once $path;
    }
});

// Obtener la ruta
$uri = trim(parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH), '/');
$parts = explode('/', $uri);

// Asignar controlador y método por defecto
$controller = !empty($parts[0]) ? ucfirst($parts[0]) . 'Controller' : 'HomeController';
$method = $parts[1] ?? 'index';

// Namespace completo del controlador
$controllerFQN = "controllers\\$controller";

// Ejecutar
if (class_exists($controllerFQN) && method_exists($controllerFQN, $method)) {
    call_user_func([$controllerFQN, $method]);
```

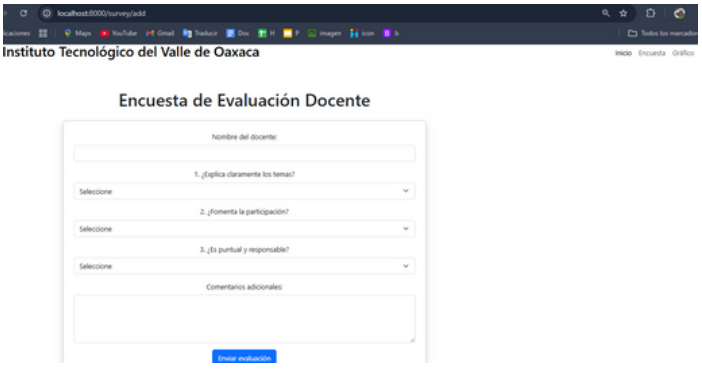
es el punto de entrada principal del servidor backend construido con php y mariadb. Aquí se configura el servidor HTTP, las rutas y la conexión a la base de datos.

Iniciar el servidor PHP:

- **Página principal: http://localhost:8000**

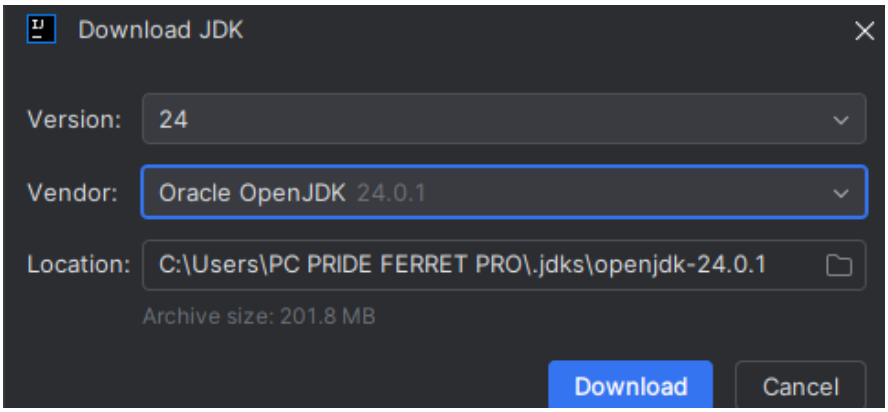
```
PS C:\Users\balta> cd C:\Users\balta\Downloads\evaldocente-main\evaldocente-main
PS C:\Users\balta\Downloads\evaldocente-main\evaldocente-main> php -S localhost:8000 router.php
[Fri May 30 20:39:34 2025] PHP 8.4.7 Development Server (http://localhost:8000) started
```

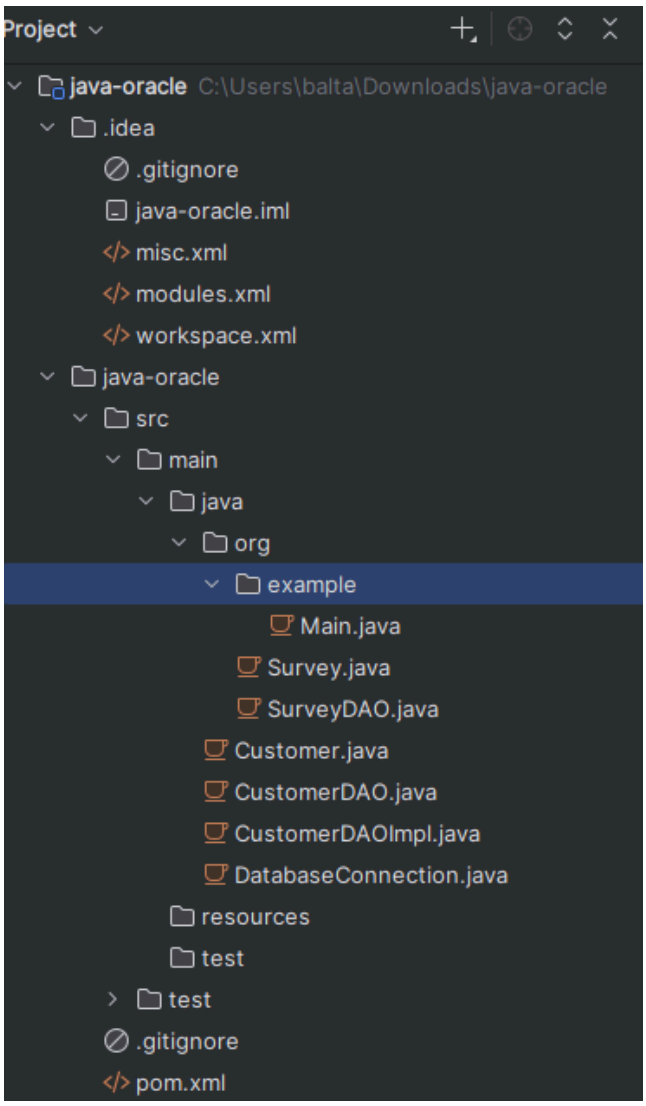
Formulario de encuesta: en http://localhost:8000/survey



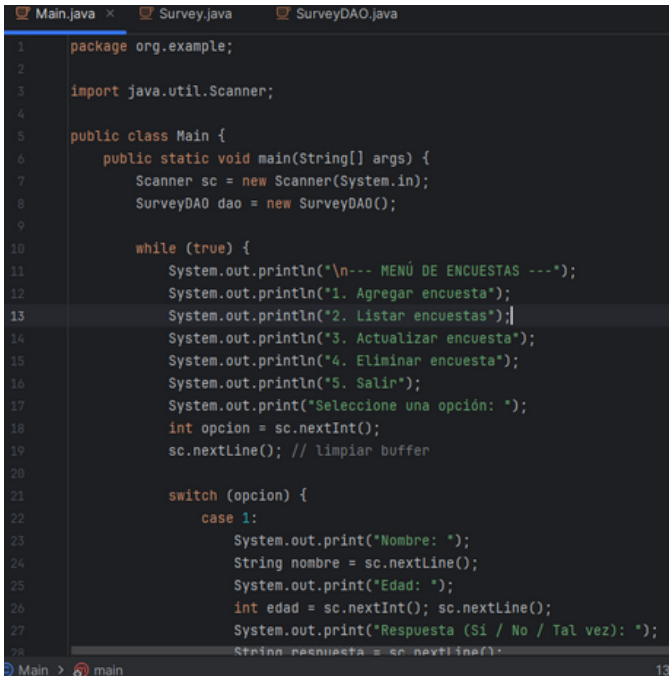
JAVA ORACLE

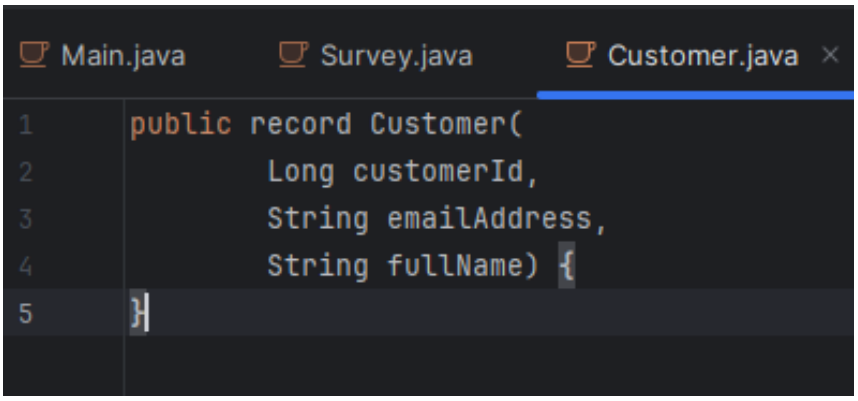
primero instale el jdk



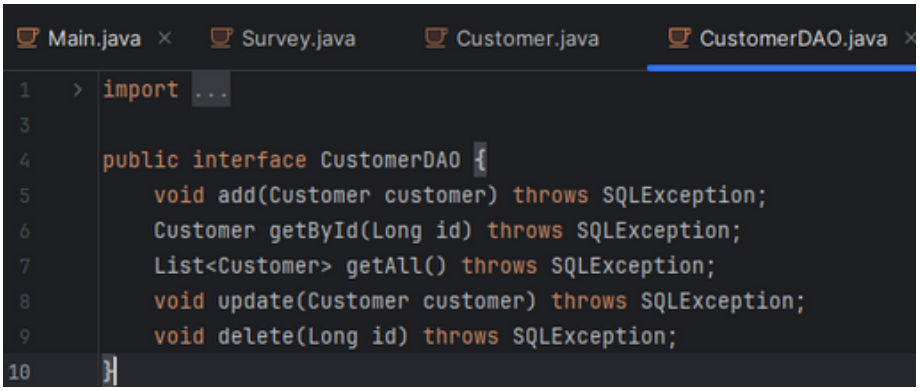


Estructura del proyecto:





Representa la tabla customers en la base de datos



Define las operaciones CRUD:

Main.java | Survey.java | Customer.java | CustomerDAO.java | CustomerDAOImpl.java

```java
import ...

public class CustomerDAOImpl implements CustomerDAO {
    private Connection connection;
    public CustomerDAOImpl(Connection connection) { this.connection = connection; }

    public void add(Customer customer) throws SQLException {
        String sql = "INSERT INTO customers (customer_id, email_address, full_name) VALU
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setLong(1, customer.customerId());
            stmt.setString(2, customer.emailAddress());
            stmt.setString(3, customer.fullName());
            stmt.executeUpdate();
        }
    }

    public Customer getById(Long customerId) throws SQLException {
        String sql = "SELECT * FROM customers WHERE customer_id = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setLong(1, customerId);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                return new Customer(
                        rs.getLong("customer_id"),
                        rs.getString("email_address"),
                        rs.getString("full_name")
                );
            }
```

Usa JDBC para ejecutar consultas:

Customer.java | CustomerDAO.java | CustomerDAOImpl.java | DatabaseConnection.java

```java
import ...

public class DatabaseConnection {
    private static final String URL = "jdbc:oracle:thin:@//172.17.0.2:1521/XE";
    private static final String USER = "root";
    private static final String PASSWORD = "root";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

Maneja la conexión a la base de datos