**Programación Orientada a Objetos**

**Parcial Final - POO**

Integrantes:

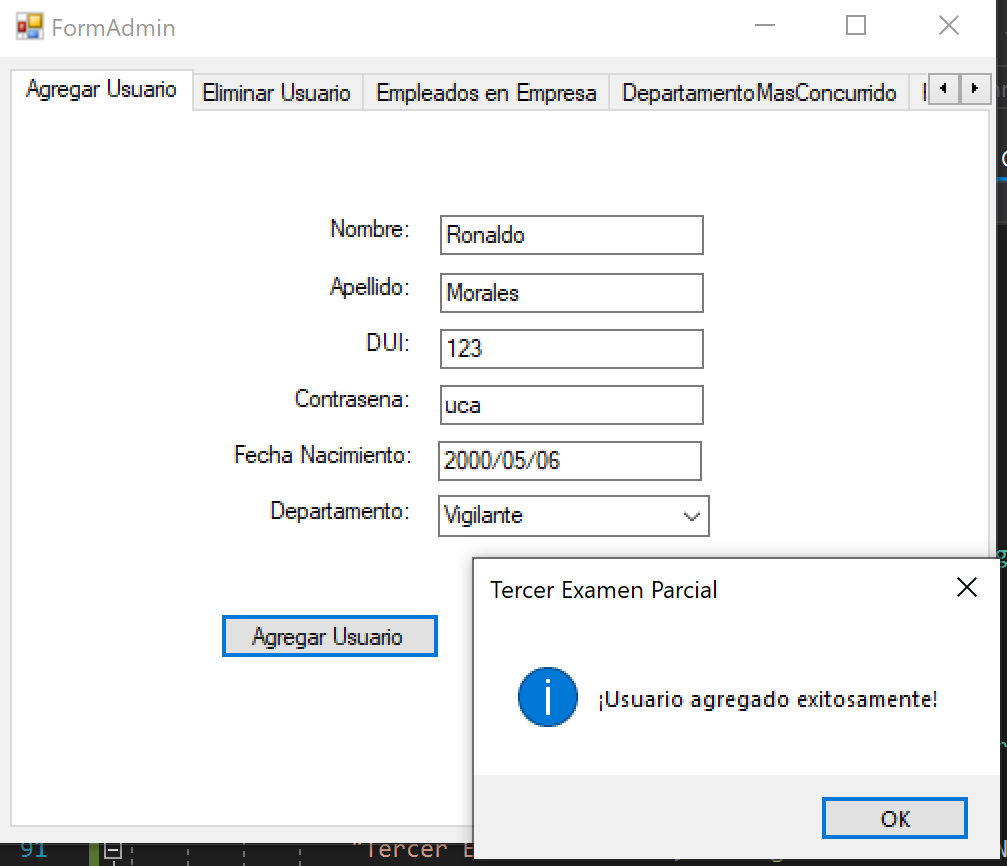
Engelhard Robert, Diego, 00356119

Catedrático: Lic. Ronaldo Canizales

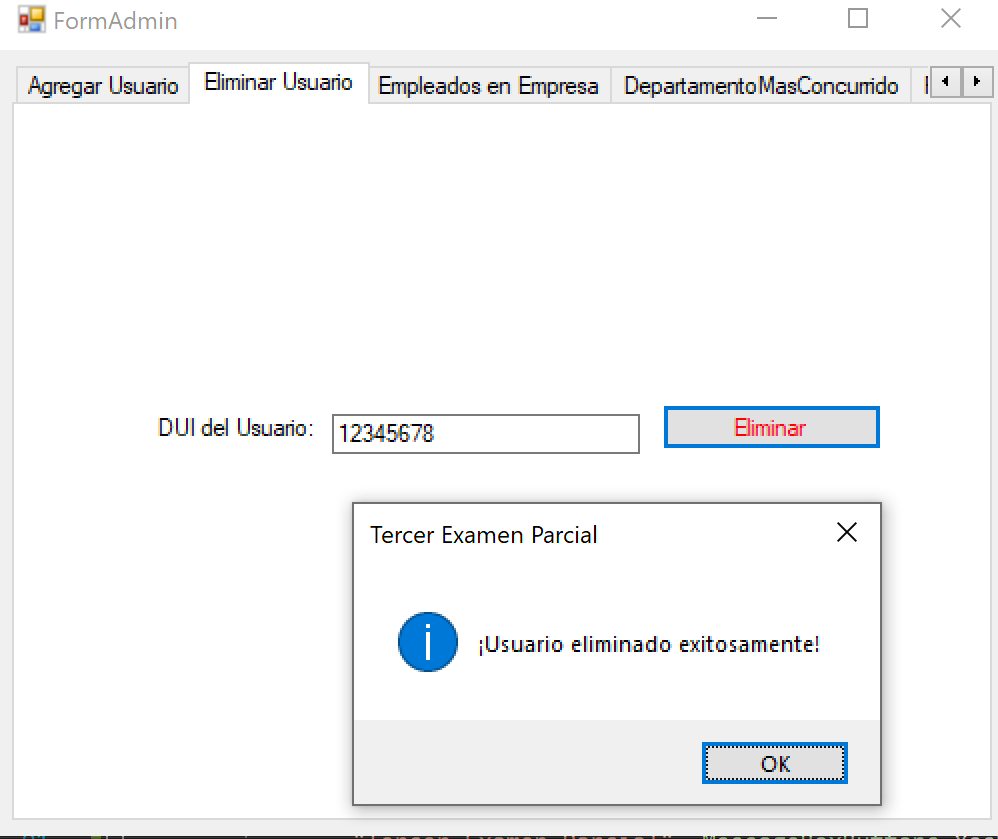
Fecha de entrega: viernes 3 de julio de 2020

**CAPTURAS DE PANTALLA**

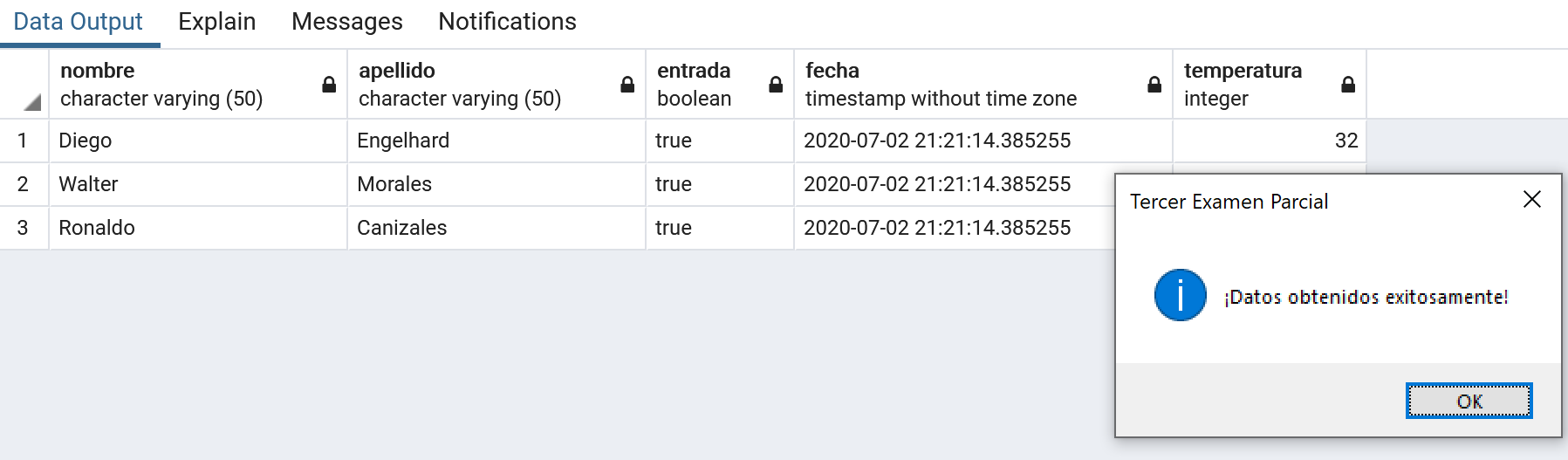
* Agregar Usuario (Admin)



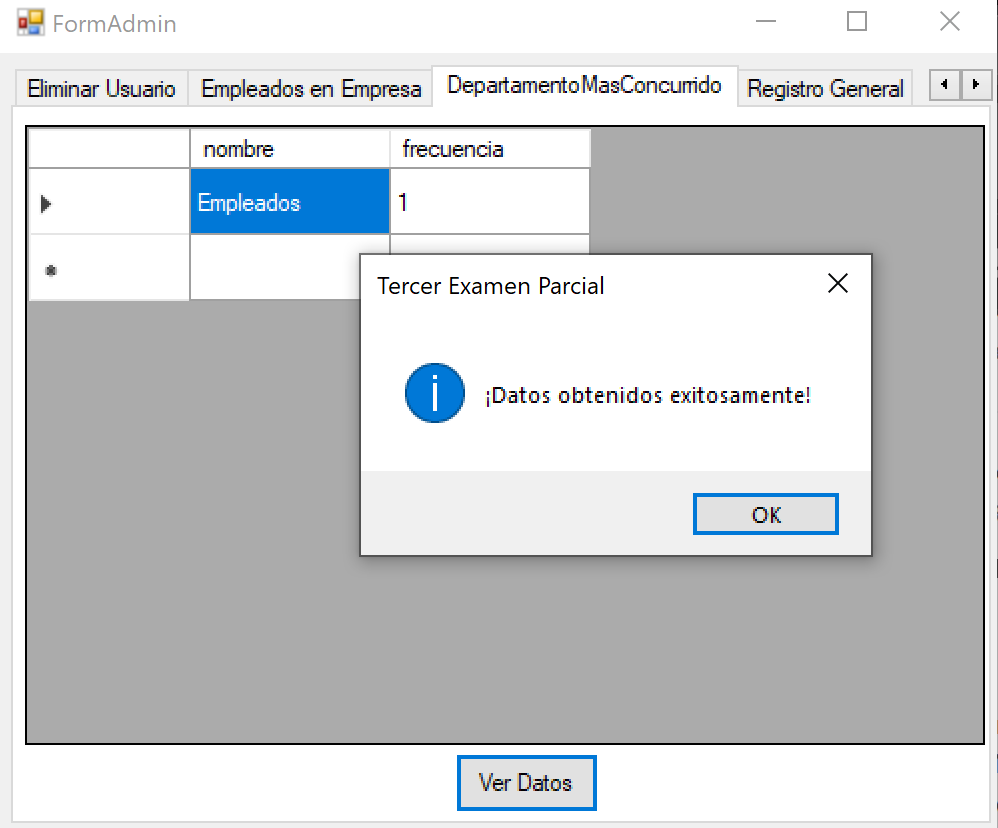
* Eliminar Usuario (Admin)



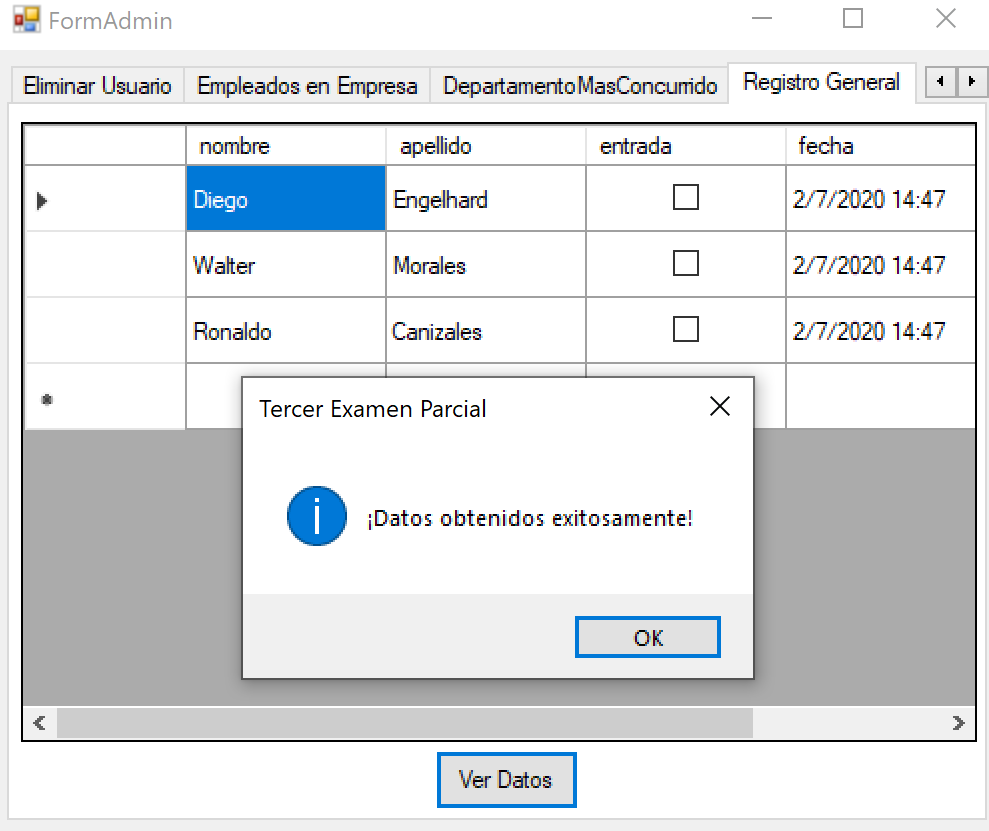
* Empleados actualmente en la empresa (Admin)



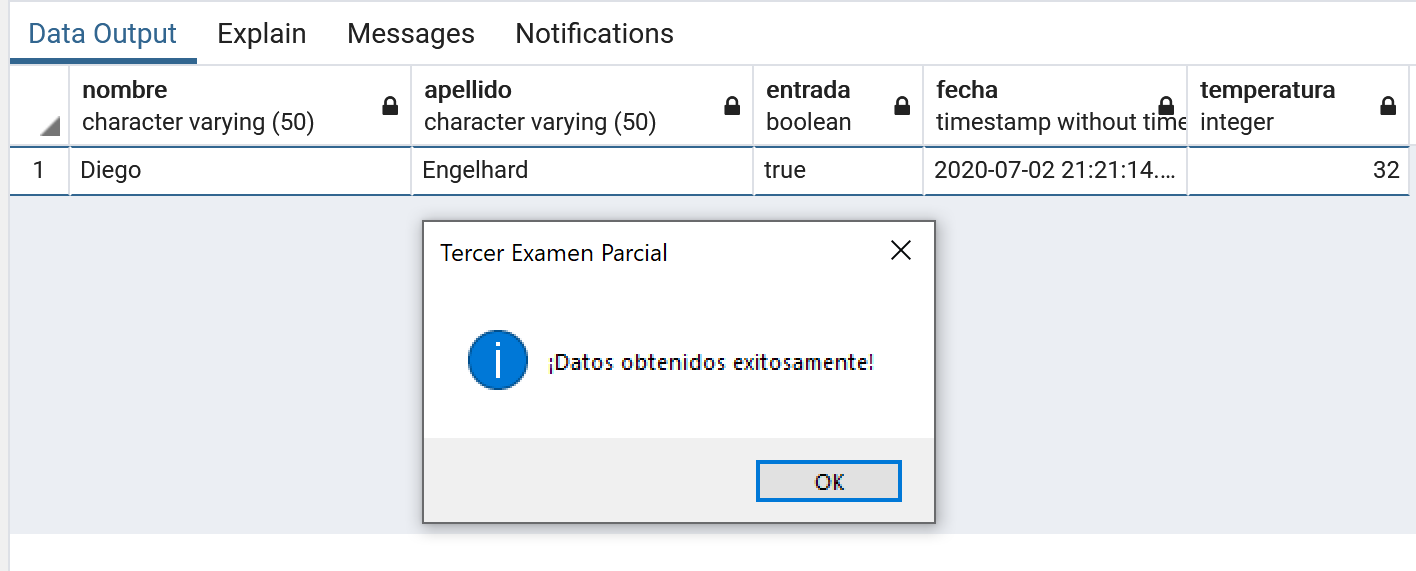
* Departamento mas concurrido (Admin)



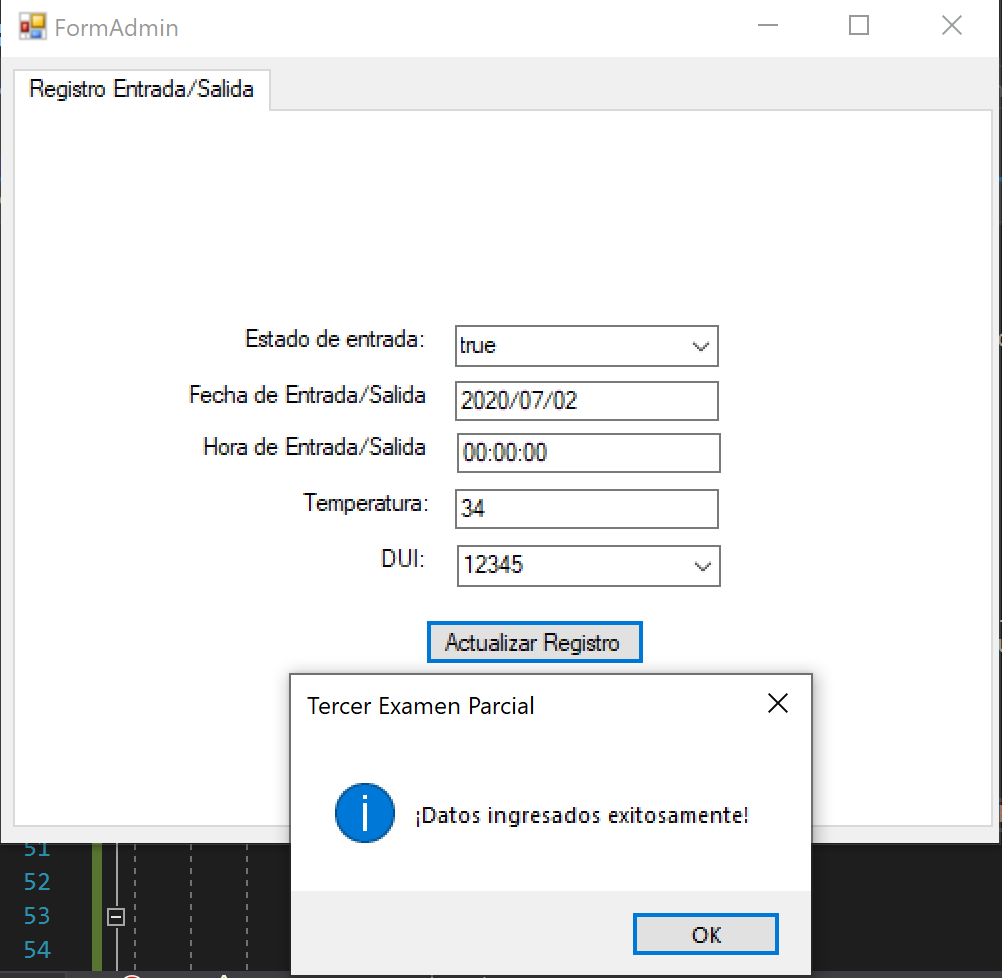
* Registro general de usuarios (Admin)



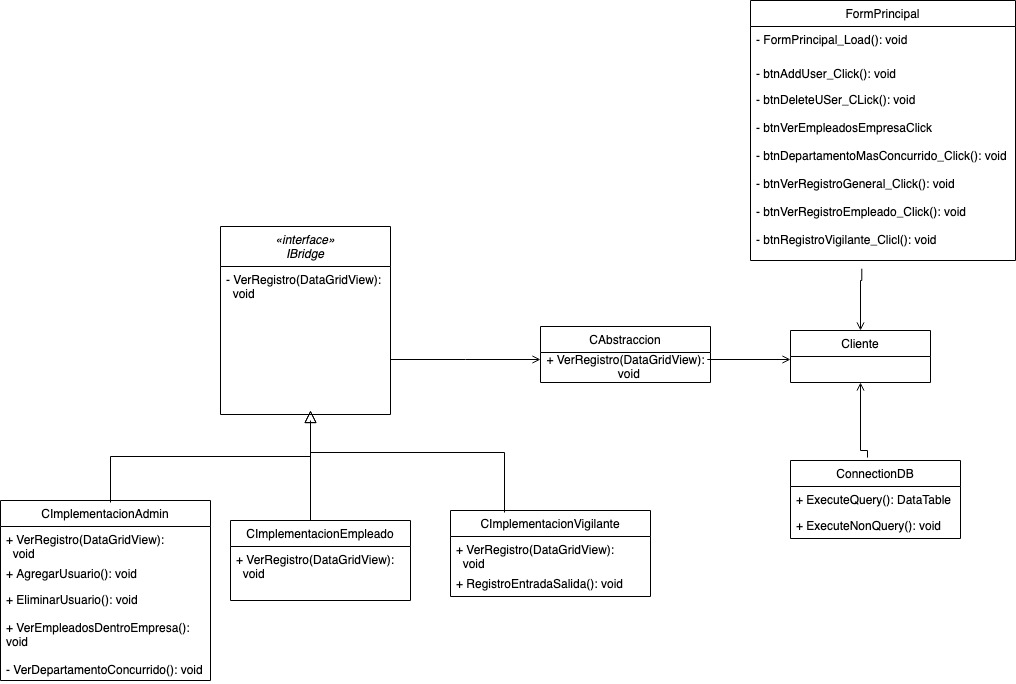
* Registro individual de usuario (Empleado)



* Registro de entrada/salida (Vigilante)



**DIAGRAMA UML**



**EXPLICACION DE USO DE PATRÓN**

Para la elaboracion de este programa se ha ocupado el patron de diseño Bridge. Bridge es extremadamente util cuando dos piezas de software están relacionadas directamente, sin embargo, puede haber una gran probabilidad de que una de las dos partes llegue a cambiar y esto puede conllevar a la necesidad de modificar la pieza que se encuentra al otro lado.

Para resolver este inconveniente, el patron Bridge propone crear una estructura de clases basada en la agregación, es decir, una clase puente es utilizada para desacoplar la clase que quiere utilizar y el cliente, de esta manera que este último no conozca la clase destino, permitiendo que cualquier parte pueda cambiar sin afectar a la otra.

Se ha ocupado el patron Bridge, ya que aporta numerosos beneficios, entre estos estan:

* + Desacopla la implementacion de la abstraccion.
  + Se usa cuando surge una nueva version de software, pero la anterior aun se va a seguir utilizando con el cliente.
  + Protege de cambio al cliente.
  + El cliente indicara con que version desea trabajar
  + Hace que la interfaz de una o mas clases parezcan que tienen la de una clase en particular

El patron birdge se ha implementado de la siguiente forma:

1. Se ha creado una clase abstraccion, la cual
2. Se ha implementado una interfaz. Esta permitira que
3. Se han dado uso a 3 implementaciones, respectivamente. Una, la del administrador, la cual es encargada de los quehaceres del administrador en la empresa. Luego, la del Empleado, encargada de la funcionalidad de las tareas del empleado. Por ultimo la del Vigilante, la cual tiene como objetivo dar registro a las entradas y salidas de los usuarios.

**SCRIPT DE LA BASE DE DATOS**

/\* Creacion de tablas \*/

CREATE TABLE departamento(

departamento\_id SERIAL,

nombre VARCHAR(50),

ubicacion VARCHAR(50),

CONSTRAINT PK\_departamento PRIMARY KEY (departamento\_id)

);

CREATE TABLE usuario(

usuario\_id SERIAL,

nombre VARCHAR(50),

apellido VARCHAR(50),

dui VARCHAR(15),

contrasena VARCHAR(50),

fechaNacimiento DATE,

departamento\_id INT,

CONSTRAINT PK\_usuario PRIMARY KEY (usuario\_id),

CONSTRAINT FK\_usuario\_departamento

FOREIGN KEY (departamento\_id) REFERENCES departamento(departamento\_id)

ON DELETE SET NULL

);

CREATE TABLE registro(

registro\_id SERIAL,

entrada BOOL,

fecha TIMESTAMP,

temperatura INT,

usuario\_id INT,

CONSTRAINT PK\_registro PRIMARY KEY (registro\_id),

CONSTRAINT FK\_registro\_usuario

FOREIGN KEY (usuario\_id) REFERENCES usuario(usuario\_id)

ON DELETE SET NULL

);

/\* Para agregar un usuario \*/

INSERT INTO usuario(nombre, apellido, dui, contrasena, fechaNacimiento, departamento\_id)

VAlUES('Walter', 'Morales', '098765', 'uca', '1997/01/01', 2);

/\* Para eliminar un usuario por DUI \*/

DELETE FROM usuario

WHERE dui = 12345678;

/\* Para ver el historial de un empleado \*/

SELECT us.nombre, re.entrada, re.fecha, re.temperatura

FROM registro re, usuario us

WHERE re.usuario\_id = us.usuario\_id

/\* Para ver el historial de todos los empleados \*/

SELECT us.nombre, re.entrada, re.fecha, re.temperatura

FROM registro re, usuario us

/\* Ver historial de empleados dentro de la empresa \*/

SELECT us.nombre, re.entrada, re.fecha, re.temperatura

FROM registro re, usuario us

WHERE re.entrada = true

/\* Para ver el historial de todos los empleados \*/

SELECT us.nombre, re.entrada, re.fecha, re.temperatura

FROM registro re, usuario us

/\* Ver departanmento mas concurrido \*/

SELECT d.nombre, count(u.departamento\_id) as frecuencia

FROM REGISTRO r, DEPARTAMENTO d, USUARIO u

WHERE r.usuario\_id = u.usuario\_id AND d.departamento\_id = u.departamento\_id

GROUP BY d.departamento\_id

ORDER BY frecuencia DESC LIMIT 1;

/\* Para actualizar el registro de un empleado \*/

UPDATE registro

SET entrada = false,

fecha = LOCALTIMESTAMP,

temperatura = 30

WHERE usuario\_id = 1;

/\* Query de usuario y su contrasena coincidan \*/

SELECT count(\*)

FROM usuario

WHERE dui = '12345678' AND contrasena = 'uca'