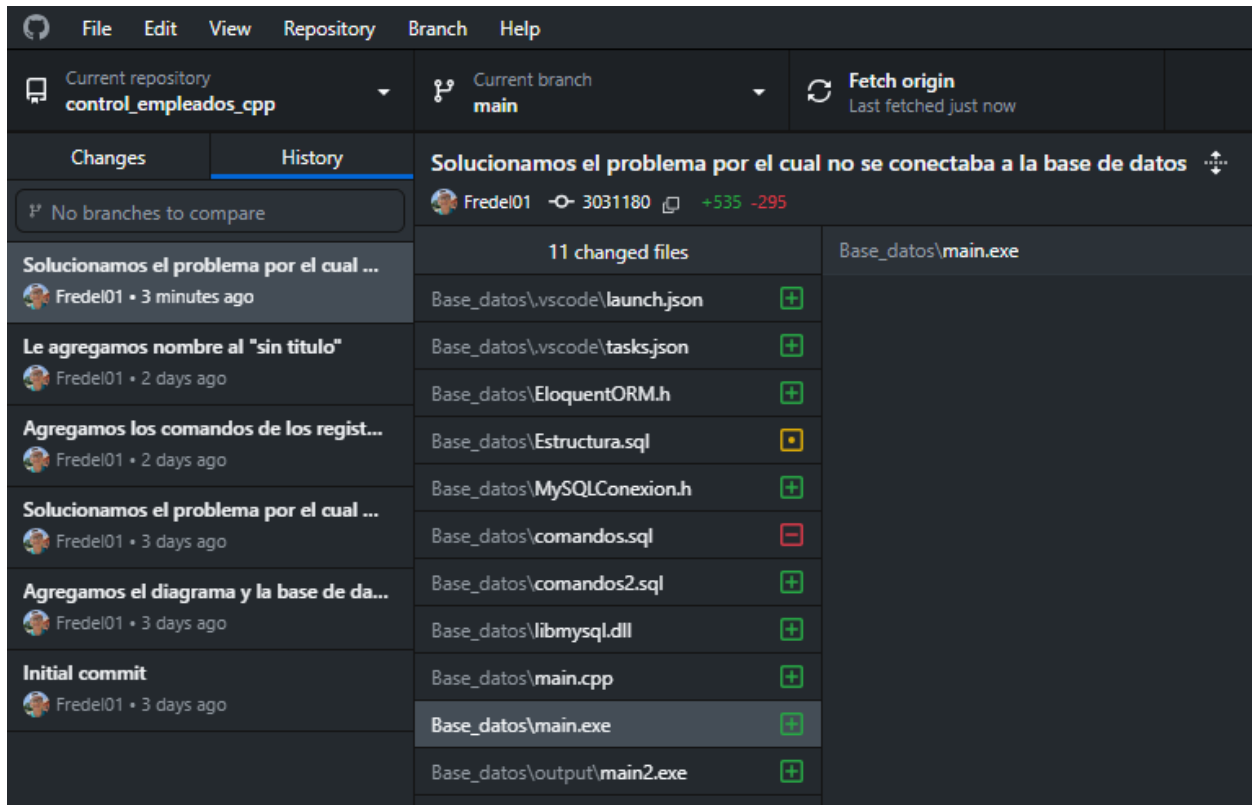


1. Creamos un repositorio local el cual llamamos “control_empleados” con la carpeta “Base_datos en donde desde visual hacíamos los commits



2. Creamos la base de datos desde MySQL server.

```
mysql> show databases;
```

Database
control_horario
empleado
information_schema
mysql
performance_schema
sys

```
6 rows in set (0.01 sec)
```

```
mysql> use control_horario;
```

Database changed

```
mysql> show tables;
```

Tables_in_control_horario
empleados
personas
registros

```
3 rows in set (0.01 sec)
```

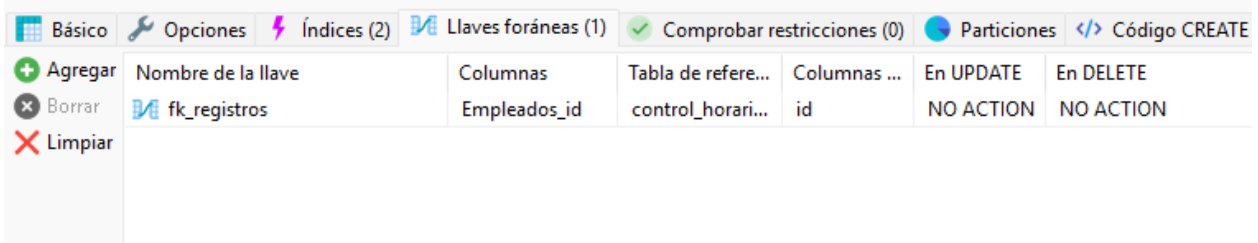
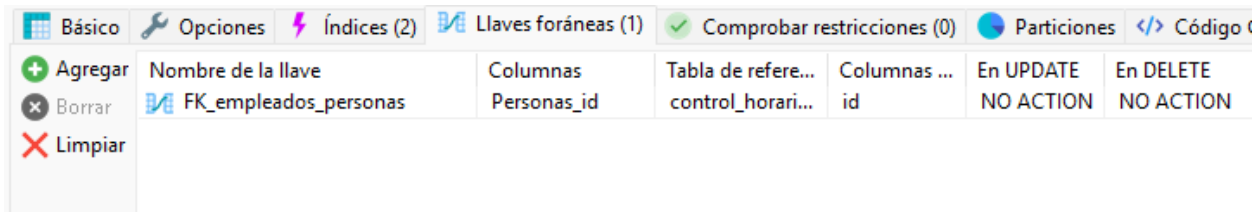
```
mysql> describe personas;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
nombre	varchar(100)	YES		NULL	
edad	int	YES		NULL	
genero	varchar(10)	YES		NULL	

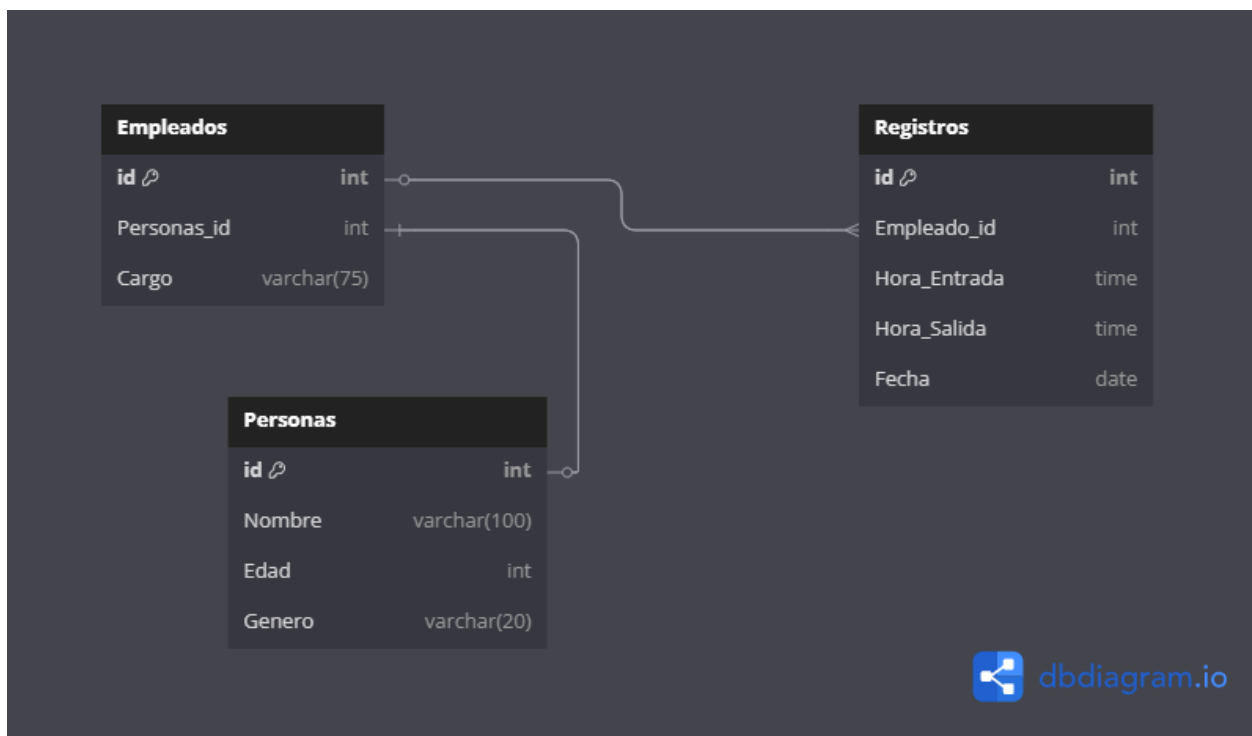
```
4 rows in set (0.01 sec)
```

3. Relacionamos las tablas en HeidiSQL desde el modo gráfico

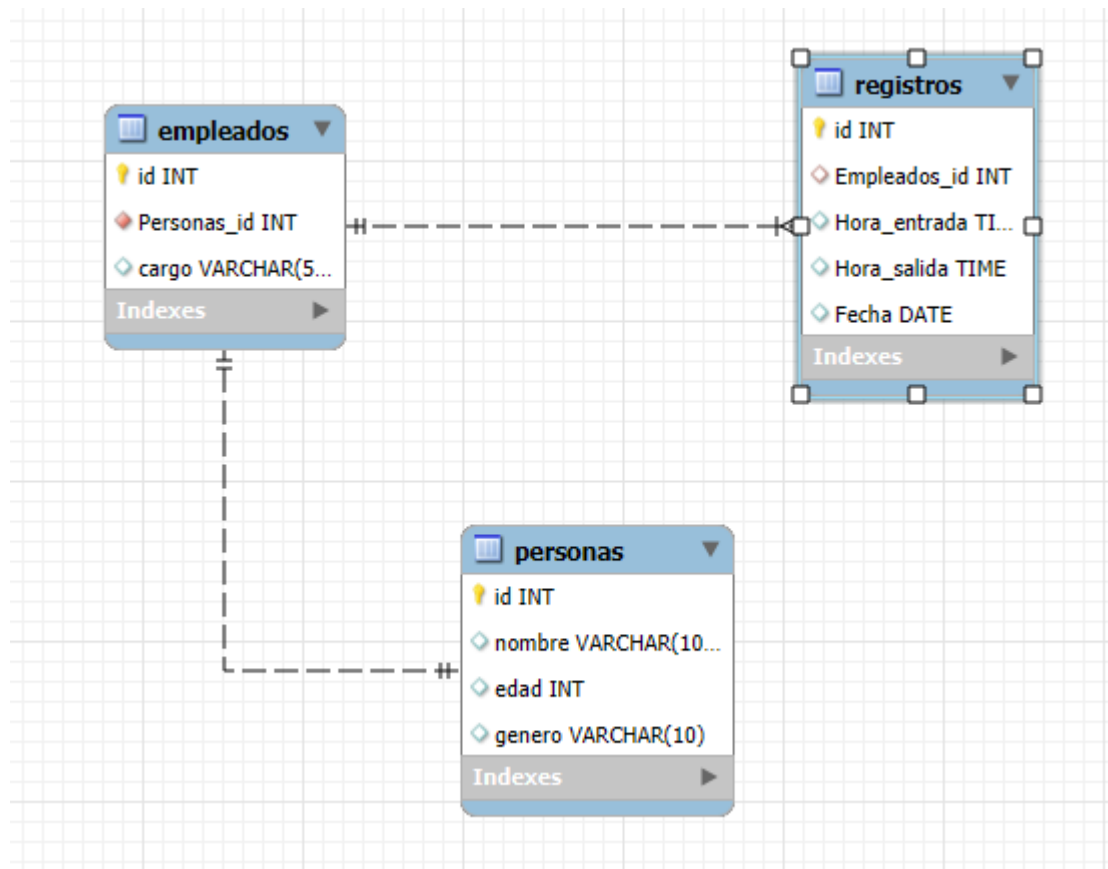
- Seleccionaba Llaves Foraneas> Nueva y desde ahí seleccionaba a que tabla y columna se iba a relacionar la llave foranea



4. Cree el diagrama relación entidad en dbdiagram.io



5. Revise el diagrama de la estructura de nuestra base de datos en WorkBench y lo compare con el diagrama creado



6. Ingresamos los datos en heidisql

control_horario.personas: 9 filas en total (exacto)

#	id	nombre	edad	genero
1	1	Guadalupe Orellana	67	Femenino
2	2	Leopoldo Moran	72	Masculino
3	3	Irlanda Arrollo	71	Femenino
4	4	Fredy Avila	45	Masculino
5	5	Rosivel Ramos	41	Femenino
6	6	Rebeka Moran	12	Femenino
7	11	Fredy Moran	18	Masculino
8	22	Mario Cortez	21	No binario
9	33	Jimmy Alvarado	22	Masculino

control_horario.empleados: 9 filas en total (exacto)

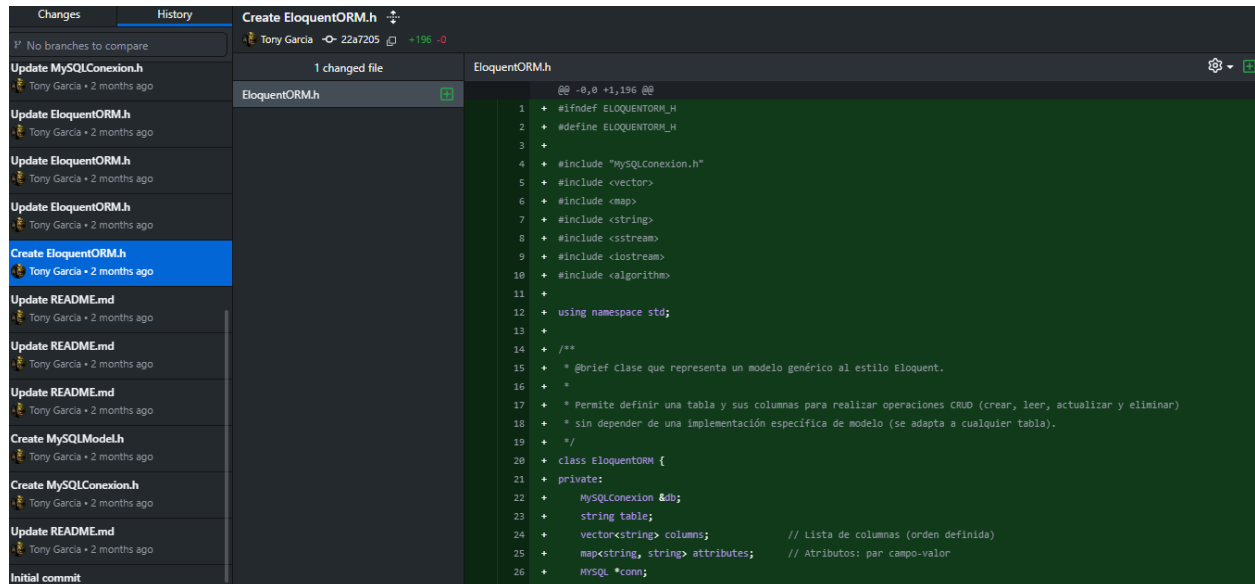
#	id	Personas_id	cargo
1	8	22	Repartidor
2	11	11	Repartidor
3	1	5	Limpieza
4	6	6	Limpieza
5	2	1	Inspector
6	3	4	Cocinero
7	7	33	Cobrador
8	5	5	Chef
9	4	3	Camarero

control_horario.registros: 9 filas en total (exacto)

#	id	Empleados_id	Hora_entrada	Hora_salida	Fecha
1	55	2	12:00:00	20:00:00	2025-04-06
2	44	1	10:00:00	18:00:00	2025-04-07
3	11	11	09:00:00	17:00:00	2025-04-08
4	22	4	09:00:00	17:00:00	2025-01-09
5	33	3	09:00:00	17:00:00	2025-03-16
6	66	6	09:00:00	18:00:00	2025-04-09
7	77	11	09:00:00	18:00:00	2025-04-07
8	99	7	09:00:00	18:00:00	2025-04-08
9	88	5	08:00:00	17:00:00	2025-04-07

7. Luego nos fuimos a visual a configurar los requisitos necesarios para conectarnos a la base de datos

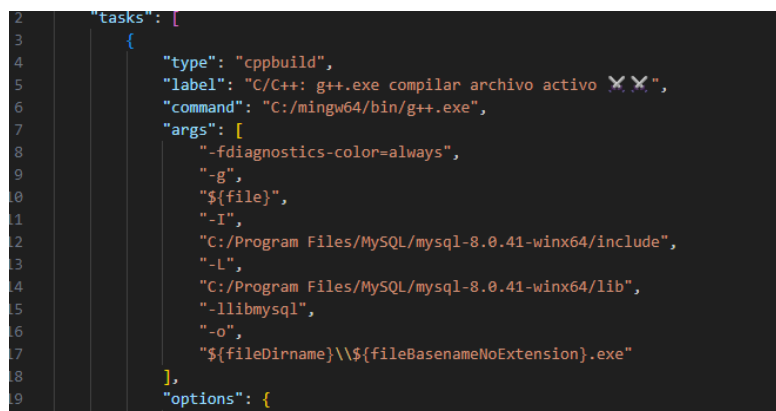
- Primero buscamos las librerías necesarias para la conexión en el repositorio del ingeniero y lo clonamos



The screenshot shows a code editor interface. On the left, a 'Changes' panel displays a list of commits by 'Tony Garcia' from 2 months ago. The commits include 'Update MySQLConexion.h', 'Update EloquentORM.h', 'Create EloquentORM.h', 'Update README.md', 'Create MySQLModel.h', and 'Create MySQLConexion.h'. The 'Create EloquentORM.h' commit is highlighted in blue. The main editor area shows the content of 'EloquentORM.h', which is a C++ header file. It includes guards for multiple definitions, includes for 'MySQLConexion.h', 'vector', 'map', 'string', 'sstream', 'iostream', and 'algorithm'. It uses the 'std' namespace and contains a class definition 'EloquentORM' with private attributes: 'MySQLConexion &db', 'string table', 'vector<string> columns', 'map<string, string> attributes', and 'MySQL *conn'.

```
@@ -0,0 +1,196 @@
1 + #ifndef ELOQUENTORM_H
2 + #define ELOQUENTORM_H
3 +
4 + #include "MySQLConexion.h"
5 + #include <vector>
6 + #include <map>
7 + #include <string>
8 + #include <sstream>
9 + #include <iostream>
10 + #include <algorithm>
11 +
12 + using namespace std;
13 +
14 + /**
15 +  * @brief Clase que representa un modelo genérico al estilo Eloquent.
16 +  *
17 +  * Permite definir una tabla y sus columnas para realizar operaciones CRUD (crear, leer, actualizar y eliminar)
18 +  * sin depender de una implementación específica de modelo (se adapta a cualquier tabla).
19 +  */
20 + class EloquentORM {
21 + private:
22 +     MySQLConexion &db;
23 +     string table;
24 +     vector<string> columns; // Lista de columnas (orden definida)
25 +     map<string, string> attributes; // Atributos: par campo-valor
26 +     MySQL *conn;
```

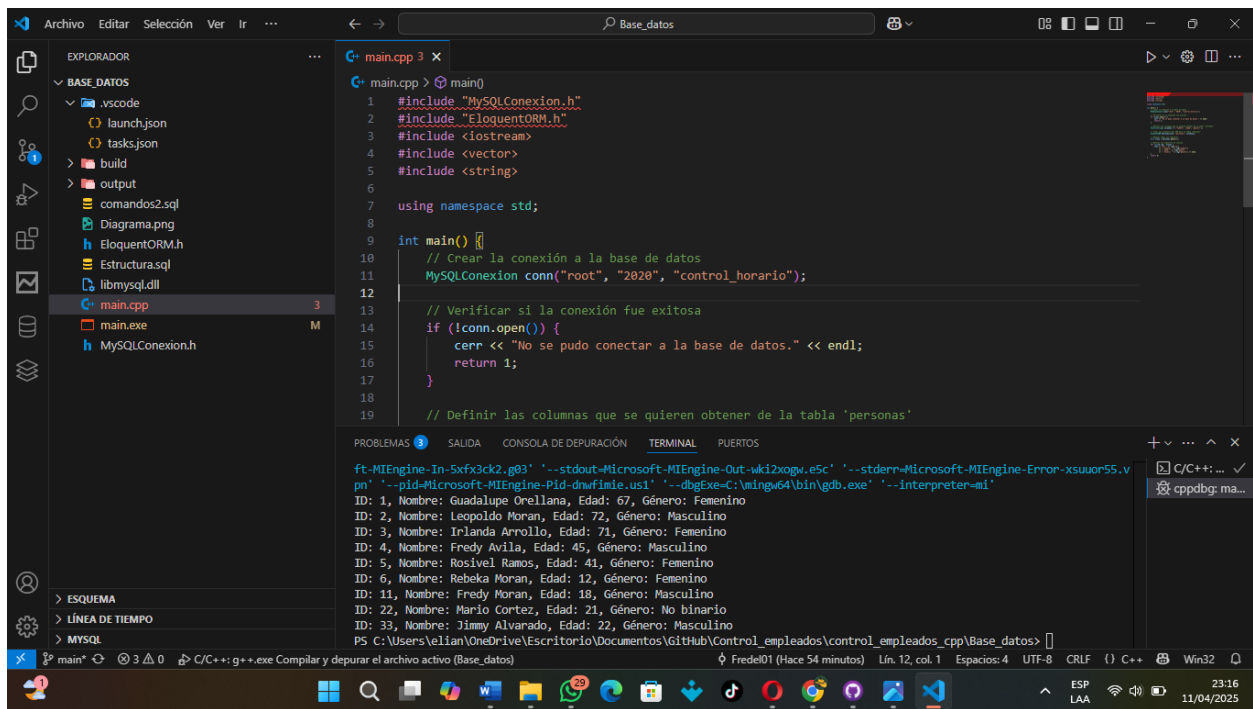
- Luego configuramos el task.json con las direcciones a las librerías necesarias para la conexión



The screenshot shows a 'task.json' file in a code editor. It defines a task for building a C++ project. The 'type' is 'cppbuild', the 'label' is 'C/C++: g++.exe compilar archivo activo', and the 'command' is 'C:/mingw64/bin/g++.exe'. The 'args' array includes flags for diagnostics, debugging, file names, and include/library paths for MySQL. The 'options' object is partially visible.

```
2 "tasks": [
3 {
4     "type": "cppbuild",
5     "label": "C/C++: g++.exe compilar archivo activo",
6     "command": "C:/mingw64/bin/g++.exe",
7     "args": [
8         "-fdiagnostics-color=always",
9         "-g",
10        "${file}",
11        "-I",
12        "C:/Program Files/MySQL/mysql-8.0.41-win64/include",
13        "-L",
14        "C:/Program Files/MySQL/mysql-8.0.41-win64/lib",
15        "-llibmysql",
16        "-o",
17        "${fileDirname}\\${fileBasenameNoExtension}.exe"
18    ],
19    "options": {
```

- Hicimos el main cpp donde realizaríamos la conexión y efectivamente funcionó



Pero para todo esto hubieron muchas luchas, el mysql server 8.0 no estaba agregado al path asi que lo tuvimos que agregar para que finalmente funcionara...

8. Hicimos los commits finales

