INSTITUTO POLITECNICO NACIONAL

Escuela Superior de Cómputo

ESCOM

UA: Programación orientada a objetos.

Profesor: Tecla Parra Roberto

Grupo: 2CM1

Proyecto final

Opción: Aplicador y evaluador de exámenes de opción múltiple

Alumnos:

Ríos Alonso Juan José

Ruiz González Ian Alexander

Link video: https://youtu.be/QrUbagA1c94

Fecha de entrega: 22 de enero de 2021

Introduccion

En la entrega de este proyecto final haremos uso de todas las habilidades adquiridas durante el curso de **programación orientada a objetos** para desarrollar una aplicación con una interfaz gráfica, conexión a base de datos, uso de sentencias de SQL y distintas librerías.

Objetivos

-Debera contar con una Interfaz grafica de usuario y en dicha interfaz un Acerca de...

-Para cadapregunta habra 4 posibles respuestas el usuario seleccionara la suya mediante un RadioButton.

-Una vez que el usuario seleccione su respuesta se le presentara la siguiente pregunta junto con sus 4 posibles respuestas.

-Cuando el usuario termine el examen se le evaluara y se le mostrara dicha evaluacion.

-Definir una clase Reactivo con los siguientes atributos:

> **Pregunta, opcionA, opcionB, opcionC, opcionD, respuesta**

-Almacenar las 10 instancias de reactivo en un vector

-Que limite el tiempo (usando un timer) que tiene el usuario para contestar cada pregunta

-Que obtenga las 10 preguntas de una base de datos (usando JDBC).

-Que permita (a un administrador) agregar nuevas preguntas a la base de datos.

-Que elija 10 reactivos de forma aleatoria de un banco de 50 reactivos.

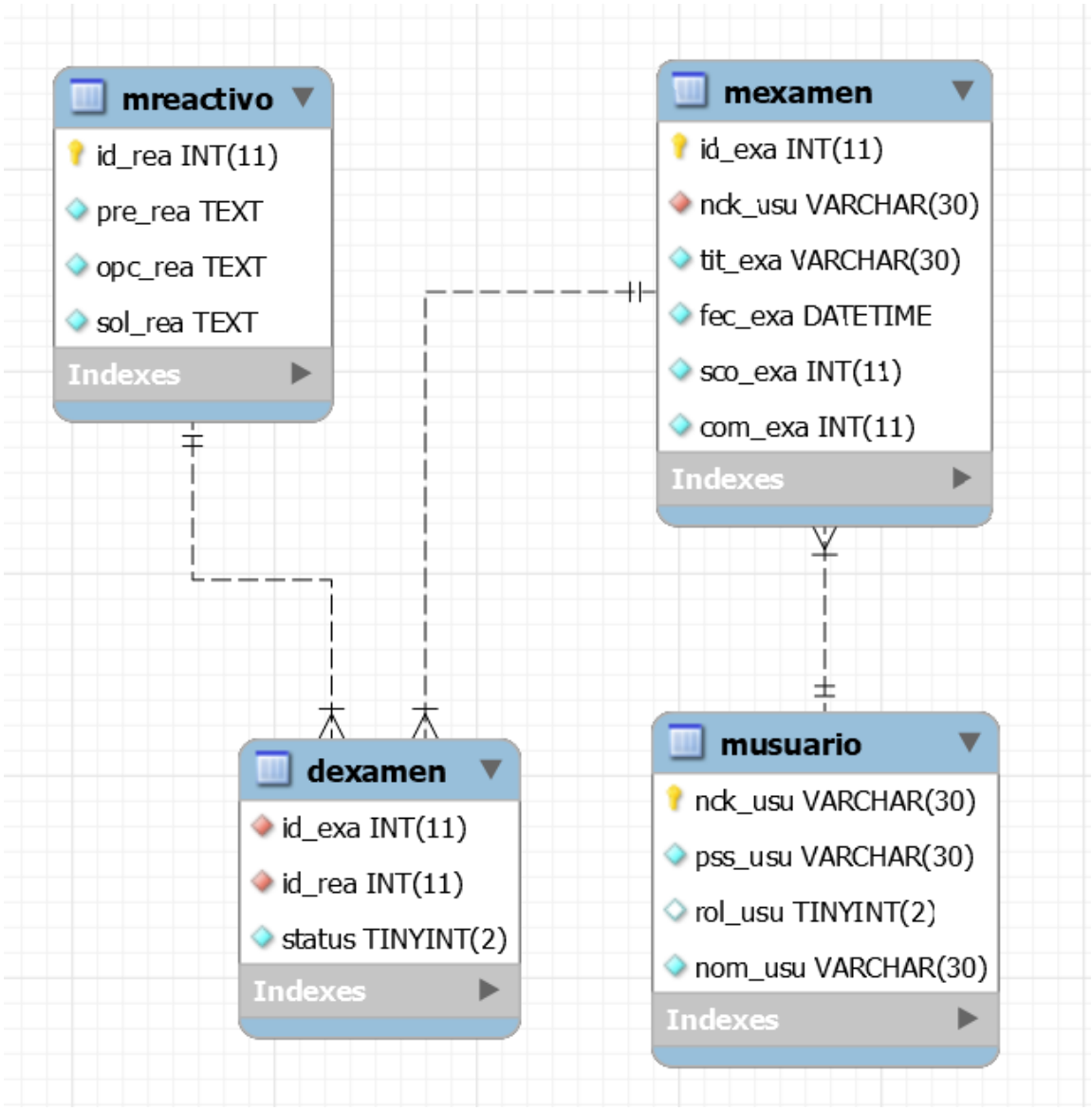
-Que permita al usuario abandonar el examen en culquier momento y que luego le permita retomar el examen donde se quedo la ultima vez.

-Ademas de la clase Reactivo definir una clase Examen con los siguientes atributos:

> **Titulo del examen, nombre de quien presenta el examen, fecha, ultima pregunta respondida, y calificacion.**

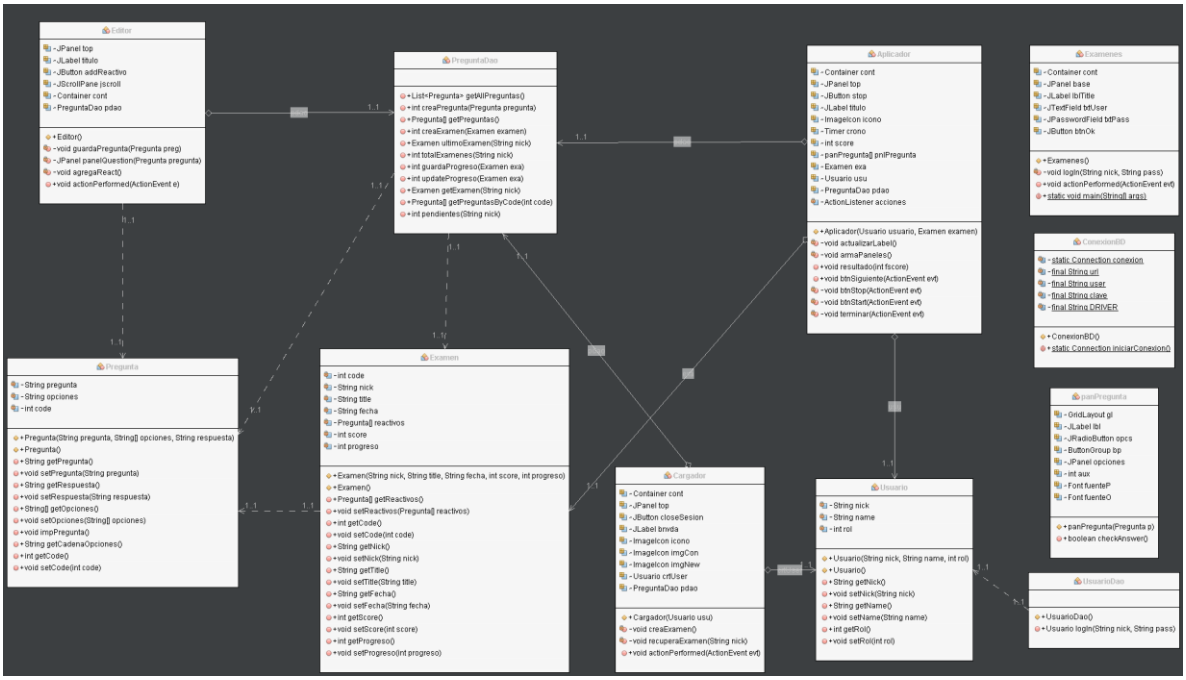
Desarrollo

Primero construimos la base de datos con ayuda del **Workbench** de **MySQL** y creamos el modelo entidad relacion para las tablas.



mreactivo: Almacena el id del reactivo, la pregunta, las opciones de respuesta y la solucion.
dexamen: Almacena el id del examen, el id del reactivo y el status para saber si esta contestada o no.
mexamen: Almacena el id del examen, el usuario que lo esta haciendo, el titulo del examen, la fecha, el score del examen y las preguntas que faltan.
musuario: Almacena el tipo de usuario que esta usando la plataforma en ese momento (si es administrador o alumno), sus contraseñas, rol de usuario y su nombre de usuario.

A continuacion muestro el diagrama de clases del programa para poder explicar las partes mas importantes del mismo.



Creamos la clase encargada de la conexión con la base de datos examen en MySQL. Una vez que todo se realizo correctamente retornamos el objeto Connection.

```
public class ConexionBD {
    private static Connection conexion;
    private static final String url = "jdbc:mysql://localhost:3306/examen_ma
te"; //Nombre de la DB
    private static final String user = "root"; //Usuario
    private static final String clave = "1234"; //Contrasenia
    private static final String DRIVER = "com.mysql.jdbc.Driver";
    public ConexionBD () {}
    public static Connection iniciarConexion() throws SQLException {
        try{
            Class.forName(DRIVER);
            conexion = DriverManager.getConnection(url, user, clave);
            System.out.println("Conectado a la base de datos...");
        }catch(Exception ex) {
            System.out.println("Error al conectar, detalles "+ex.getMessage(
));
        }
        return conexion;
    }
}
```

Diseñamos la clase pregunta, la cual nos facilitara el manejo de la obtención de datos de la BD.

```
    public Pregunta(String pregunta, String[] opciones, String respuesta) {
        this.pregunta = pregunta;
        this.respuesta = respuesta;
        for (int i = 0; i < opciones.length; i++)
            this.opciones[i] = opciones[i];
    }
```

Después, implementamos un método en otra clase. Este método obtendrá todas las preguntas de una base de datos y se auxilia de otros dos métodos, uno para obtener los reactivos y otro para obtener la respuesta

```
public Pregunta[] getPreguntas() {
        Pregunta[] p = new Pregunta[10];
        int i = 0;
        try{
            ResultSet rs = null;
            Connection con = ConexionBD.iniciarConexion();
            String sql = "SELECT * FROM mreactivo ORDER BY rand() limit 10;"
;
            PreparedStatement ps = con.prepareStatement(sql);
            rs = ps.executeQuery();
            while(rs.next()) {
                p[i] = new Pregunta();
                p[i].setPregunta(rs.getString("pre_rea"));
                String[] opciones = rs.getString("opc_rea").split(",");
                p[i].setOpciones(opciones);
                p[i].setRespuesta(rs.getString("sol_rea"));
                p[i].setCode(rs.getInt("id_rea"));
                i++;
            }
            con.close();
        }catch(Exception ex) {
            System.out.println("Error en getPreguntas "+ex.getMessage());
        }
        return p;
    }
```

En este punto puede decir que hay varias funciones claves dentro del proyecto por lo cual hare mencion de las que me parecen más importantes de aquí en adelante

Funcion que crea un examen en el cual agrega en nick de la persona que lo esta resolviendo, obtiene las preguntas de la base y almacena el tiempo fecha y hora del ultimo examen contestado.

```
private void creaExamen() {
        Examen exa = new Examen();
        exa.setTitle("Matematicas 1-
"+pdao.totalExamenes(crtUser.getNick()));
        exa.setNick(crtUser.getNick());

        if (pdao.creaExamen(exa) > 0) {
            exa = pdao.ultimoExamen(crtUser.getNick());
            Pregunta[] pack = pdao.getPreguntas();
            exa.setReactivos(pack);
            this.setVisible(false);
            new Aplicador(crtUser, exa);
        }

    }
```

La parte del aplicador donde se verifican y almacenan las respuestas despues de dar click en el boton de siguiente a la hora de pasar a la siguiente pregunta verificando si esta el correcta o no.

```
public void btnSiguiente(ActionEvent evt) {
        crtPanel ++;
        if (crtPanel < 10) {
            crono.restart();
            time.setText("90");
            pnlPregunta[crtPanel-1].setVisible(false);
            score += (pnlPregunta[crtPanel-1].checkAnswer())? 10:0;
            body.add(pnlPregunta[crtPanel]);
            this.validate();
        } else {
            crono.stop();
            time.setText("Se acabó");
            pnlPregunta[crtPanel-1].setVisible(false);
            resultado(score);
            bottom.setVisible(false);
        }


    }
```

La parte donde se generan las sentencias de SQl para las consultas durante todos los procesos que es la clase **PreguntaDao.java** y casi todo su contenido.
Ya que en este apartado se crean examenes, se recuperan examenes, preguntas, guarda progreso, actualiza y obtiene el examen que se estaba resolviendo.

```java
 public int creaExamen(Examen examen) {
        int status = 0;

        try{
            ResultSet rs = null;
            Connection con = ConexionBD.iniciarConexion();
            String sql = "INSERT INTO mexamen(nck_usu, tit_exa) values (?, ?
)";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, examen.getNick());
            ps.setString(2, examen.getTitle());
            status = ps.executeUpdate();
            con.close();
        }catch(Exception ex) {
            System.out.println("Error en creaExamen "+ex.getMessage());
        }

        return status;
    }

    public Examen ultimoExamen(String nick) {
        Examen examen = new Examen();
        try{
            ResultSet rs = null;
            Connection con = ConexionBD.iniciarConexion();
            String sql = "SELECT * FROM mexamen WHERE nck_usu = ? and com_ex
a < 10;";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, nick);
            rs = ps.executeQuery();
            if (rs.next()) {
                examen.setNick(rs.getString("nck_usu"));
                examen.setCode(rs.getInt("id_exa"));
                examen.setFecha(rs.getString("fec_exa"));
                examen.setTitle(rs.getString("tit_exa"));
            }
            con.close();
        }catch(Exception ex) {
            System.out.println("Error en ultimoExamen "+ex.getMessage());
        }
        return examen;
    }

    public int totalExamenes(String nick) {
        int total = 0;
```

```
        try{
            ResultSet rs = null;
            Connection con = ConexionBD.iniciarConexion();
            String sql = "SELECT count(*) FROM mexamen WHERE nck_usu = ?";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, nick);
            rs = ps.executeQuery();
            if (rs.next()) {
                total = rs.getInt(1);
            }
            con.close();
        }catch(Exception ex) {
            System.out.println("Error en totalExamens "+ex.getMessage());
        }
        return total;
    }

    public int guardaProgreso(Examen exa) {
        int status = 0;

        try{
            ResultSet rs = null;
            Connection con = ConexionBD.iniciarConexion();
            String sql = "INSERT INTO dexamen(id_exa, id_rea) values (?, ?),
(?, ?), (?, ?), (?, ?), (?, ?), (?, ?), (?, ?), (?, ?), (?, ?), (?, ?);";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setInt(1, exa.getCode());
            ps.setInt(2, exa.getReactivos()[0].getCode());
            ps.setInt(3, exa.getCode());
            ps.setInt(4, exa.getReactivos()[1].getCode());
            ps.setInt(5, exa.getCode());
            ps.setInt(6, exa.getReactivos()[2].getCode());
            ps.setInt(7, exa.getCode());
            ps.setInt(8, exa.getReactivos()[3].getCode());
            ps.setInt(9, exa.getCode());
            ps.setInt(10, exa.getReactivos()[4].getCode());
            ps.setInt(11, exa.getCode());
            ps.setInt(12, exa.getReactivos()[5].getCode());
            ps.setInt(13, exa.getCode());
            ps.setInt(14, exa.getReactivos()[6].getCode());
            ps.setInt(15, exa.getCode());
            ps.setInt(16, exa.getReactivos()[7].getCode());
            ps.setInt(17, exa.getCode());
            ps.setInt(18, exa.getReactivos()[8].getCode());
            ps.setInt(19, exa.getCode());
```

```
            ps.setInt(20, exa.getReactivos()[9].getCode());
            status = ps.executeUpdate();
            con.close();
        }catch(Exception ex) {
            System.out.println("Error en guardaProgreso "+ex.getMessage());
        }

        return status;
    }

    public int updateProgreso(Examen exa) {
        int status = 0;

        try{
            ResultSet rs = null;
            Connection con = ConexionBD.iniciarConexion();
            String sql = "UPDATE mexamen SET sco_exa = ?, com_exa = ? where
id_exa = ?;";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setInt(1, exa.getScore());
            ps.setInt(2, exa.getProgreso());
            ps.setInt(3, exa.getCode());
            status = ps.executeUpdate();
            con.close();
        }catch(Exception ex) {
            System.out.println("Error en updateProgreso "+ex.getMessage());
        }

        return status;
    }


    //
    public Examen getExamen(String nick) {
        Examen examen = new Examen();
        try{
            ResultSet rs = null;
            Connection con = ConexionBD.iniciarConexion();
            String sql = "SELECT nck_usu,id_exa,tit_exa, fec_exa, sco_exa, c
om_exa, pre_rea, opc_rea, sol_rea from mexamen natural join dexamen natural
join mreactivo WHERE nck_usu = ? and com_exa < 10;";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, nick);
            rs = ps.executeQuery();
```

```
            if (rs.next()) {
                examen.setNick(rs.getString("nck_usu"));
                examen.setCode(rs.getInt("id_exa"));
                examen.setFecha(rs.getString("fec_exa"));
                examen.setTitle(rs.getString("tit_exa"));
                examen.setProgreso(rs.getInt("com_exa"));
                examen.setScore(rs.getInt("sco_exa"));

                examen.setReactivos(this.getPreguntasByCode(rs.getInt("id_ex
a")));
            }
            con.close();
        }catch(Exception ex) {
            System.out.println("Error en getExamen "+ex.getMessage());
        }
        return examen;

    }
```

Capturas de pantalla

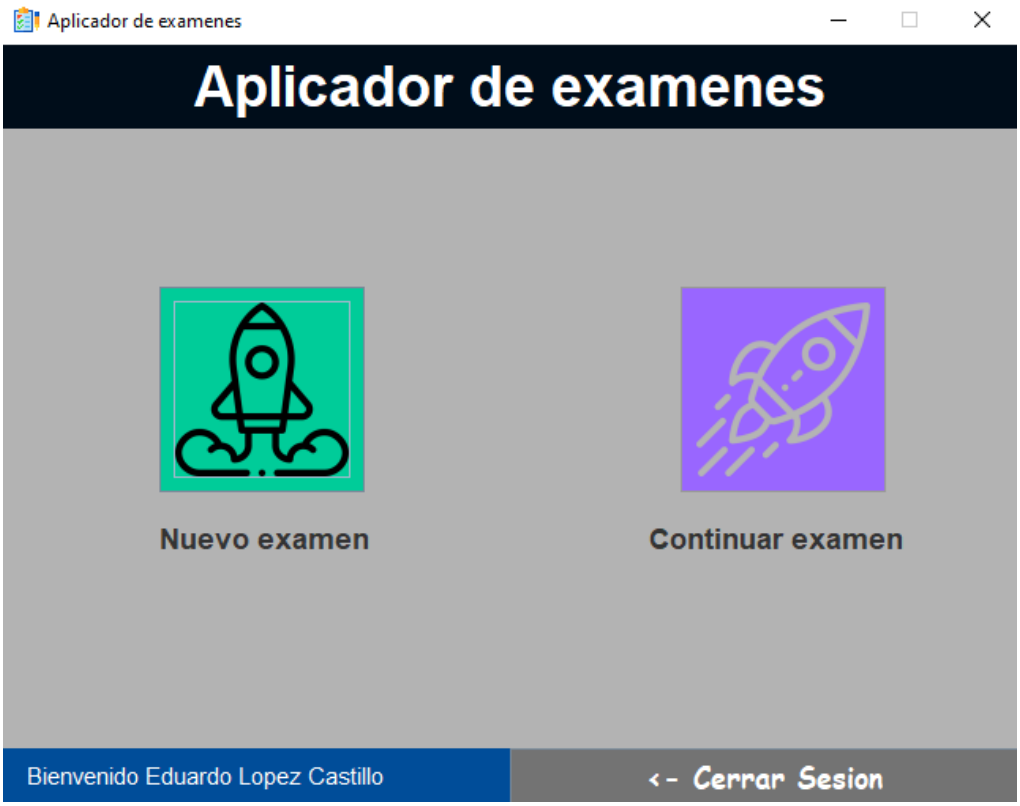
Login


Examensito
Acceso al Sistema
INSTITUTO POLITÉCNICO NACIONAL
ESCOM
Usuario
admin
Bienvenido ADMINISTRADOR
Inicio de sesión exitoso
Aceptar
Aceptar
Cancelar

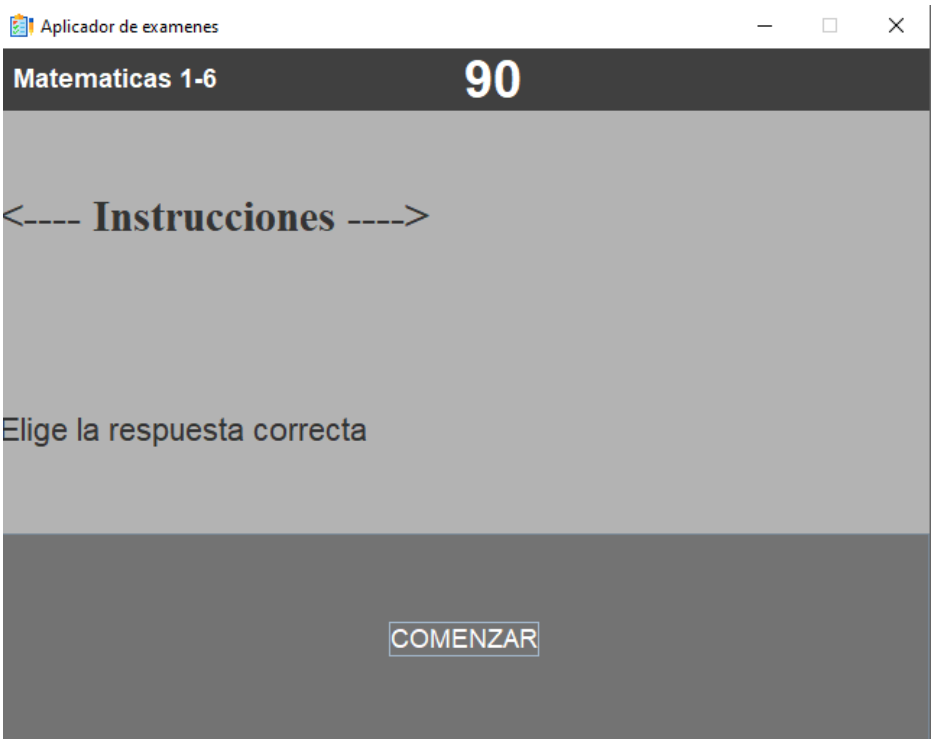

Editor de reactivos
Editor de Reactivos
Pregunta: CUATRO VECES Z MENOS TRES VECES LA SUMA DE X Y Y
a) 4Z - 3X +3Y
b) 4Z - 3 (X + Y)
c) 4Z + 3XY
d) 4Z-3X+Y
Pregunta: Z MAS EL PRODUCTO DE X Y Y
a) Z + 1 (X X Y)
b) Z-(X+Y)
c) Z (X Y)
d) Z+XY
Pregunta: PERIMETRO DE UN RECTÁNGULO DONDE SU LONGITUD ES L Y SU ANCHURA W.
a) P = 2 (L + W)
b) P = 2L + W
c) P = 2L + 2W
d) P = 4L + W
Pregunta: 6X = - 6
a) 6
b) 1
c) -1
d) 0
Pregunta: -64X = 16
a) 4
b) -1/4
c) 80
d) 48
Pregunta: X / 4 = 8
a) 32
+ Añadir reactivo
<- Cerrar Sesion

Acceso como administrador

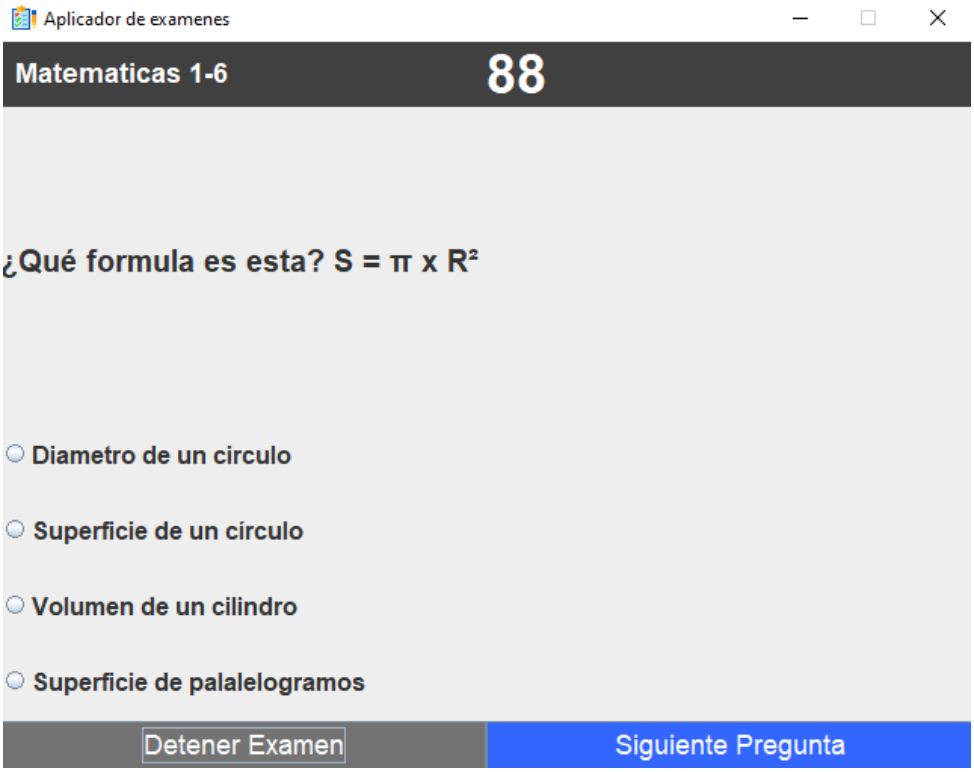
Acceso como usuario



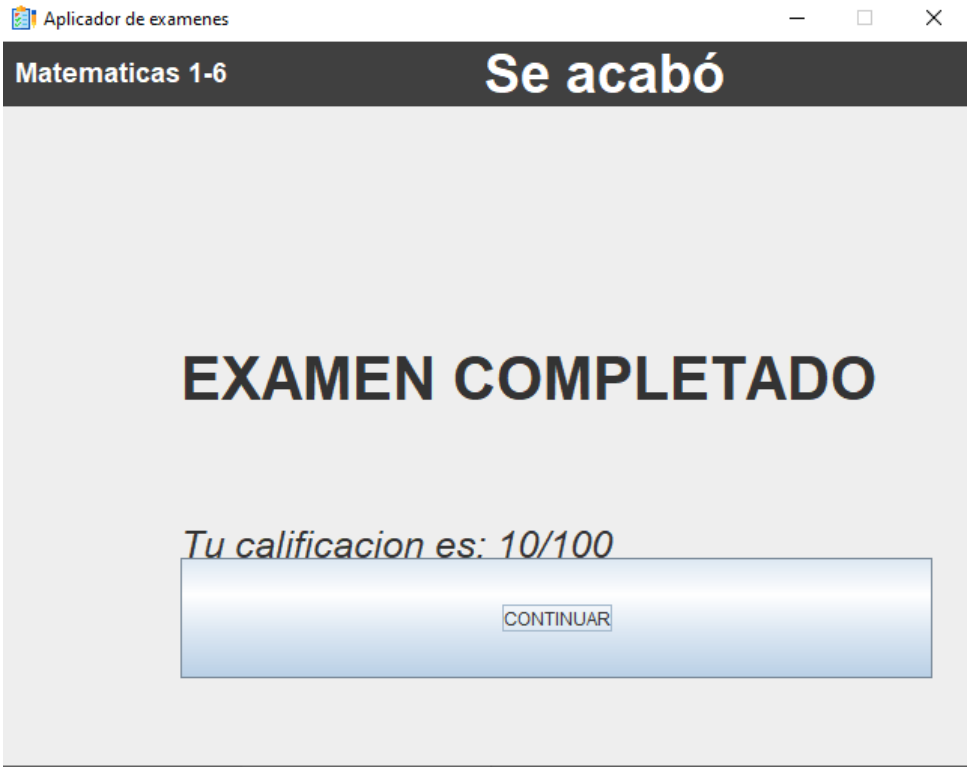
Pantalla de examen



Pantalla de instrucciones



Preguntas



Resultados

Conclusiones generales (ambos)

A nuestro parecer el proyecto mas que tener dificultades en la interfaz el verdadero reto fue en la parte de las bases de datos, ya que la parte del back decidia todo el funcionamiento dentro de la misma y como generar las relaciones necesarias entre ellas para que funcionara de forma optima junto a la interfaz.