



**INSTITUTO TECNOLÓGICO
DEL VALLE DE OAXACA**



**INGENIERIA EN TECNOLOGIAS DE LA
INFORMACION Y COMUNICACIONES**



TALLER DE BASE DE DATOS

GRUPO: TIC'S 4'A

NOMBRE: CARLOS EDUARDO BALTAZAR RAMIREZ

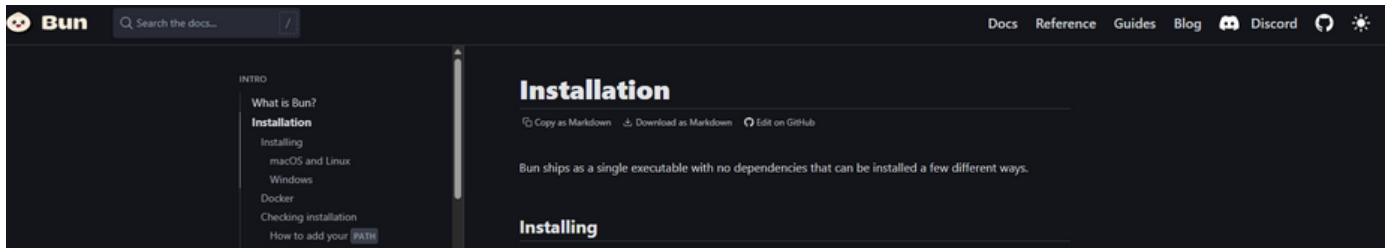
Aplicación con conexión a Base de Datos

30 de mayo DEL 2025

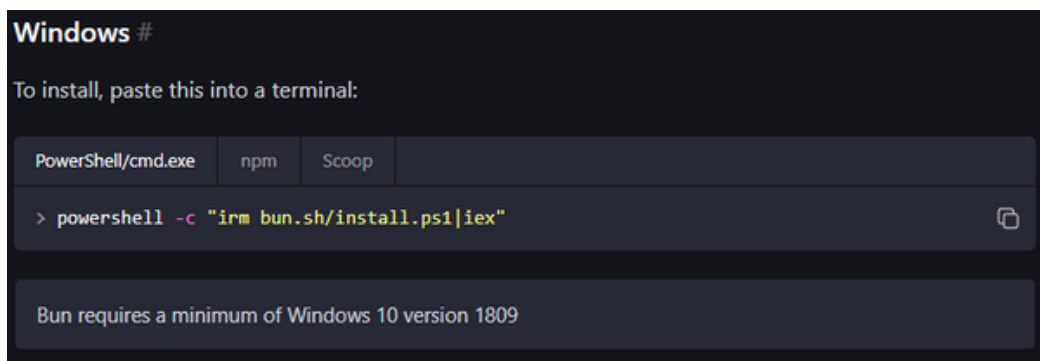
JAVASCRIPT POSTGREST-MAIN

instalacion del bun

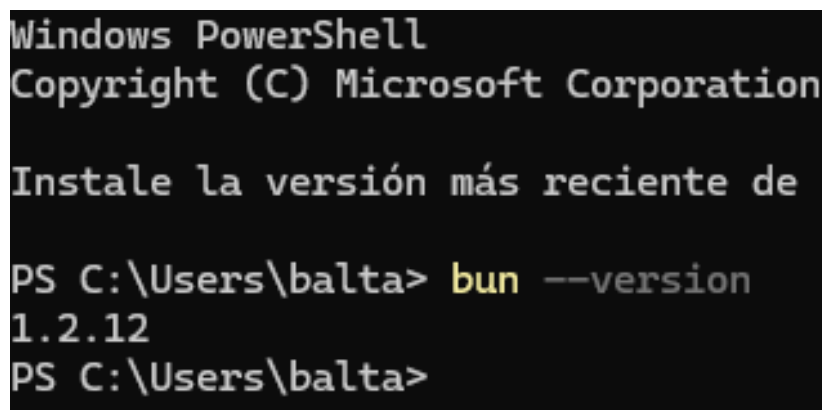
1. Nos dirigimos hacia la pagina de instalacion "Bun" : <https://bun.sh/docs/installation>



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.

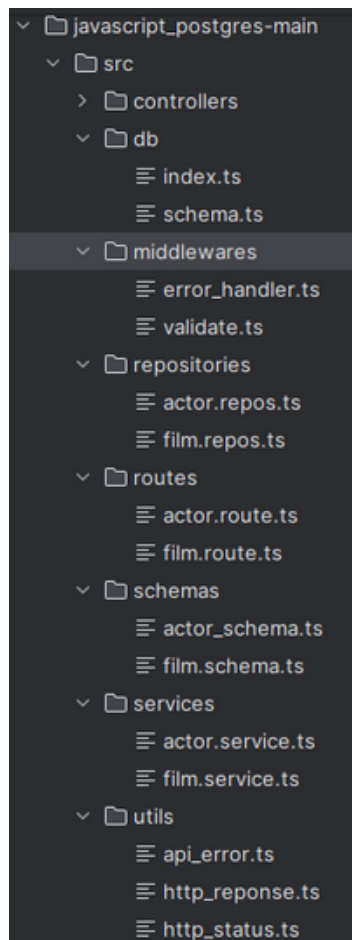


3. Copiaremos el codigo de powershell abriremos una nueva ventana de este mismo programa y pegaremos el cdigo una vez hecho, para revisar si se instalo la ultima version metemos el siguiente comando



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
1 import { pgTable, serial, varchar, integer } from 'drizzle-orm/pg-core';
2
3 export const actors = pgTable('actor', {
4   actor_id: serial('actor_id').primaryKey(),
5   first_name: varchar('first_name', { length: 100 }),
6   last_name: varchar('last_name', { length: 100 })
7 });
8
9 export const films = pgTable('film', {
10   film_id: serial('film_id').primaryKey(),
11   title: varchar('title', { length: 255 }),
12   description: varchar('description', { length: 500 }),
13   release_year: integer('release_year'),
14   language_id: integer('language_id'),
15   rental_duration: integer('rental_duration'),
16   length: integer('length'),
17 });
```

Tabla film

```
actor.repos.ts x film.repos.ts
1 import { db } from '../db/connection.ts';
2 import { actors } from '../db/schema.ts';
3 import { eq } from "drizzle-orm/sql/expressions/conditions";
4
5 export const ActorRepository = {
6   findAll: async () => db.select().from(actors),
7   findById: async (id: number) => {
8     const [actor] = await db
9       .select()
10      .from(actors)
11      .where(eq(actors.actor_id, id));
12     return actor;
13   },
14   add: async (data: { first_name: string; last_name: string }) =>
15     db.insert(actors).values(data).returning(),
16 };
```

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

```
actor.repos.ts x film.repos.ts x
1 import { db } from '../db';
2 import { films } from '../db/schema.ts';
3 import { eq } from "drizzle-orm/sql/expressions/conditions";
4
5 export const filmRepository = {
6   findAll: async () => db.select().from(films),
7   findById: async (id: number) => {
8     const [film] = await db
9       .select()
10      .from(films)
11      .where(eq(films.film_id, id));
12     return film;
13   },
14   add: async (data: { title: string; description: string; rental_duration: number }) =>
15     db.insert(films).values(data).returning(),
16 };
```

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

```
actor_schema.ts x film.schema.ts
1 import { z } from 'zod';
2
3 export const actorSchema = z.object({
4   first_name: z.string().min(1, "El nombre es obligatorio"),
5   last_name: z.string().min(1, "El apellido es obligatorio"),
6 });
```

```
actor_schema.ts x film.schema.ts x
1 import { z } from 'zod';
2
3 export const FilmSchema = z.object({
4   title: z.string(),
5   description: z.string(),
6   rental_duration: z.number()
7 });
8
9 export type FilmInput = z.infer<typeof FilmSchema>;
```

Aquí 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 x film.service.ts
1 import { ActorRepository } from '../repositories/actor.repos.ts';
2
3 export const ActorService = {
4   getAll: () => ActorRepository.findAll(),
5   getById: (id: number) => ActorRepository.findById(id),
6   add: (first_name: string, last_name: string) =>
7     ActorRepository.add({ first_name, last_name }),
8 };
```

```
actor.service.ts x film.service.ts x
1 import { filmRepository } from '../repositories/film.repos.ts';
2 export const filmService = {
3   getAll: () => filmRepository.findAll(),
4   getById: (id: number) => filmRepository.findById(id),
5   add: (title: string, description: string, rental_duration: number) =>
6     filmRepository.add({ title, description, rental_duration }),
7 };
```

```

1 import { ActorService } from '../services/actor.service.ts';
2 import { HttpResponse } from '../utils/http_response.ts';
3
4
5 export const ActorController = {
6   getAll: async () => {
7     try {
8       const actors = await ActorService.getAll();
9       return HttpResponse.ok(actors, "Actores recuperados correctamente");
10     } catch (error) {
11       return HttpResponse.error("Error al recuperar los actores");
12     }
13   },
14
15   getById: async (id: number) => {
16     try {
17       const actor = await ActorService.getById(id);
18       if (!actor) {
19         return HttpResponse.notFound("Actor no encontrado");
20       }
21       return HttpResponse.ok([actor], "Actor encontrado");
22     } catch (error) {
23       return HttpResponse.error("Error al recuperar el actor");
24     }
25   },
26
27   add: async (body: { first_name: string; last_name: string }) => {
28     try {
29

```

```

1 import { HttpResponse } from '../utils/http_response.ts';
2 import { FilmService } from '../services/film.service.ts';
3
4 export const FilmController = {
5   getAll: async () => {
6     try {
7       const actors = await filmService.getAll();
8       return HttpResponse.ok(actors, "Películas recuperados correctamente");
9     } catch (error) {
10      return HttpResponse.error("Error al recuperar las películas");
11    }
12  },
13
14   getById: async (id: number) => {
15     try {
16       const actor = await filmService.getById(id);
17       if (!actor) {
18         return HttpResponse.notFound("Película no encontrado");
19       }
20       return HttpResponse.ok([actor], "Película encontrado");
21     } catch (error) {
22       return HttpResponse.error("Error al recuperar el actor");
23     }
24  },
25
26   add: async (body: { title: string; description: string; rental_duration: number }) => {
27     try {
28       const newFilm = await filmService.add(body.title, body.description, body.rental_duration);
29     }
30   }
31 }

```

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.

```

1 import { Hono } from 'hono';
2 import { actorSchema } from '../schemas/actor_schema.ts';
3 import { ActorController } from '../controllers/actor.controller.ts';
4 import { validateBody } from '../middlewares/validate.ts'; //
5
6 const actorRouter = new Hono();
7
8 actorRouter.get('/actors', async () : Promise<Response> => {
9   const { status, body } = await ActorController.getAll();
10   return new Response(JSON.stringify(body), {
11     status: status,
12     headers: { 'Content-Type': 'application/json' }
13   });
14 });
15
16 actorRouter.get('/actors/:id', async (c) => {
17   const id = Number(c.req.param('id'));
18   const { status, body } = await ActorController.getById(id);
19   return new Response(JSON.stringify(body), {
20     status: status,
21     headers: { 'Content-Type': 'application/json' }
22   });
23 });
24
25 actorRouter.post(
26   '/actors',
27   validateBody(actorSchema), //
28   async (c) => {
29

```

```

1 import { Hono } from 'hono';
2 import { FilmSchema, type FilmInput } from '../schemas/film.schema';
3 import { FilmController } from '../controllers/film.controller';
4 import { validateBody } from '../middlewares/validate';
5
6 const filmRouter = new Hono();
7
8 filmRouter.get('/films', async () : Promise<Response> => {
9   const { status, body } = await FilmController.getAll();
10   return new Response(JSON.stringify(body), {
11     status: status,
12     headers: { 'Content-Type': 'application/json' }
13   });
14 });
15
16 filmRouter.get('/films/:id', async (c): Promise<Response> => {
17   const id = Number(c.req.param('id'));
18   const { status, body } = await FilmController.getById(id);
19   return new Response(JSON.stringify(body), {
20     status: status,
21     headers: { 'Content-Type': 'application/json' }
22   });
23 });
24
25 filmRouter.post(
26   '/films',
27   validateBody(FilmSchema),
28   async (c): Promise<Response> => {
29

```

```

1 import { Hono } from 'hono';
2 import { db } from '../src/db/index';
3 import { actors } from '../src/db/schema';
4
5 const app = new Hono();
6
7 // Ruta GET /actors
8 app.get('/actors', async (c) => {
9   try {
10     const result = await db.select().from(actors).execute();
11     return c.json(result);
12   } catch (error) {
13     console.error('Database error:', error);
14     return c.json({ error: 'Error al obtener actores' }, 500);
15   }
16 });
17
18 // Ruta GET /films
19 app.get('/films', async (c) => {
20   try {
21     const result = await db.select().from(films).execute();
22     return c.json(result);
23   } catch (error) {
24     console.error('Database error:', error);
25     return c.json({ error: 'Error al obtener películas' }, 500);
26   }
27 });
28
29 n > app.ts

```

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

```

C:\Windows\System32\cmd.e X + v
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

```

localhost:3000/actors

Aplicaciones | Maps | YouTube

Dar formato al texto ☒

```

{
  "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/>

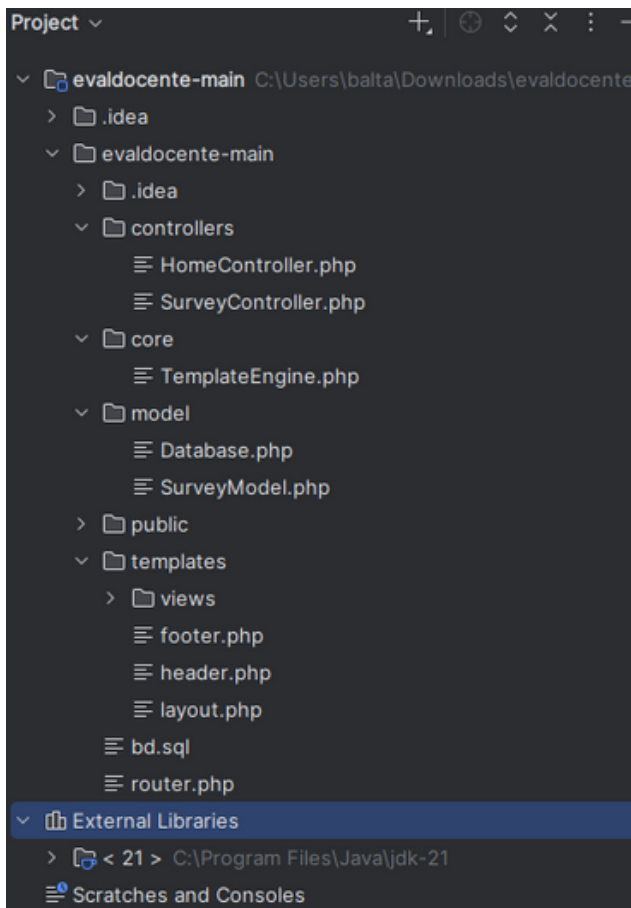
VS17 x64 Thread Safe (2025-May-06 14:18:08)

- [Zip](#) [32.37MB]
sha256: b2cce3ca2ae599ef1b13ae0b84b17aa25f71a94e3098d52c7407238375c1056a
- [Debug Pack](#) [37.13MB]
sha256: 7d4d38328e9766a22548602696dc82f7f76e79ffe32d3570c5fb067fba621ab
- [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:


```

1 <?php
2
3 namespace core;
4
5 class TemplateEngine
6 {
7     public static function render($view, $data = [])
8     {
9         extract($data);
10        ob_start();
11        include __DIR__ . "../templates/views/$view.php";
12        $content = ob_get_clean();
13        include __DIR__ . "../templates/layout.php";
14    }
15
16    public static function renderPartial($view, $data = [])
17    {
18        extract($data);
19        include __DIR__ . "../templates/views/$view.php";
20    }
21 }

```

Motor de plantillas para renderizar vistas.

Conexión a maria db

```

1 <?php
2
3 namespace model;
4
5 class Database {
6     private static $conn;
7
8     private function __construct() {}
9
10    public static function getConnection() {
11        if (self::$conn == null) {
12            self::$conn = new \mysqli("127.0.0.1", "root", "123987", "sample", 3306);
13            if (self::$conn->connect_error) {
14                throw new \Exception("Error de conexión: " . self::$conn->connect_error);
15            }
16        }
17    }
18    return self::$conn;
19 }
20
21

```

```

1 <?php
2
3 namespace model;
4
5 use http\Params;
6
7 class SurveyModel {
8     private $conn;
9
10    public function __construct($conn) {
11        $this->conn = $conn;
12    }
13
14    public function save($docente, $p1, $p2, $p3, $comentarios) {
15        $stmt = $this->conn->prepare(
16            "INSERT INTO respuestas (docente, pregunta1, pregunta2, pregunta3, comentarios)
17            VALUES (?, ?, ?, ?, ?)"
18        );
19
20        $stmt->bind_param("siiiis", $docente, $p1, $p2, $p3, $comentarios);
21
22        if ($stmt->execute()) {
23            return [true, "Gracias por enviar tu evaluación."];
24        } else {
25            return [false, "Error al guardar los datos: " . $stmt->error];
26        }
27    }
28 }

```



```

1 |<?php
2
3 |<require_once dirname(__DIR__) . '/core/TemplateEngine.php';
4 |<const BASE_URL = '/';
5
6 |<spl_autoload_register(function ($class) {
7 |<    $baseDir = dirname(__DIR__);
8 |<    $path = $baseDir . '/' . str_replace('\\', '/', $class) . '.php';
9
10 |<    if (file_exists($path)) {
11 |<        require_once $path;
12 |<    }
13 |<});
14
15 |<// Obtener la ruta
16 |<$uri = trim(parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH), '/');
17 |<$parts = explode('/', $uri);
18
19 |<// Asignar controlador y método por defecto
20 |<$controller = !empty($parts[0]) ? ucfirst($parts[0]) . 'Controller' : 'HomeController';
21 |<$method = $parts[1] ?? 'index';
22
23 |<// Namespace completo del controlador
24 |<$controllerFQN = "controllers\\$controller";
25
26 |<// Ejecutar
27 |<if (class_exists($controllerFQN) && method_exists($controllerFQN, $method)) {
28 |<    call_user_func([$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

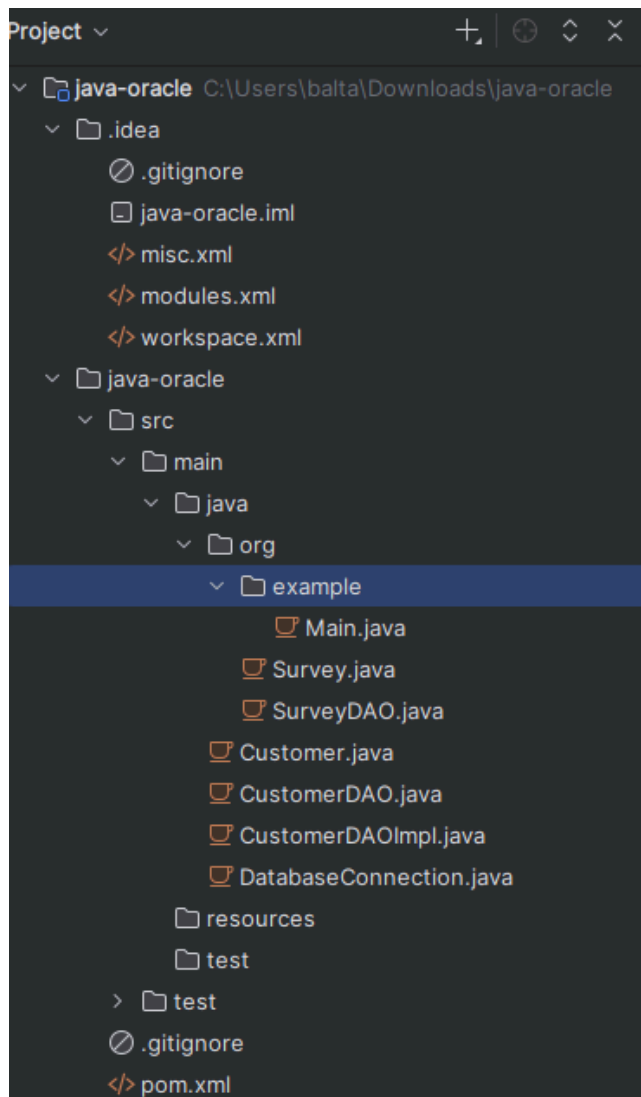
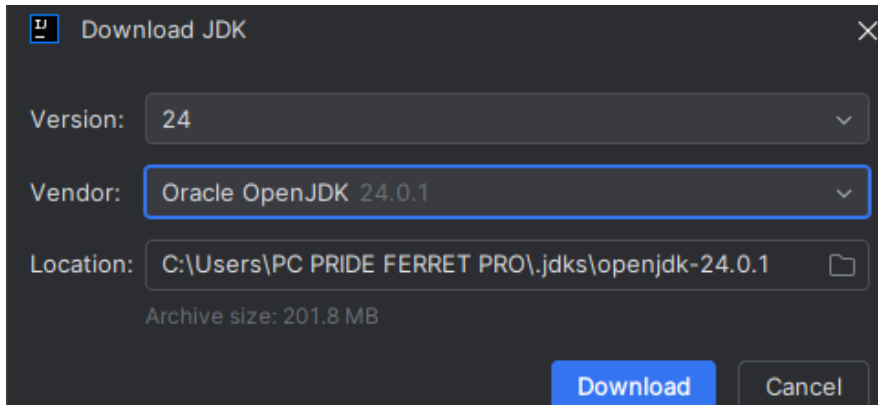
```

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



JAVA ORACLE

primero instale el jdk



Estructura del proyecto:

```

Main.java x Survey.java SurveyDAO.java
1 package org.example;
2
3 import java.util.Scanner;
4
5 public class Main {
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         SurveyDAO dao = new SurveyDAO();
9
10        while (true) {
11            System.out.println("\n--- MENÚ DE ENCUESTAS ---");
12            System.out.println("1. Agregar encuesta");
13            System.out.println("2. Listar encuestas");
14            System.out.println("3. Actualizar encuesta");
15            System.out.println("4. Eliminar encuesta");
16            System.out.println("5. Salir");
17            System.out.print("Seleccione una opción: ");
18            int opcion = sc.nextInt();
19            sc.nextLine(); // limpiar buffer
20
21            switch (opcion) {
22                case 1:
23                    System.out.print("Nombre: ");
24                    String nombre = sc.nextLine();
25                    System.out.print("Edad: ");
26                    int edad = sc.nextInt(); sc.nextLine();
27                    System.out.print("Respuesta (Si / No / Tal vez): ");
28                    String respuesta = sc.nextLine();

```

```

Main.java Survey.java Customer.java x
1 public record Customer(
2     Long customerId,
3     String emailAddress,
4     String fullName) {
5 }

```

**Representa la tabla customers
en la base de datos**

```

Main.java x Survey.java Customer.java CustomerDAO.java x
1 > import ...
3
4 public interface CustomerDAO {
5     void add(Customer customer) throws SQLException;
6     Customer getById(Long id) throws SQLException;
7     List<Customer> getAll() throws SQLException;
8     void update(Customer customer) throws SQLException;
9     void delete(Long id) throws SQLException;
10 }

```

Define las operaciones CRUD:

```

Main.java Survey.java Customer.java CustomerDAO.java CustomerDAOImpl.java x
1  > import ...
4
5  public class CustomerDAOImpl implements CustomerDAO {
6      private Connection connection;
7  >  public CustomerDAOImpl(Connection connection) { this.connection = connection; }
8
9      public void add(Customer customer) throws SQLException {
10         String sql = "INSERT INTO customers (customer_id, email_address, full_name) VALUES (" +
11             customer.getId() + ", " + customer.getEmailAddress() + ", " + customer.getFullName() + ")";
12         try (PreparedStatement stmt = connection.prepareStatement(sql)) {
13             stmt.setLong(1, customer.getId());
14             stmt.setString(2, customer.getEmailAddress());
15             stmt.setString(3, customer.getFullName());
16             stmt.executeUpdate();
17         }
18     }
19
20     public Customer getById(Long customerId) throws SQLException {
21         String sql = "SELECT * FROM customers WHERE customer_id = ?";
22         try (PreparedStatement stmt = connection.prepareStatement(sql)) {
23             stmt.setLong(1, customerId);
24             ResultSet rs = stmt.executeQuery();
25             if (rs.next()) {
26                 return new Customer(
27                     rs.getLong("customer_id"),
28                     rs.getString("email_address"),
29                     rs.getString("full_name")
30                 );
31             }
32         }
33     }
34 }

```

Usa JDBC para ejecutar consultas:

```

Customer.java CustomerDAO.java CustomerDAOImpl.java DatabaseConnection.java x
1  > import ...
4
5  public class DatabaseConnection {
6      private static final String URL = "jdbc:oracle:thin:@//172.17.0.2:1521/XE";
7      private static final String USER = "root";
8      private static final String PASSWORD = "root";
9
10     public static Connection getConnection() throws SQLException {
11         return DriverManager.getConnection(URL, USER, PASSWORD);
12     }
13 }

```

Maneja la conexión a la base de datos