

18 de Mayo del 2015

PROGRAMMER



UPLSP

MANUAL DE PROGRAMADOR

</CODE>

DAC

Universidad Politécnica de SLP

Proyecto Final

Programación III

E15-634

Intranet

Integrantes del equipo:

- AGUSTIN IRVIN GARCIA PEREZ
- SIDNEY ALEXA WALDO HERNANDEZ
- GERARDO FRANCO DELGADO

José Ramón Tolentino Jiménez

18 de Mayo del 2015

Contenido

| | | |
|---|----|---|
| Intranet: | 5 | 3 |
| Programa: | 5 | |
| Objetivo: | 6 | |
| Justificación: | 6 | |
| Alcance y limitaciones: | 6 | |
| Introducción: | 7 | |
| Proyecto en GitHub: | 7 | |
| Nuestro proyecto: | 7 | |
| ¿Qué es control de versión? | 8 | |
| ¿Qué es git? | 9 | |
| ¿Qué es GitHub? | 9 | |
| Requisitos: | 10 | |
| Conocimientos previos: | 10 | |
| ¿Qué es modelo vista controlador? | 10 | |
| ¿Qué es javaFX? | 11 | |
| ¿Qué es FXML? | 11 | |
| Configurar eclipse: | 12 | |
| Crear Archivo FXML: | 14 | |
| Parte 1: Explicación: | 15 | |
| Parte 2: DataBaseSQL | 15 | |
| Conexión base de datos: | 15 | |
| Parte 3: LogIn: | 18 | |
| Parte Gráfica: | 18 | |
| Programación: | 18 | |
| Parte 4: MainAdministrador | 20 | |
| Registrar Profesor interfaz: | 20 | |
| Registrar Profesor | 21 | |
| Eliminar interfaz | 22 | |
| Eliminar profesor: | 22 | |
| Buscar Profesor interfaz: | 22 | |
| Buscar Profesor: | 23 | |
| Parte 5: DataAlumn | 23 | |

| | | |
|---|----|---|
| Parte 6: MainAlumno | 24 | |
| Capítulo 7: Calificaciones | 27 | 4 |
| Capítulo 8: MainProfesor | 27 | |
| Capítulo 9: Validación..... | 28 | |
| Capítulo 10: Alumnos, Grupos, Materias, Profesores y Administradores..... | 28 | |
| Diagrama UML:..... | 29 | |
| Tabla de ilustraciones..... | 30 | |
| Bibliografía | 31 | |

Intranet:

Programa:

El programa se encargará de automatizar el control de una intranet, para la elaboración de esta, se utilizarán bases de datos, por lo que el lenguaje de programación será Java con mysql, para así cumplir con las siguientes funcionalidades:

El administrador podrá:

- Registrar n número de maestros
- Registrar n número de alumnos
- Registrar n número de materias
- Registrar n número de administradores
- Eliminar algún maestro, alumno o materia.
- Consultas de profesores, materias y alumnos
- Inscribir alumno en un grupo
- Habilitar que parciales el profesor puede calificar

El profesor podrá:

- Ver su información
- Registrar las calificaciones de un alumnos
- Registrar faltas de un alumno
- Corregir calificaciones y/o faltas en un caso dado

El alumno podrá:

- Ver su información
- Ver sus calificaciones
- Ver sus faltas



Ilustración 1, Análisis Intranet

Cabe mencionar que al principio, el programa tendrá un login, para identificar y diferenciar si es un alumno, un profesor o administrador.

Los conceptos que se utilizarán para la elaboración de este proyecto será el manejo de bases de datos, encapsulamiento, herencia, clases abstractas e implementación, y en general las buenas prácticas y conocimientos de programación orientada a objetos con el lenguaje de programación java.

Objetivo:

Poder dar una solución a un sistema educativo cualquiera, para llevar un control automatizado de calificaciones, ayudando y facilitando la organización de cualquier institución educativa.

Otro de los objetivos es difundir el conocimiento de javaFX, y como se utiliza para crear aplicaciones, ya que como hay muy poca información acerca de su utilización puede ser gran aporte a la comunidad.



Ilustración 2, Objetivos Intranet

Justificación:

Compartimos la ideología de opensource por lo que queremos compartir este proyecto para que pueda ser mejorado, y enseñe a las personas que quieran programar en javaFX como se podría hacer.

Alcance y limitaciones:

El alcance que se tendría sería a un sistema de control de calificaciones (intranet), dentro de las limitaciones, el programa estaría pensado para una institución educativa solamente, que facilitará el proceso de la organización para controlar las calificaciones y asignar materias y grupos a los profesores.

Introducción:

En este proyecto se escogió javaFX, ya que quisimos investigar y crear un software con esta opción, como no se tiene tanta documentación, nuestro objetivo es subir este proyecto a GitHub con su respectivos manuales de programación y de usuario, para ayudar a los programadores que les interesa desarrollar software con javaFX, en entender cómo funciona, y como se puede programar.

Antes de explicar el código es necesario decir lo que se utilizó en este proyecto, desde la instalación de javaFX y la explicación de cómo funciona, y decir que IDE se utilizó, y como se fue trabajando para la creación del proyecto, ya que todo se creó por versión de control.

Proyecto en GitHub:

<https://github.com/gfdgeras/JavaFX-INTRANET>

Nuestro proyecto:



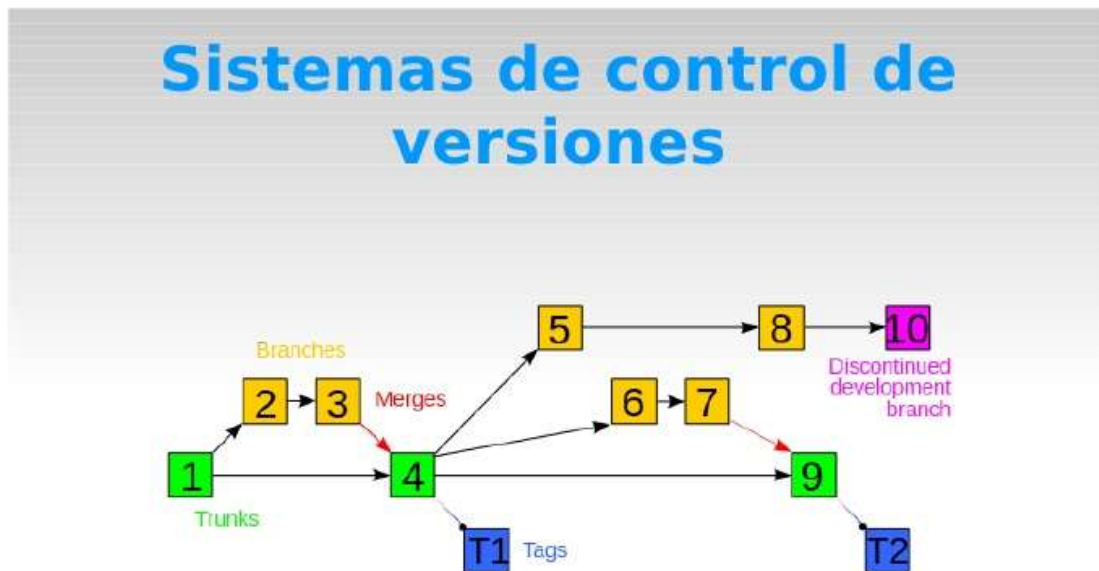
open source

Ilustración 3, Ideología Intranet

Nuestro software es opensource, con dos finalidades, difundir el conocimientos a otros usuarios de cómo se puede crear aplicaciones con javaFX, y la segunda que este se pueda mejorar por otros usuarios para que se acople a sus necesidades y lo puedan llegar a utilizar en escuelas públicas, privadas, etc.

Se eligió usar gitHub como plataforma para difundir el proyecto, y para trabajar en la creación del desarrollo de software, ya que gracias a su

control de versión, se pueden ir guardando los avances por medio de commits, así como también otra ventaja es que diferentes usuarios pueden ir modificando código, cada quien en sus computadoras, y al subirlo se fusiona con los avances de los demás, es perfecto para trabajar en un grupo de programadores.

*Ilustración 4, Control de versiones*

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación.

Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión dando lugar a los llamados sistemas de control de versiones o VCS (del inglés Version Control System).

Estos sistemas facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas (por ejemplo, para algún cliente específico).

Ejemplos de este tipo de herramientas son entre otros: CVS, Subversion, SourceSafe, ClearCase, Darcs, Bazaar, Plastic SCM, Git, Mercurial, Perforce, Fossil SCM.

El control de versiones se realiza principalmente en la industria informática para controlar las distintas versiones del código fuente dando lugar a los sistemas de control de código fuente o SCM (siglas del inglés Source Code Management). Sin embargo, los mismos conceptos son aplicables a otros ámbitos como documentos, imágenes, sitios web, etc.

¿Qué es git?



Ilustración 5, Logo Git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Al

principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o front end como Cogito o StGIT. Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena. Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de programación del núcleo Linux.

El mantenimiento del software Git está actualmente (2009) supervisado por Junio Hamano, quien recibe contribuciones al código de alrededor de 280 programadores.

¿Qué es GitHub?

GitHub es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Utiliza el framework Ruby on Rails por GitHub, Inc. (anteriormente conocida como Logical Awesome). Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc.



Ilustración 6, Logo GitHub

Requisitos:

- Última versión de Java JDK 8 (includes JavaFX 8).
 - <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- Eclipse 4.3 o superior con el plugin e(fx)clipse. La forma más sencilla de obtenerlo es descargarse la distribución preconfigurada desde e(fx)clipse website. Como alternativa puedes usar un sitio de actualización para tu instalación de Eclipse.
 - <http://efxclipse.bestsolution.at/install.html#all-in-one>
- Scene Builder 2.0 o superior
 - <http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html>

Conocimientos previos:

¿Qué es modelo vista controlador?

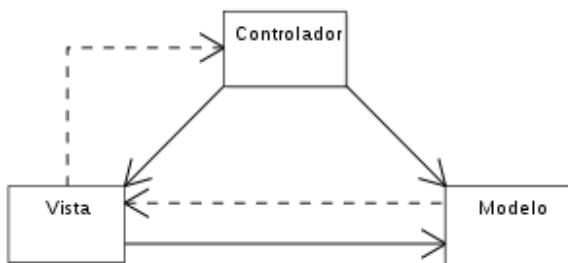


Ilustración 7, Modelo Vista Controlador

El modelo–vista–controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres

componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

¿Qué es javaFX?

JavaFX es una familia de productos y tecnologías de Sun Microsystems, adquirida por Oracle Corporation, para la creación de Rich Internet Applications (RIAs), esto es, aplicaciones web que tienen las características y capacidades de aplicaciones de escritorio, incluyendo



Ilustración 8, JavaFX

aplicaciones multimedia interactivas. Las tecnologías incluidas bajo la denominación JavaFX son JavaFX Script y JavaFX Mobile, aunque hay más productos JavaFX planeados.

Las aplicaciones JavaFX pueden ser ejecutadas en una amplia variedad de dispositivos. En su versión (JavaFX 1.3, abril 2010) permite crear aplicaciones de escritorio, para celulares, la Web, TV, consolas de videojuegos, reproductores Blu-ray, entre otras plataformas planeadas. En octubre de 2011 fue lanzada la versión 2.0. Para el desarrollo de aplicaciones JavaFX un lenguaje declarativo, tipado llamado JavaFX Script, además puede integrarse código Java en programas JavaFX. JavaFX es compilado a código Java, por lo que las aplicaciones JavaFX pueden ser ejecutadas en computadores con la máquina virtual de Java instalada (JRE), o celulares corriendo Java ME.

JavaFX fue anunciado en la conferencia de desarrolladores JavaOne en mayo de 2007 y liberado en diciembre de 2008. La intención de Sun Microsystems respecto de JavaFX es competir en el espacio que ya ocupan Flash de Adobe, y Silverlight de Microsoft.

En palabras de James Gosling "La mayoría de los lenguajes de script están orientados a las páginas web; éste está orientado a las interfaces que son altamente animadas"

¿Qué es FXML?

FXML es un lenguaje basado en XML y utilizado para definir interfaces en JavaFX. Aunque java JavaFX permite crear interfaces mediante su código, al usar FXML es mucho más sencillo adaptar las ventanas y paneles a los diferentes tamaños necesarios. Es ideal para diseñar cualquier interfaz de usuario ya que es compatible con la clase Scene de JavaFX.

Configurar eclipse:

Hay que indicarle a Eclipse que use JDK 8 y también dónde se encuentra el ejecutable del Scene Builder:

1. Abre las Preferencias de Eclipse (menú Window | Preferences y navega hasta Java | Installed JREs.
2. Si no lo tienes el jre1.8 en la lista de JREs, entonces pulsa Add..., selecciona Standard VM y elige el Directorio de instalación (JRE Home directory) de tu JDK 8.
3. Elimina otros JREs o JDKs de tal manera que JDK 8 se convierta en la opción por defecto.

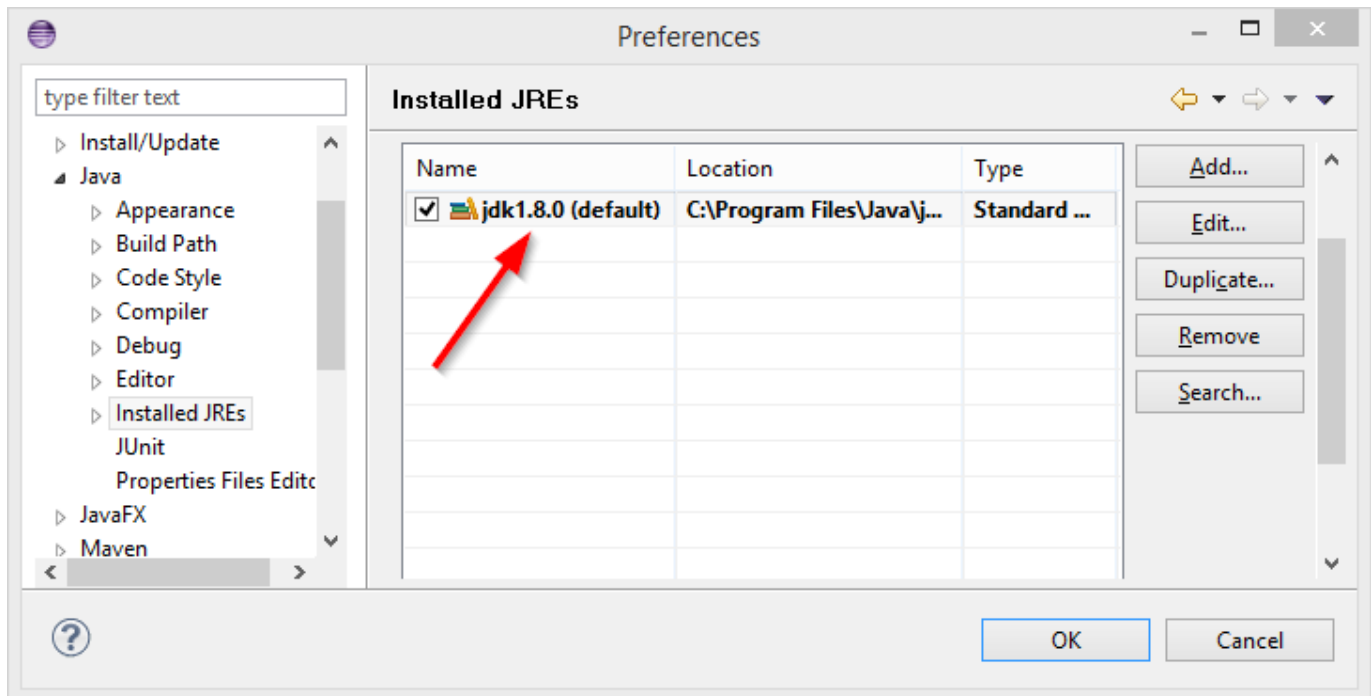


Ilustración 9, Configurar eclipse paso 1

4. Navega a Java | Compiler. Establece el nivel de cumplimiento del compilador en 1.8 (Compiler compliance level).

13

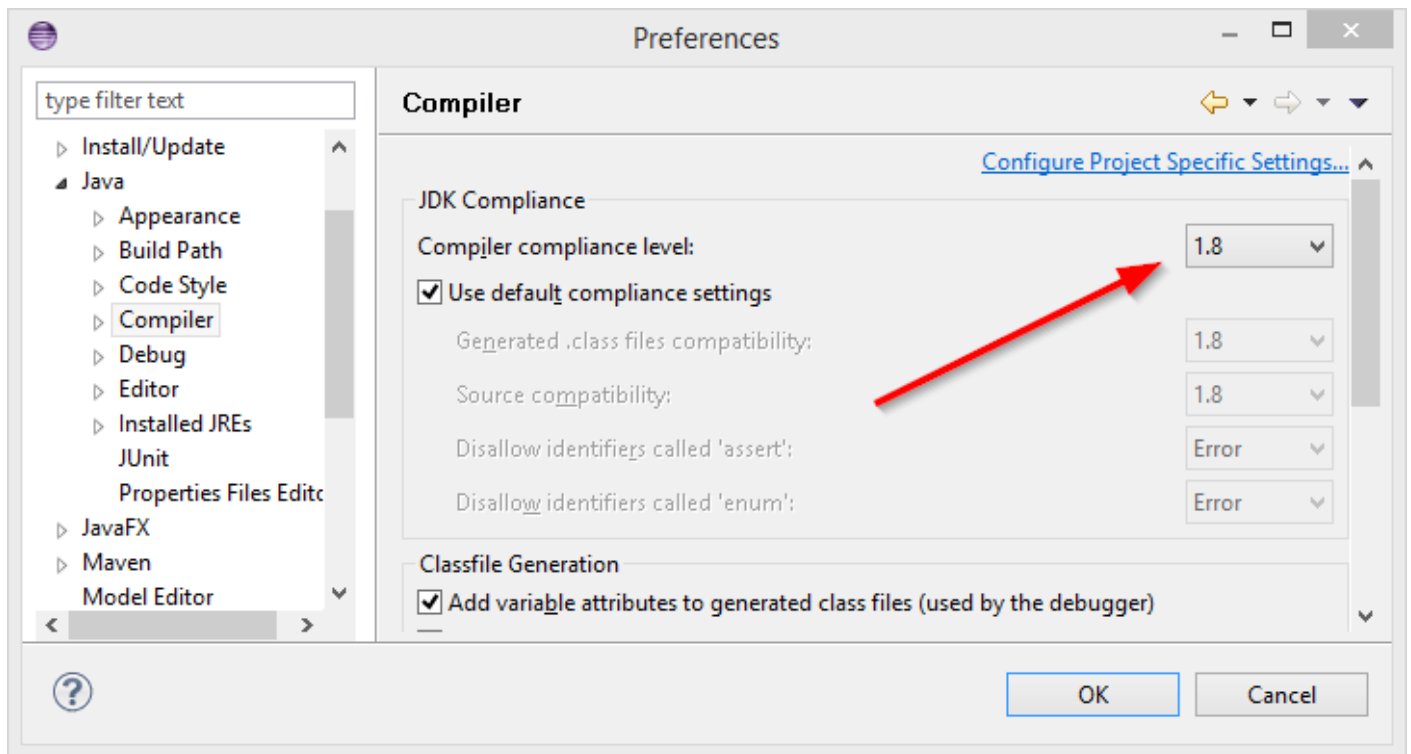


Ilustración 10, Configuración javaFX

5. Navega hasta Java | JavaFX. Especifica la ruta al ejecutable del Scene Builder.

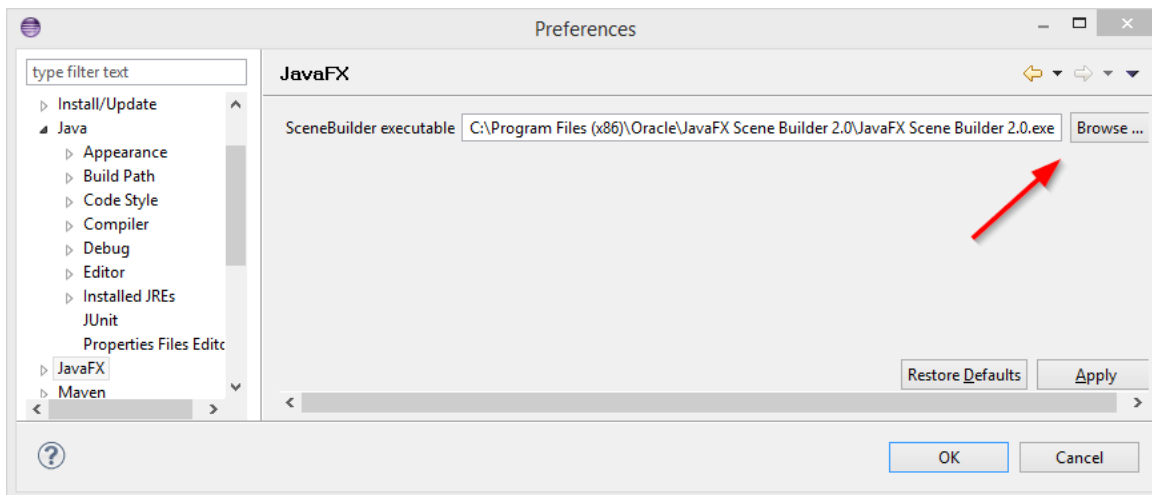


Ilustración 11, Configuración eclipse

Crear Archivo FXML:

14

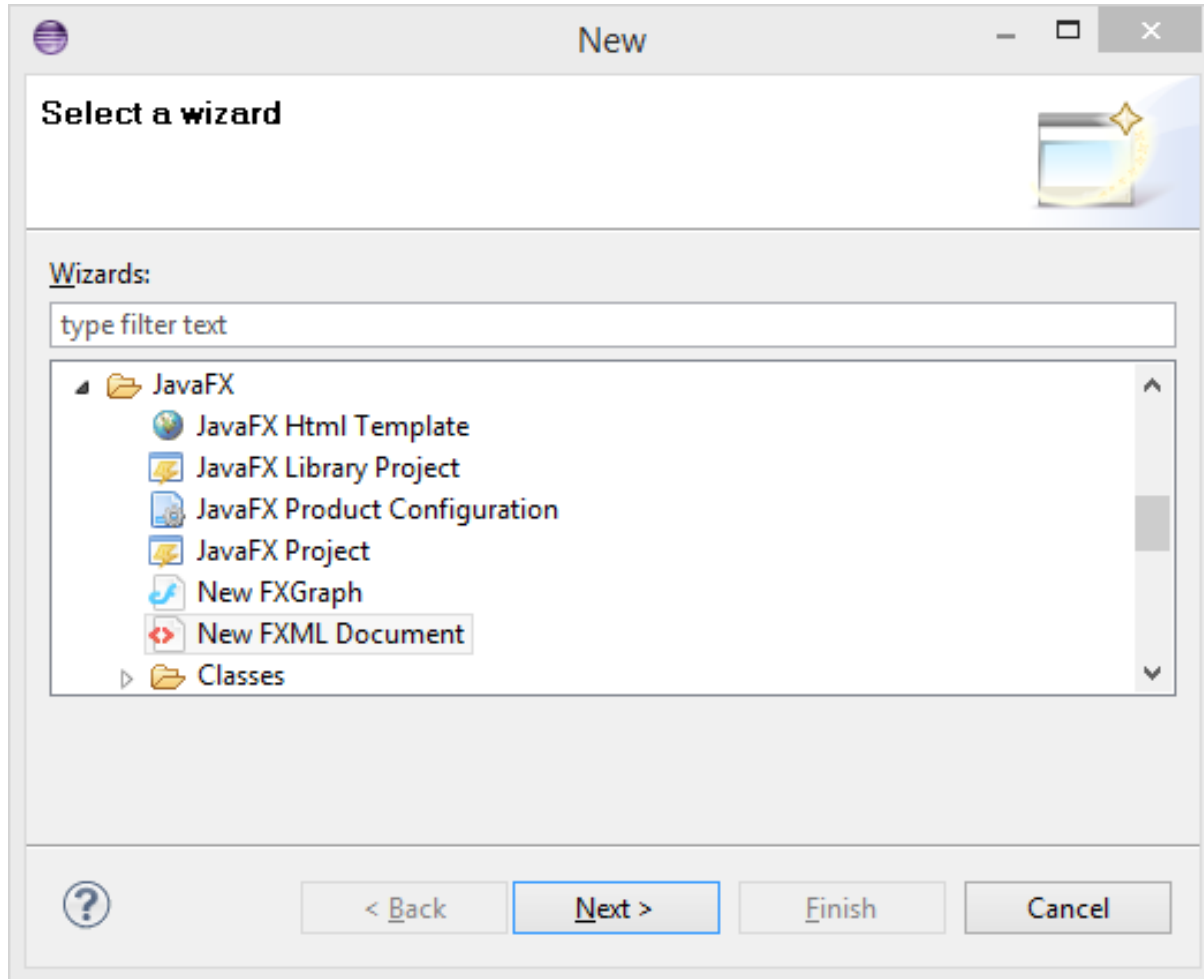


Ilustración 12, Archivo FXML

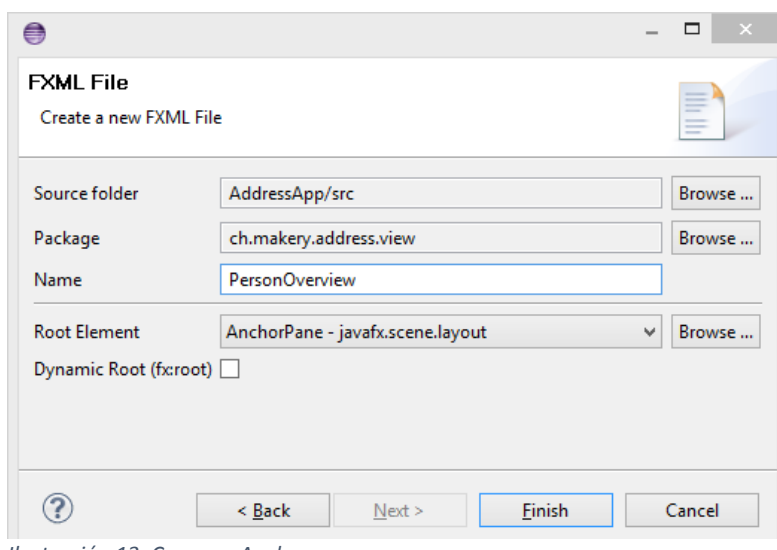


Ilustración 13, Crear un Anchor

Parte 1: Explicación:

Para la creación de este software se crearon paquetes para tener por separado lo que es el diseño de la programación, por lo que todos los archivos FXML se encuentran en un paquete separado, en la carpeta "view". Cabe decir que cada escenario tiene sus propios bordes, y su anchor; Lo que contiene los bordes es una barra de tareas, donde se pueden realizar varias opciones dependiendo de lo que el usuario quiera.

En este manual se explicará paso por paso la creación del software, y las clases que se utilizaron.

Parte 2: DataBaseSQL

Esta clase es la más importante ya que en ella se crea la conexión con la base de datos, y también contiene todos los métodos que son necesarios para hacer búsquedas, insertar datos, eliminar, y realizar consultas específicas, son métodos que ya hacen todo, y los queremos compartir con la comunidad de programadores.

Conexión base de datos:

```
public DataBaseSQL(){
    this.driver = "com.mysql.jdbc.Driver";
    this.pass = "";
    this.user = "root";
    this.server = "jdbc:mysql://localhost/intranet";
    try{
        Class.forName(driver);
        connection = DriverManager.getConnection(server,user,pass);
        System.out.println("Conexion realizada con exito");
    }catch(ClassNotFoundException | SQLException e){
        System.out.println(e.getMessage());
        JOptionPane.showMessageDialog(null, "Conexion no realizada a la base de datos.");
    }
}
```

Ilustración 14, Conexión Base de datos

Para poder usar estas clases se debe de importar jdbc.jar, y como se puede observar la conexión se hace en localhost, y se conecta con la base de datos llamada intranet.

A continuación se mostrarán ejemplo de los métodos más significantes de esta clase:

```
//Metodo para obtener un registro de la base de datos
public HashMap<String, String>fetchArray(String table, String delimiter, String index){
    //Todo lo almacenamos en un mapa
    HashMap<String, String> mapa = new HashMap<String, String>();

    try{
        Statement query = (Statement) connection.createStatement();
        String comando = "SELECT * FROM " + table + " WHERE " + delimiter + " = " + index;
        System.out.println("fetchArray: " + comando);
        ResultSet rs = query.executeQuery(comando);

        if( !rs.next() ){
            System.out.println("VACIO");
            return null;
        }
        ResultSetMetaData rsmd = rs.getMetaData();
        rs.first();
        String valor, key;

        int cant = rsmd.getColumnCount();

        for(int i = 1; i <= cant; i++){
            key = rsmd.getColumnName(i);
            valor = rs.getString(key);
            //System.out.println(key + "-->" + valor);
            mapa.put(key, valor);
        }

    }catch(Exception e){
        System.out.print("Error " + e);
        return null;
    }
    return mapa;
}
```

Ilustración 15, Metodo importante para BD

De lo más importante es el uso de HashMap, tabla hash, matriz asociativa, mapa hash, tabla de dispersión o tabla fragmentada es una estructura de datos que asocia llaves o claves con valores. La operación principal que soporta de manera eficiente es la búsqueda: permite el acceso a los elementos (teléfono y dirección, por ejemplo) almacenados a partir de una clave generada (usando el nombre o número de cuenta, por ejemplo). Funciona transformando la clave con una función hash en un hash, un número que identifica la posición (casilla o cubeta) donde la tabla hash localiza el valor deseado.

Las tablas hash se suelen implementar sobre vectores de una dimensión, aunque se pueden hacer implementaciones multidimensionales basadas en varias claves.

Como en el caso de los arrays, las tablas hash proveen tiempo constante de búsqueda promedio $O(1)$, sin importar el número de elementos en la tabla. Sin embargo, en casos particularmente malos el tiempo de búsqueda puede llegar a $O(n)$, es decir, en función del número de elementos.

Comparada con otras estructuras de arrays asociadas, las tablas hash son más útiles cuando se almacenan grandes cantidades de información.

Las tablas hash almacenan la información en posiciones pseudo-aleatorias, así que el acceso ordenado a su contenido es bastante lento. Otras estructuras como árboles binarios auto-balanceables tienen un tiempo promedio de búsqueda mayor (tiempo de búsqueda $O(\log n)$), pero la información está ordenada en todo momento.

Ahora un ejemplo de cómo insertamos datos en la base de datos:

```
//Metodo que inserta valores en una tabla, NOTA: TODO se inserta en mayusculas
public boolean insert(String tabla, String[] values){
    try {
        Statement query = (Statement) connection.createStatement();
        String q1;
        //Si la tabla es profesor, alumno o administrador, la contraseña la ingresamos tal cual
        if(tabla.equals("profesor") || tabla.equals("alumno") || tabla.equals("administrador")){
            int i = 1;
            q1 = "insert into " + tabla + " values("; //Vamos creando el query

            for(String txt : values){ //Agregamos los valores
                if(i < values.length)
                    q1 += txt.toUpperCase() + ", "; //Pasamos todo a mayuscula
                else
                    q1 += txt + ", "; //Si es password lo mandamos tal cual
                i++;
            }
        }else{
            q1 = "insert into " + tabla + " values(";

            for(String txt : values){
                if(txt.equals("Default") || txt.equals("NULL")){
                    q1 += txt + ", ";
                }else
                    q1 += txt.toUpperCase() + ", "; //Agregamos los valores
            }
        }

        q1 = q1.substring(0, q1.length()-2); //Eliminamos la coma de mas
        q1 += ")";

        System.out.println("insert: " + q1);
        query.executeUpdate(q1); //Ejecutamos el query
        System.out.println("Exito");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Ilustración 16, Metodo para insertar datos en la BD

Parte 3: Login:

Parte Gráfica:

Se creó con Scene Builder



Ilustración 17, Interfaz de login

Programación:

De lo más importante, es saber cómo se selecciona el tipo de usuario, como se hace la consulta en la base de datos, y como los elementos de lado de FXML se relacionan con el controlador.

A continuación vemos como se relacionan los elementos de FXML con el controlador:

```

@FXML private TextField clave;
@FXML private TextField pass;
@FXML private RadioButton radioAlumno;
@FXML private RadioButton radioProfesor;
@FXML private RadioButton radioAdministrador;

```

Ilustración 18, Se hace la conexión entre FXML y el controlador

Se hace la conexión a la base de datos

```

//Base de datos.....
DataBaseSQL db = new DataBaseSQL();
HashMap<String, String> dataS;
String data;

```

Se crea un método que se manda llamar desde el fxml cuando el botón es presionado, una vez que este es presionado se selecciona que tipo de usuario es:

```

//Si es seleccionado el alumno en radio button.....
if(radioAlumno.isSelected()){
    //Se guardan en variables lo ingresado en los textfield
    claveUsuario = clave.getText();
    passUsuario = pass.getText();
    System.out.println("-" + passUsuario);
    if((dataS = db.fetchArray("alumno", "MATRICULA", claveUsuario)) != null){
        if(dataS.get("PASSWORD").equals(passUsuario)){
            //Con polimorfismo se adapta a la forma de la clase hijo, de
            Application app2 = new MainAlumno(claveUsuario);
            //Inicia esenario nuevo
            Stage anotherStage = new Stage();
            //Por si hay excepciones se pone try and catch
            try {
                app2.start(anotherStage); //Se inicia la aplicacion
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace(); //IMPRIME ERROR
            }
        }else{
            JOptionPane.showMessageDialog(null, "Contraseña no valida")
            System.out.println(dataS.get("PASSWORD"));
        }
    }
    else
        JOptionPane.showMessageDialog(null, "Usuario no valido");
}

```

Ilustración 19, Dependiendo del radioButton se inicializa una aplicación

Como se puede ver en la imagen se obtienen los datos que están en los textfield, y si concuerda el usuario con la contraseña se iniciara la aplicación dependiendo del usuario que seas, cabe decir que se usa polimorfismo en esta sección para que la clase padre aplicación cambie a su clase hijo. De forma muy similar se realiza lo mismo con administrador y alumno.

Parte 4: MainAdministrador

En esta parte el administrador podrá registrar, eliminar, hacer consultas, y hacer inscripciones, con el fin de ser más dinámico este manual solo se mostrará un ejemplo de cada uno:

Registrar Profesor interfaz:




Ilustración 20, Interfaz para registrar Profesor

```
public void sendPrtsr(){

    DataBaseSQL db = new DataBaseSQL();
    String[] data = new String[7];
    data[0] = nomP.getText();
    if(!Validaciones.isName(data[0], 50, "nombre"))
        return;
    data[1] = rfcP.getText();
    if(!Validaciones.isExactSize(data[1], 10, "RFC"))
        return;
    data[2] = telP.getText();
    if(!Validaciones.isPhone(data[2]))
        return;
    data[3] = dirP.getText();
    if(!Validaciones.isCorrectSize(data[3], 50, "Direccion"))
        return;
    data[4] = usuarioP.getText();
    if(!Validaciones.isPositiveInt(data[4], "Clave", 4))
        return;
    data[5] = emailP.getText();
    if(!Validaciones.isMail(data[5], 30))
        return;

    if(passP.getText().equals(pass2P.getText()))
        data[6] = passP.getText();
    else{
        error("Las contraseñas no concuerdan");
        return;
    }
}
```

Ilustración 21, Código para registrar a un profesor

Eliminar interfaz

Ilustración 22, Interfaz para eliminar un registro

Eliminar profesor:

```
public void deleteProfe(){
    DataBaseSQL db = new DataBaseSQL();

    if(db.free("DELETE FROM PROFESOR WHERE CLAVE = " + delUserP.getText()))
        error("Usuario eliminado con exito");
    else
        error("Error al eliminar el usuario");
}
```

Buscar Profesor interfaz:

Ilustración 23, Búsqueda específica

Buscar Profesor:

```
public void searchProf(){
    DataBaseSQL db = new DataBaseSQL();
    HashMap<String, String> data;

    data = db.fetchArray("PROFESOR", "CLAVE", bclaveP.getText());

    if(data == null){
        error("El profesor no existe");
        return;
    }

    bnomP.setText(data.get("NOMBRE"));
    brfcP.setText(data.get("RFC"));
    bdirP.setText(data.get("DIRECCION"));
    btelP.setText(data.get("TEL"));
    bemailP.setText(data.get("EMAIL"));
}
```

Ilustración 24, Código para buscar a un profesor y agregarlo

Parte 5: DataAlumn

En esta clase se usa para que en la aplicación de alumnos se puedan insertar datos en la tabla, es decir que en esta tabla vienen todos los atributos de los alumnos que se van a mostrar en la tabla, es importante señalar que para que funcione se necesita los get de los atributos.

A continuación se muestra el código más importante de esta clase:

```
public SimpleStringProperty clave = new SimpleStringProperty();
public SimpleStringProperty p1 = new SimpleStringProperty();
public SimpleStringProperty p2 = new SimpleStringProperty();
public SimpleStringProperty p3 = new SimpleStringProperty();
public SimpleStringProperty tP = new SimpleStringProperty();
public SimpleStringProperty fin = new SimpleStringProperty();
public SimpleStringProperty tFin = new SimpleStringProperty();
public SimpleStringProperty total = new SimpleStringProperty();
public SimpleStringProperty extra = new SimpleStringProperty();
public SimpleStringProperty inas = new SimpleStringProperty();
```

Ilustración 25, Clase SimpleStringProperty para realizar la tabla de javaFX

Para utilizar SimpleStringProperty es necesario utilizar la librería javafx.beans.property.SimpleStringProperty, como se puede observar se crean los atributos que irán en la tabla, todos de tipo String, pero para que pueda funcionar se deben de poner los get que se muestran algunos como ejemplo

```
public String getClave(){  
    return clave.get();  
}  
public String getP1(){  
    return p1.get();  
}  
public String getP2(){  
    return p2.get();  
}  
public String getP3(){  
    return p3.get();  
}  
public String getTP(){  
    return tP.get();  
}
```

Ilustración 26, gets necesarios para javaFX

Con estos get desde la clase MainAlumno se obtendrá la lista de los alumnos con sus respectivos valores.

Parte 6: MainAlumno

En esta clase se usan tablas con javaFX, por eso creemos que en cuestión de programación nuestro software es innovador, ya que los métodos que usa para la base de datos y como maneja la información en tablas, hay muy poca información, por eso mismo estamos orgullosos de nuestro proyecto, creemos que va hacer un gran aporte a la comunidad de desarrolladores de java.

Se mostrará la relación entre vista y controlador:



```

/* Datos Alumno */
@FXML TextField nomA;
@FXML TextField mata;
@FXML TextField tela;
@FXML TextField dirA;
@FXML TextField mailA;
@FXML TextField carreraA;

/* Tabla */
@FXML private TableView<DataAlumn> tableMaterias;
@FXML private TableColumn claveM;
@FXML private TableColumn p1;
@FXML private TableColumn p2;
@FXML private TableColumn p3;
@FXML private TableColumn tP;
@FXML private TableColumn fin;
@FXML private TableColumn tFin;
@FXML private TableColumn calificacionFinal;
@FXML private TableColumn extra;
@FXML private TableColumn inas;
ObservableList<DataAlumn> materias;

```

Ilustración 27, FXML a controlador

De lo más importante es `ObservableList<DataAlumn>`, ya que con esto, es aquí donde en forma de lista se guardaran a los alumnos.

Ahora se ve verá cómo se debe de inicializar una tabla en javaFX.

```

public void initTable(){
    claveM.setCellValueFactory(new PropertyValueFactory<DataAlumn, String>("clave"));
    p1.setCellValueFactory(new PropertyValueFactory<DataAlumn, String>("p1"));
    p2.setCellValueFactory(new PropertyValueFactory<DataAlumn, String>("p2"));
    p3.setCellValueFactory(new PropertyValueFactory<DataAlumn, String>("p3"));
    tP.setCellValueFactory(new PropertyValueFactory<DataAlumn, String>("tP"));
    fin.setCellValueFactory(new PropertyValueFactory<DataAlumn, String>("fin"));
    tFin.setCellValueFactory(new PropertyValueFactory<DataAlumn, String>("tFin"));
    calificacionFinal.setCellValueFactory(new PropertyValueFactory<DataAlumn, String>("total"));
    extra.setCellValueFactory(new PropertyValueFactory<DataAlumn, String>("extra"));
    inas.setCellValueFactory(new PropertyValueFactory<DataAlumn, String>("inas"));

    materias = FXCollections.observableArrayList();
    tableMaterias.setItems(materias);

    fillTable();
}

```

Ilustración 28, Se inicializan columnas de una tabla en javaFX

Esta parte fue en las que más batalló el equipo ya que aquí se dice en que columna se va a ingresar que atributo de la lista de los alumnos que viene de la clase DataAlumn, como se puede observar todos los datos son de tipo String, cabe decir que aquí solo la iniciamos para decirle en que columna se mostrará que, ya para llenar la tabla es como sigue:

```
@FXML
public void fillTable(){
    DataBaseSQL db = new DataBaseSQL();
    List<HashMap<String, String>> data1;
    data1 = db.getAll("CALIFICACIONES", "MATRICULA_ALUM", matricula);

    for(int i = 0; i < data1.size(); i++){

        DataAlumn dtA = new DataAlumn();

        dtA.clave.set(data1.get(i).get("CLAVE_MATERIA"));
        dtA.p1.set(data1.get(i).get("CAL1"));
        dtA.p2.set(data1.get(i).get("CAL2"));
        dtA.p3.set(data1.get(i).get("CAL3"));

        float aux1 = (Float.parseFloat(data1.get(i).get("CAL1")) + Float.parseFloat(data1.get(i).get("CAL2")) + Float.parseFloat(data1.get(i).get("CAL3"))) / 3 * 0.6F;

        dtA.tP.set(String.valueOf(aux1));
        dtA.fin.set(data1.get(i).get("FINAL"));

        float aux2 = Float.parseFloat(data1.get(i).get("FINAL")) * 0.4F;

        dtA.tFin.set(String.valueOf(aux2));
        dtA.total.set(String.valueOf(aux1 + aux2));
        dtA.extra.set(data1.get(i).get("EXTRA"));
        dtA.inas.set(data1.get(i).get("INASISTENCIAS"));

        materias.add(dtA);
    }
}
```

Ilustración 29, Se llena la tabla con los valores específicos

Capítulo 7: Calificaciones

Es muy parecida a la clase `DataAlumn` pero esta sirve para la aplicación de profesor para insertar los datos en la tabla donde el profesor podrá subir las calificaciones y las faltas de los alumnos, porque aquí se muestran los atributos de las calificaciones usando la clase `SimpleStringProperty`

```
public SimpleStringProperty alumno = new SimpleStringProperty();
public SimpleStringProperty matricula = new SimpleStringProperty();
public SimpleStringProperty parcial1 = new SimpleStringProperty();
public SimpleStringProperty parcial2 = new SimpleStringProperty();
public SimpleStringProperty parcial3 = new SimpleStringProperty();
public SimpleStringProperty parcialFinal = new SimpleStringProperty();
public SimpleStringProperty extra = new SimpleStringProperty();
public SimpleStringProperty faltas = new SimpleStringProperty();
```

Ya sabemos que debe de llevar su respectivo `get`, por eso ya no pondremos esa parte, que se explicó con `DataAlumn`.

Capítulo 8: MainProfesor

Al iniciar nuestra aplicación de profesor se muestran los datos de que profesor es, y esto se puede realizar con el siguiente código:

```
public void buscarDatos(){
    DataBaseSQL db = new DataBaseSQL();
    HashMap<String, String> data;
    data = db.fetchArray("PROFESOR", "CLAVE", claveProfesor);

    bnomP.setText(data.get("NOMBRE"));
    bclaveP.setText(data.get("CLAVE"));
    brfcP.setText(data.get("RFC"));
    bdirP.setText(data.get("DIRECCION"));
    btelP.setText(data.get("TEL"));
    bemailP.setText(data.get("EMAIL"));
}
```

Capítulo 9: Validación

28

En esta clase se hacen todas las validaciones con la base de datos, en checar que se estén introduciendo datos válidos y lógicos.

Capítulo 10: Alumnos, Grupos, Materias, Profesores y Administradores

Estas clases se utilizan para las tablas de administrador en búsquedas generales que se puedan buscar todos los datos.

Diagrama UML:

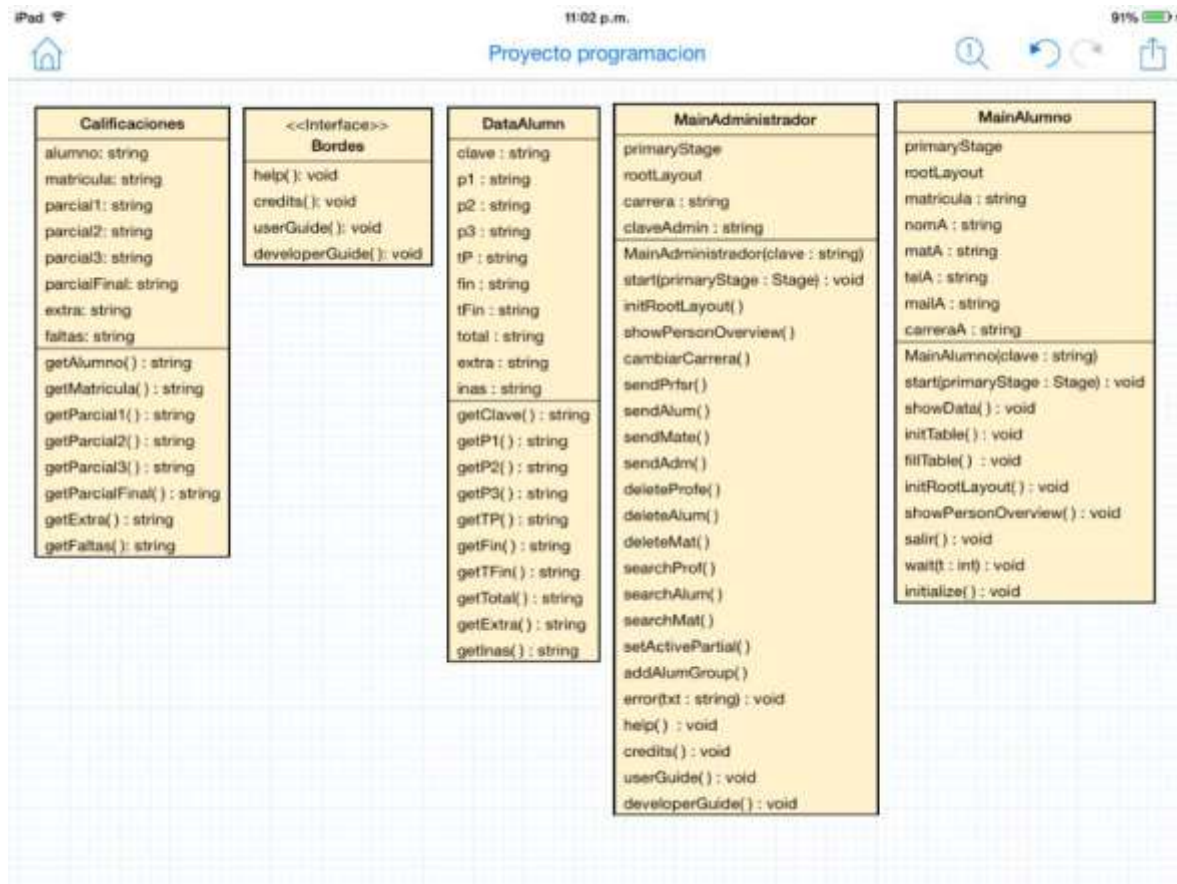


Ilustración 31, Diagrama UML

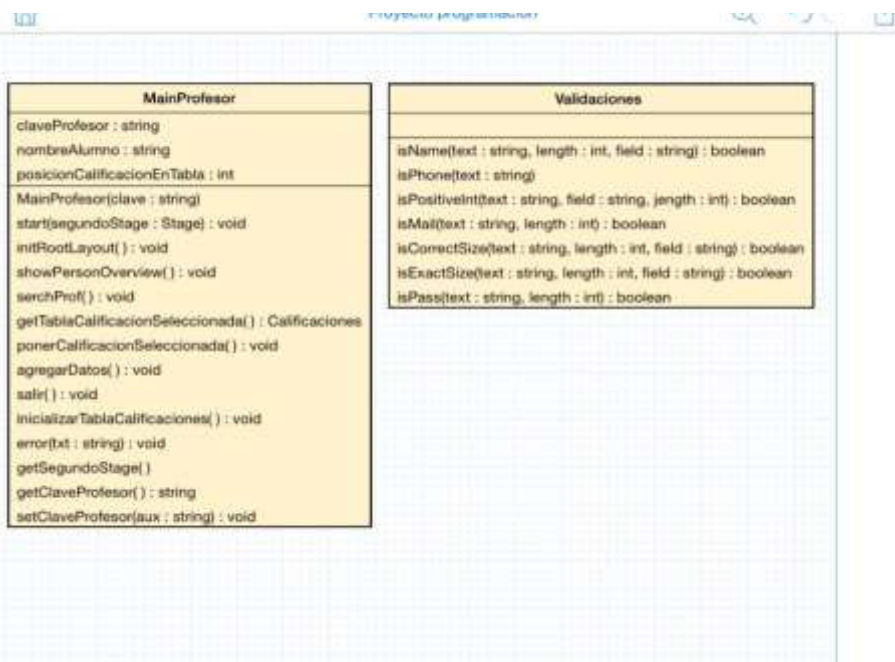


Ilustración 30, Diagrama UML imagen 2



Ilustración 32, Diagrama UML imagen 3

Tabla de ilustraciones

| | |
|--|----|
| Ilustración 1, Análisis Intranet | 5 |
| Ilustración 2, Objetivos Intranet | 6 |
| Ilustración 3, Ideología Intranet..... | 7 |
| Ilustración 4,Control de versiones | 8 |
| Ilustración 5, Logo Git | 9 |
| Ilustración 6, Logo GitHub..... | 9 |
| Ilustración 7, Modelo Vista Controlador..... | 10 |
| Ilustración 8, JavaFX | 11 |
| Ilustración 9, Configurar eclipse paso 1 | 12 |
| Ilustración 10, Configuración javaFX..... | 13 |
| Ilustración 11, Configuración eclipse | 13 |
| Ilustración 12, Archivo FXML..... | 14 |
| Ilustración 13, Crear un Anchor | 14 |
| Ilustración 14, Conexión Base de datos | 15 |
| Ilustración 15, Metodo importante para BD..... | 16 |
| Ilustración 16, Metodo para insertar datos en la BD | 17 |
| Ilustración 17, Interfaz de login | 18 |
| Ilustración 18, Se hace la conexión entre FXML y el controlador | 19 |
| Ilustración 19, Dependiendo del radioButton se inicializa una aplicación | 19 |
| Ilustración 20, Interfaz para registrar Profesor..... | 20 |
| Ilustración 21, Código para registrar a un profesor | 21 |
| Ilustración 22, Interfaz para eliminar un registro | 22 |
| Ilustración 23, Búsqueda específica | 22 |

| | | |
|--|----|----|
| Ilustración 24, Código para buscar a un profesor y agregarlo | 23 | |
| Ilustración 25, Clase SimpleStringProperty para realizar la tabla de javaFX..... | 23 | 31 |
| Ilustración 26, gets necesarios para javaFX | 24 | |
| Ilustración 27, FXML a controlador | 25 | |
| Ilustración 28, Se inicializan columnas de una tabla en javaFX | 25 | |
| Ilustración 29, Se llena la tabla con los valores específicos | 26 | |
| Ilustración 30, Diagrama UML imagen 2 | 29 | |
| Ilustración 31, Diagrama UML..... | 29 | |
| Ilustración 32, Diagrama UML imagen 3 | 30 | |

Bibliografía

Ceballos. (2005). *Java 2*. México: AlfaOmega.

López, J. E. (2002). *Java para todos*. México: AlfaOmega.