

**UNIVERSIDAD VERACRUZANA**

**FACULTAD DE ESTADÍSTICA E INFORMÁTICA**

**PROYECTO DE PRACTICAS PROFESIONALES**

**PROYECTO PRACTICAS PROFESIONALES**

**PRESENTA:**

**ALARCÓN SANTOS EMILIO ANTONIO**

**BALTAZAR ISLAS OMAR**

**RUÍZ ALARCÓN RICARDO**

**PROFESOR DE EXPERENCIA EDUCATIVA:**

**MTRO. CASTAÑEDA SANCHEZ FREDY**

Tabla de contenido

[1. Definición del estándar de codificación 3](#_Toc45642773)

[2. Revisión de código apegado al estándar 5](#_Toc45642774)

[3. Entrega de modelos de diseño revisados por el profesor de principios de diseño de software 6](#_Toc45642775)

[CU07 Asignar proyecto a estudiante 7](#_Toc45642776)

[CU09 Consultar progreso 9](#_Toc45642777)

[CU10 Entregar reporte 11](#_Toc45642778)

[4. Modelo de datos 13](#_Toc45642779)

[5. Capa de persistencia de datos 15](#_Toc45642780)

[6. Revisión de código apegada a prácticas de programación defensiva 17](#_Toc45642781)

[7. Definición de pruebas unitarias del modulo 18](#_Toc45642782)

[8. Reporte de pruebas unitarias 22](#_Toc45642783)

[9. Documentación del api generada apegada al estándar 23](#_Toc45642784)

[10. Definición de la UI 29](#_Toc45642785)

[11. Conclusiones 31](#_Toc45642786)

[12. Referencias 32](#_Toc45642787)

# Definición del estándar de codificación

|  |  |
| --- | --- |
| Proposito | Para guiar la implementación de programas Java |
| Encabezados | Todos los programas comienzan con un encabezado descriptivo. |
| Formato de encabezados | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* Asignación de programa: El número de programa \*/  /\* Nombre: Nuestro nombres \*/  /\* Fecha: Fecha en que se empezó a desarrollar el programa \*/  /\* Descripción: Una breve descripción de lo que hace el programa \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| Lista de contenidos | Proporciona un resumen de los contenidos de la lista |
| Ejemplo de lista de contenidos | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* Lista de contenidos: \*/  /\* Instrucciones de reutilización \*/  /\* Instrucciones de modificación \*/  /\* Includes \*/  /\* Declaracíón de clases: \*/  /\* CData \*/  /\* ASet \*/  /\* Código fuente en c:/classes/CData.java: \*/  /\* CData \*/  /\* CData() \*/  /\* Empty() \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| Instrucciones de reutilización | * Describe cómo se usa el programa: formato de declaración, valores de parámetros, tipos y formatos. * Proporciona advertencias de valores ilegales, condiciones de desbordamiento u otras condiciones que podrían resultar en una operación incorrecta. |
| Ejemplos de instrucciones de reutilización | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* Instrucciones de reutilización: \*/  /\* int PrintLine(char \*line\_of\_character) \*/  /\* Proposito:imprimir un string, ‘line\_of\_character’, en una linea \*/  /\* Limitaciones: la longitude de la linea no debe exceder LINE\_LENGTH \*/  /\* Devolver 0 si el impresor no esta listo para imprimir, sino 1 \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| Identificadores | Se utilizan nombres descriptivos para todas las variables, nombres de funciones, constantes y otros identificadores. Se evitan las abreviaturas o las variables de una letra. |
| Ejemplo de identificadores | Int numeroDeEstduiantes; /\* Esto es BUENO \*/  Float: x4, j, ftave; /\* Esto es MALO \*/ |
| Comentarios | * Se documenta el código para que el lector pueda entender su funcionamiento. * Los comentarios explican tanto el propósito como el comportamiento del código. * Se comentaron declaraciones de variables para indicar su propósito. |
| Buen comentario | If(contadorDeRegistros > limit) /\* ¿Todos los registros deben ser procesados? \*/ |
| Mal comentario | If(contadorDeRegistros > limit) /\* Verifica si el Contador de registros excede el limite \*/ |
| Sección principal | Se preceden las secciones principales del programa mediante un comentario de bloque que describe el procesamiento realizado en la siguiente sección. |
| Ejemplo | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* La sección del programa examina el contenido de la matriz "calificaciones" \*/  /\* y calcula la calificación promedio de la clase. \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| Espacios en blanco | * Se escriben programas con suficiente espacio para que no aparezcan abarrotados. * Se separa cada construcción de programa con al menos un espacio. |
| Identación | * Una tabulación comprende cuatro espacios en blanco y es equivalente a una sangria. * Sangría cada nivel de llave del nivel anterior. * Las llaves de cierre deben estar en líneas por sí mismas y alineadas. |
| Ejemplo de identación | while (distanciaFaltante > limite) {  codigoExitoso = mueveRobot (lugarSeleccionado);  if (codigoExitoso == movimientoFallido) {  printf(“El movimiento del robot ha fallado.\n”);  }  } |
| Capitalización | * Se capitalizaron todas las definiciones. * Minúsculas todos los demás identificadores y palabras reservadas. |
| Capitalization Examples | #define DEFAULT-NUMERO-DE-ESTUDIANTES 15  int tamañoClase = DEFAULT-NUMERO-DE-ESTUDIANTES; |

# Revisión de código apegado al estándar

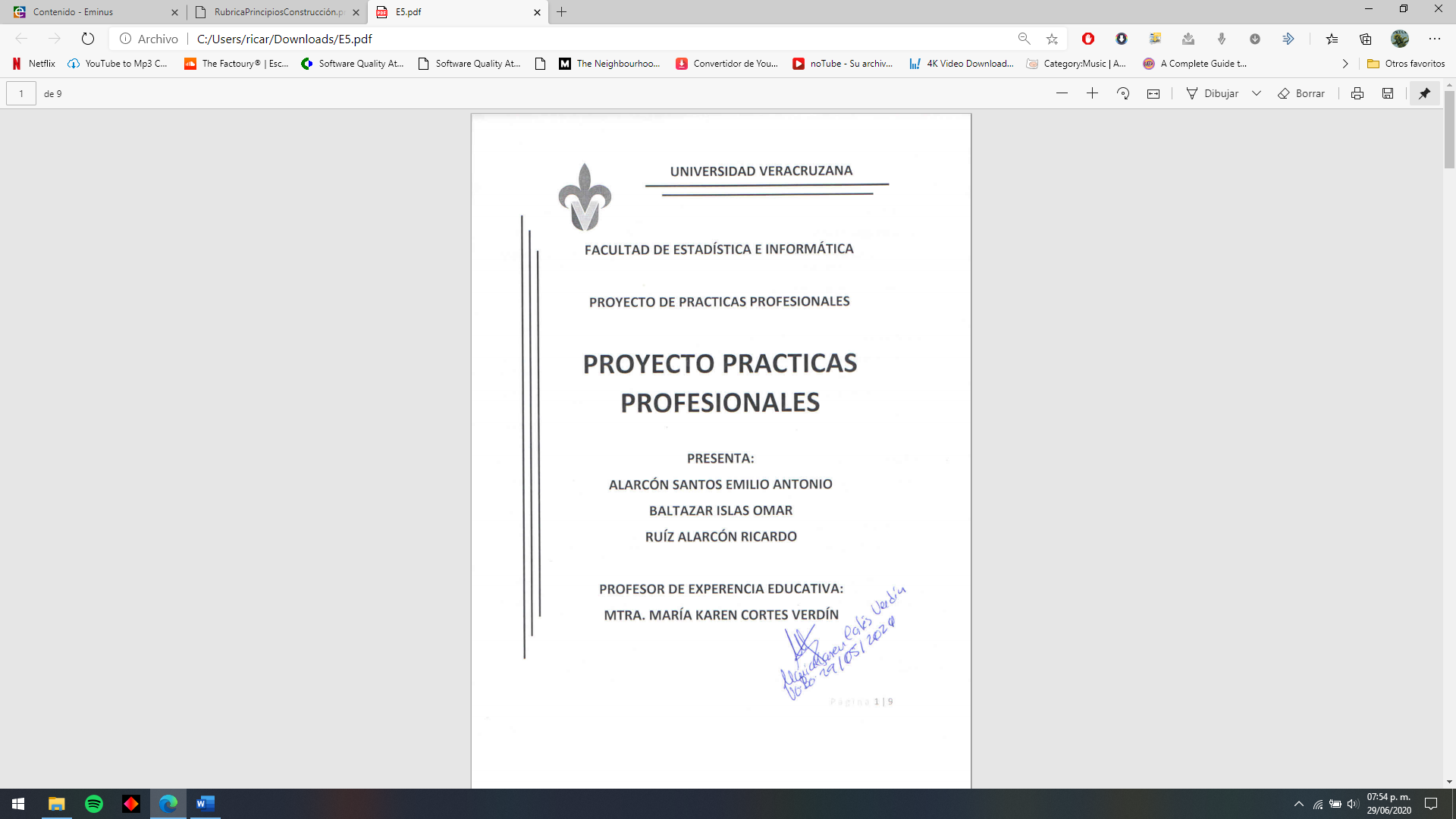
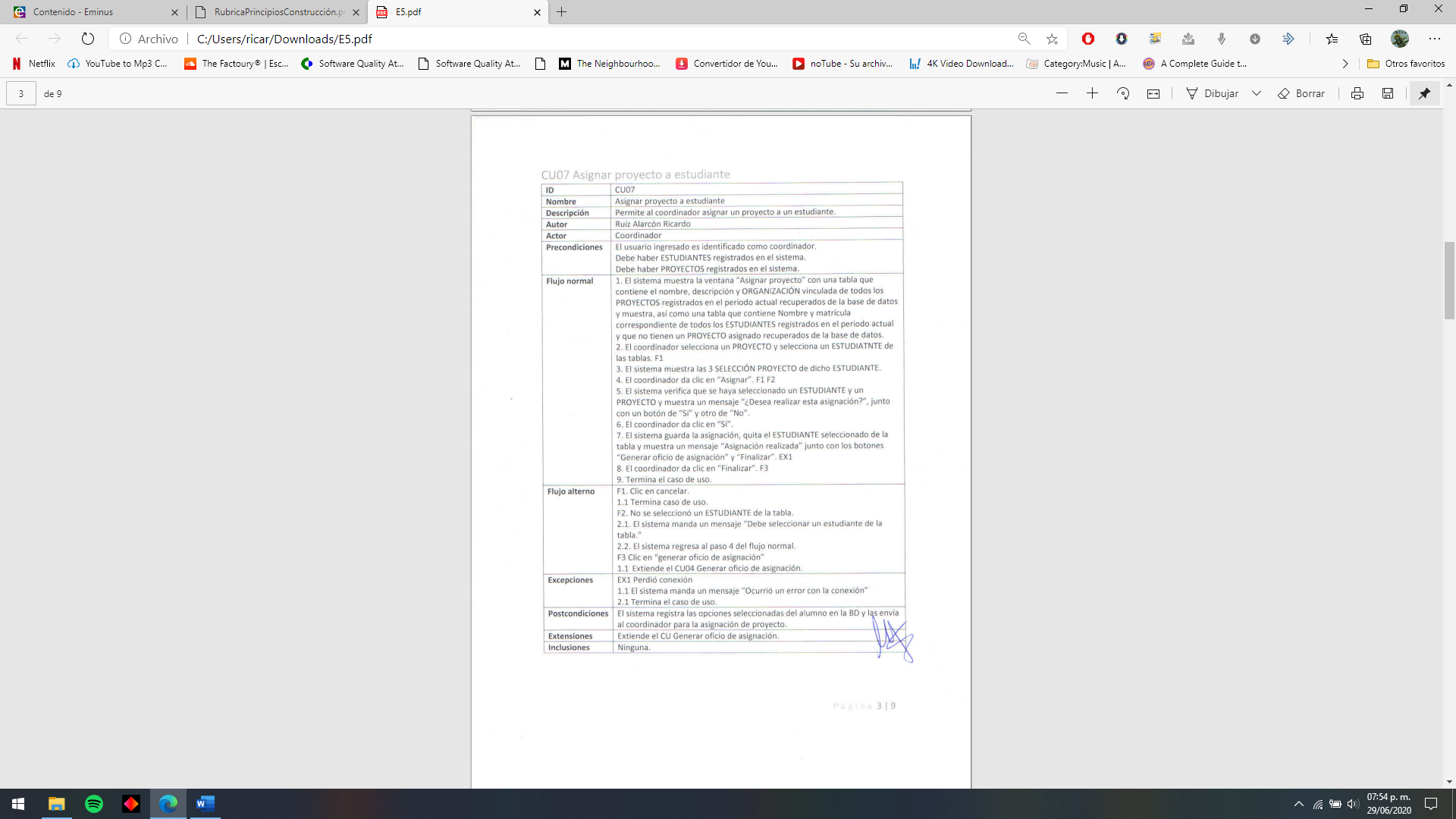
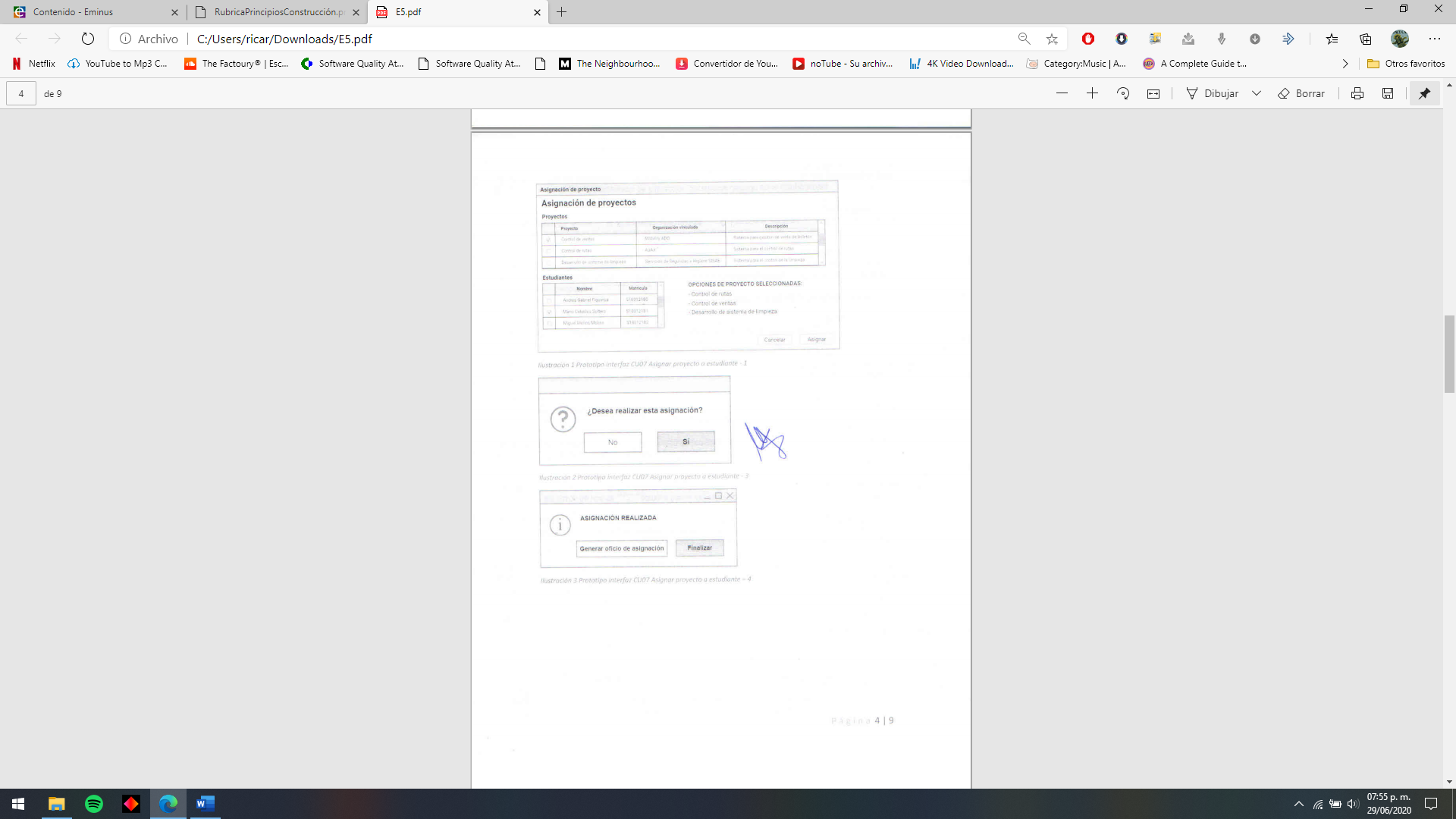
|  |  |  |
| --- | --- | --- |
| **Elemento** | **Contenido Semántico** | **Puntaje final** |
| 1. ENCABEZADOS | El main del programa comienza con un encabezado descriptivo de lo que hace el sistema/programa. |  |
| 1. LISTA DE CONTENIDOS | El main del programa contiene un resumen de los contenidos de la lista. |  |
| 1. INSTRUCCIONES DE REUTILIZACIÓN | La sección de instrucciones de reutilización en la lista de contenido describe como se usa el programa y proporciona advertencias de valores ilegales, condiciones u especificaciones que podrían resultar en una operación incorrecta. |  |
| 1. IDENTIFICADORES | Las variables, nombres de funciones, constantes y otros identificadores utilizan nombres descriptivos en notación camello. |  |
| 1. COMENTARIOS | El código esta comentado para poder entender el funcionamiento explicando el propósito y comportamiento del código. |  |
| 1. SECCIÓN PRINCIPAL | Todas las secciones de cada clase comienzan con un bloque comentado que describe los procesamientos realizados. |  |
| 1. ESPACIOS EN BLANCO | El código esta separado con suficientes espacios y cada construcción del programa se separa con al menos un espacio. |  |
| 1. IDENTACIÓN | Se respetan los niveles de identación mencionada en el estándar |  |
| 1. CAPITALIZACIÓN | Se capitalizan todas las definiciones y las variables y métodos se escriben en minúsculas con notación camello. |  |

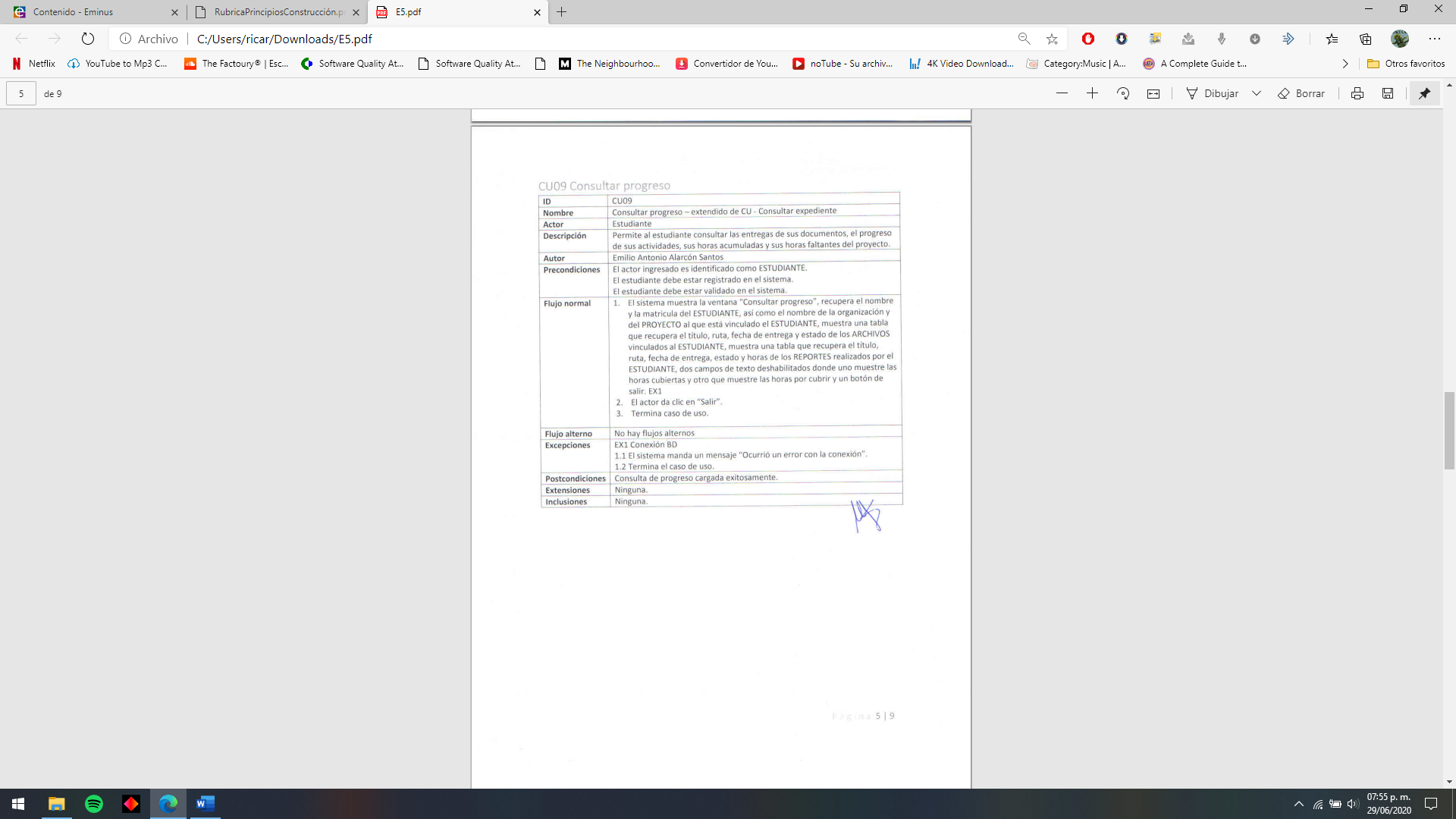
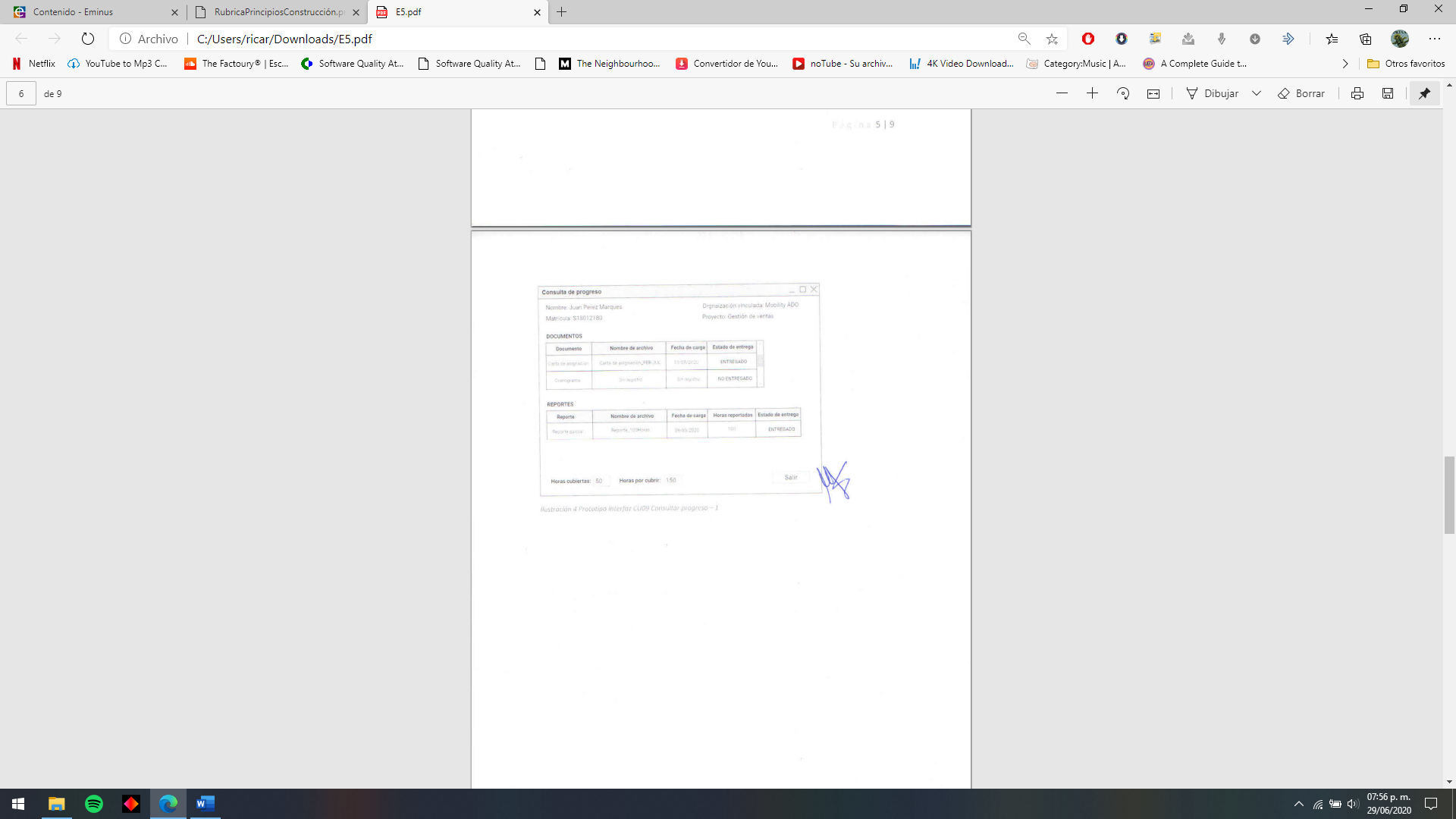
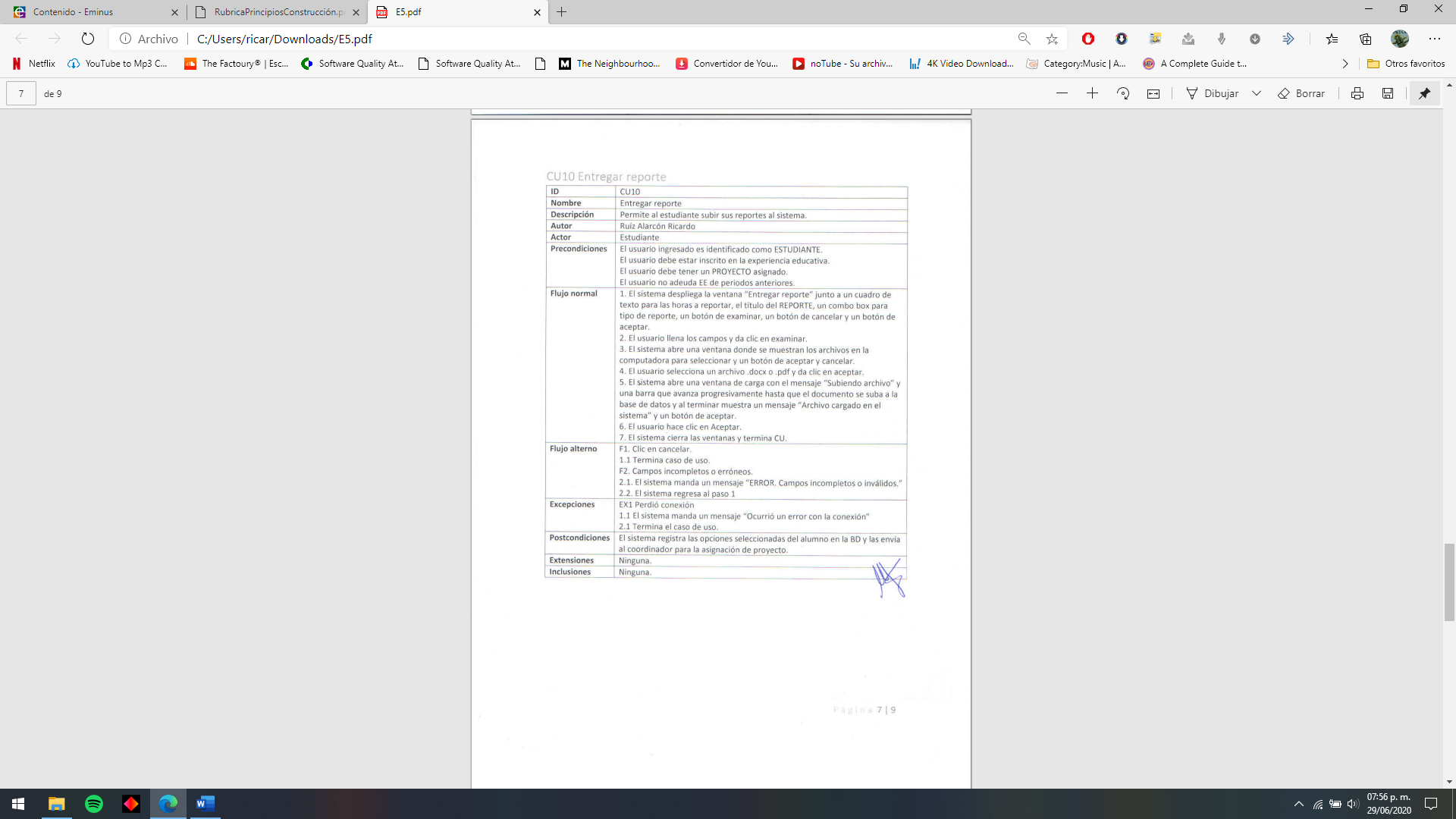
TOTAL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

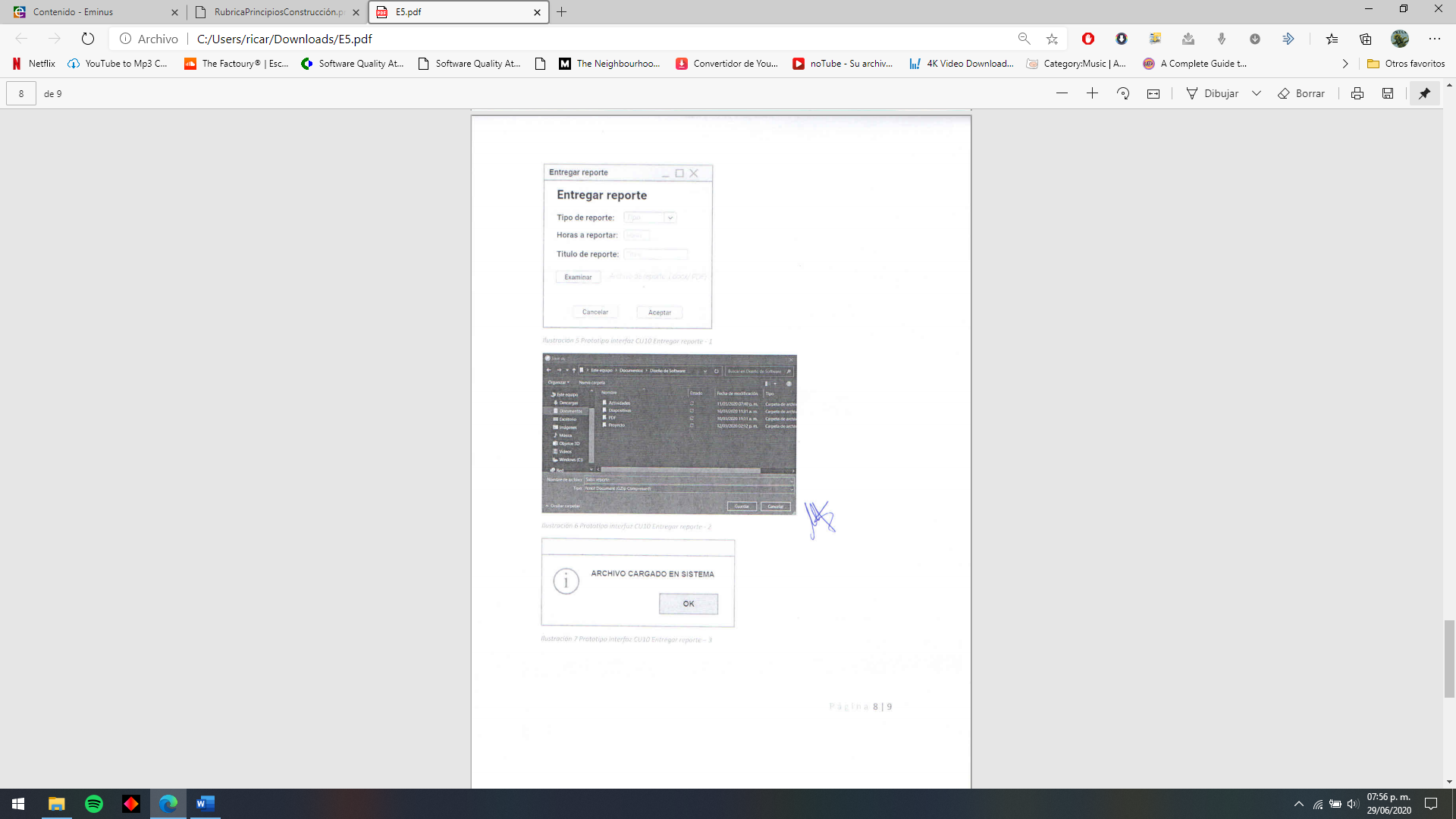
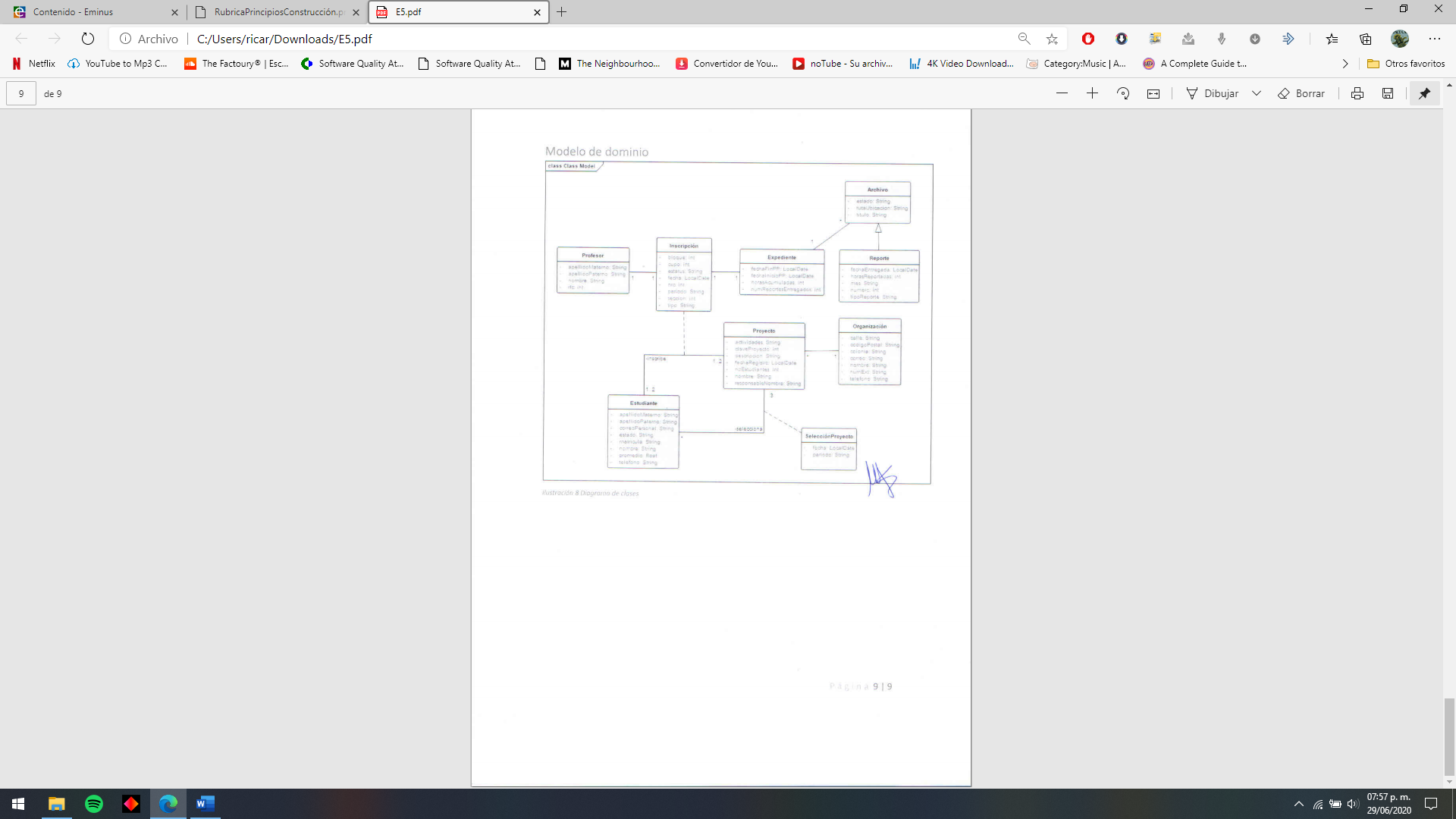
# Entrega de modelos de diseño revisados por el profesor de principios de diseño de software

El documento fue entregado a la Dra. María Karen Cortes, profesora de la experiencia educativa de principios de diseño de software, el cuál fue firmado el 29 de mayo del 2020, y fue entregado al Mtro. Fredy Castañeda en tiempo y forma. De requerir el documento PDF se puede solicitar a cualquier integrante del equipo.

Se anexan fotos de las paginas firmadas por la Dra. Karen del documento original.

## CU07 Asignar proyecto a estudiante

|  |  |
| --- | --- |
| **ID** | CU07 |
| **Nombre** | Asignar proyecto a estudiante |
| **Descripción** | Permite al coordinador asignar un proyecto a un estudiante. |
| **Autor** | Ruíz Alarcón Ricardo |
| **Actor** | Coordinador |
| **Precondiciones** | El usuario ingresado es identificado como coordinador.  Debe haber ESTUDIANTES registrados en el sistema.  Debe haber PROYECTOS registrados en el sistema. |
| **Flujo normal** | 1. El sistema muestra la ventana “Asignar proyecto” que recupera y muestra una tabla que contiene la clave, nombre, descripción y ORGANIZACIÓN vinculada de todos los PROYECTOS registrados en el periodo actual recuperados de la base de datos que no tengan un ESTUDIANTE asignado, y muestra, una tabla que contiene la matricula, nombre, apellido paterno y apellido materno correspondiente de todos los ESTUDIANTES registrados en el periodo actual y que no tienen un PROYECTO asignado recuperados de la base de datos.  2. El coordinador selecciona un PROYECTO y selecciona un ESTUDIANTE de la tabla. F1  3. El sistema recupera y muestra las 3 SELECCIÓN PROYECTO del ESTUDIANTE seleccionado.  4. El coordinador da clic en asignar. F1 F2  5. El sistema verifica que se haya seleccionado un ESTUDIANTE y un PROYECTO y muestra un mensaje “¿Desea realizar esta asignación?”, junto con un botón de “Sí” y otro de “No”.  6. El coordinador da clic en “Sí”. F3  7. El sistema registra la asignación generando así la nueva INSCRIPCION, y muestra un mensaje “Asignación realizada” junto con los botones “Generar oficio de asignación” y “Finalizar”. EX1  8. El coordinador da clic en “Finalizar”. F4  9. Termina el caso de uso. |
| **Flujo alterno** | F1. Clic en cancelar.  1.1 Termina caso de uso.  F2. No se seleccionó una opción de la tabla.  2.1. El sistema manda un mensaje “Debe realizar sus selecciones.”  2.2. El sistema regresa al paso 4 del flujo normal.  F3. Clic en no realizar asignación  3.1 El sistema regresa al paso 1 del flujo normal.  F4. Clic en “generar oficio de asignación”  4.1 Extiende el CU04 Generar oficio de asignación |
| **Excepciones** | EX1 Perdió conexión  1.1 El sistema manda un mensaje “Ocurrió un error con la conexión”  2.1 Termina el caso de uso. |
| **Postcondiciones** | El sistema registra las opciones seleccionadas del alumno en la BD y las envía al coordinador para la asignación de proyecto. |
| **Extensiones** | CU04 Generar oficio de asignación. |
| **Inclusiones** | Ninguna. |

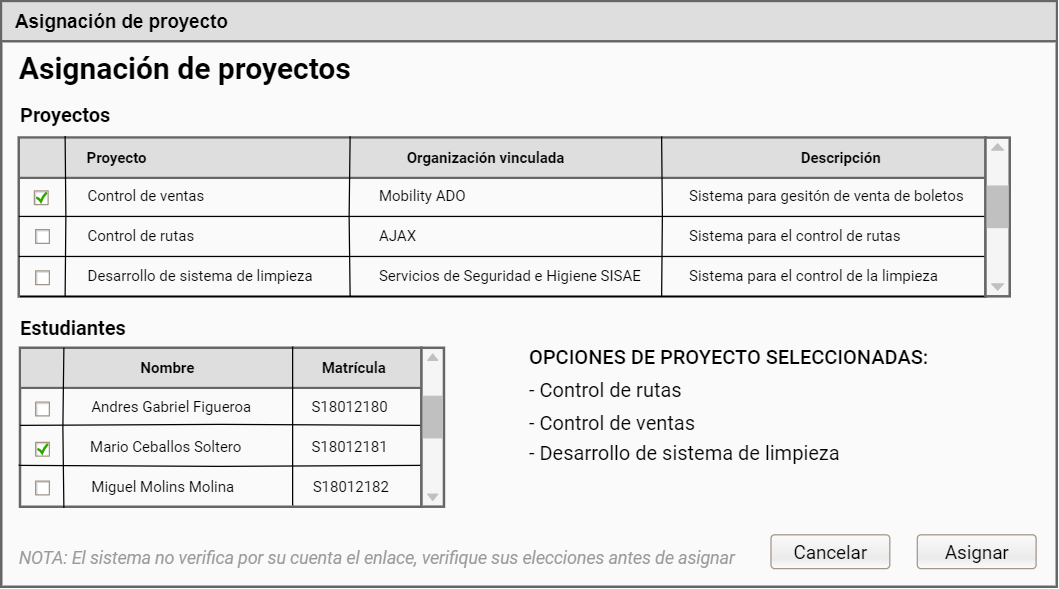


Ilustración 1 Prototipo interfaz CU07 Asignar proyecto a estudiante - 1

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración 2 Prototipo interfaz CU07 Asignar proyecto a estudiante - 2

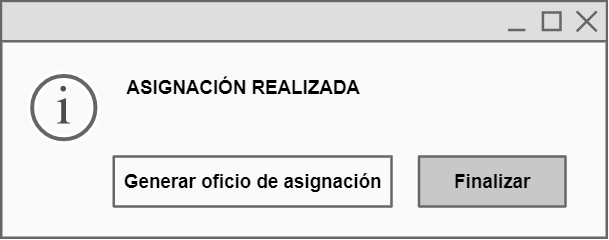
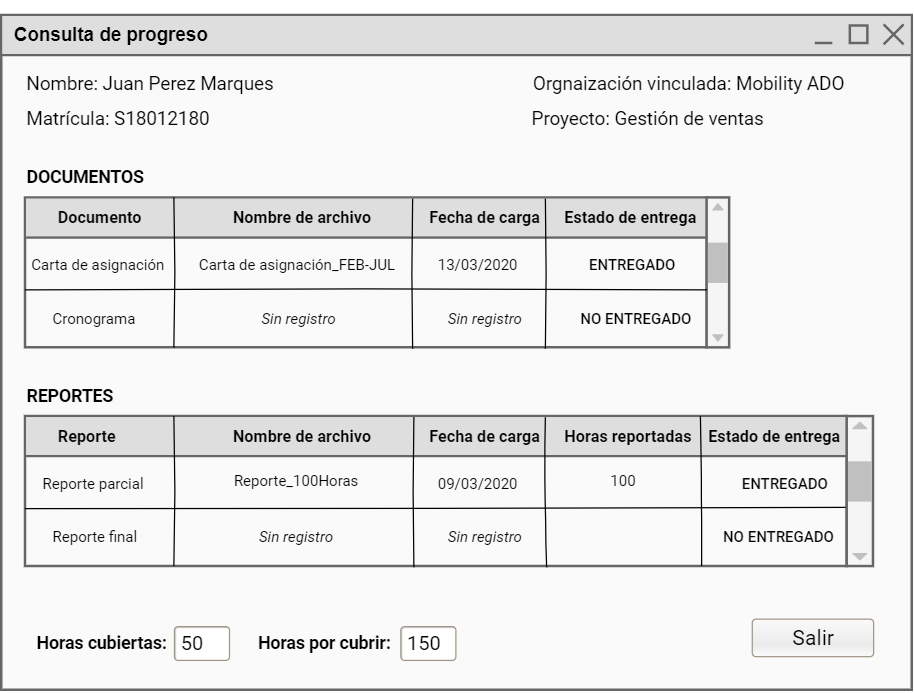


Ilustración 3 Prototipo interfaz CU07 Asignar proyecto a estudiante – 3

## CU09 Consultar progreso

|  |  |
| --- | --- |
| **ID** | CU09 |
| **Nombre** | Consultar progreso – extendido de CU - Consultar expediente |
| **Actor** | Estudiante |
| **Descripción** | Permite al ESTUDIANTE consultar las entregas de sus documentos, el progreso de sus actividades, sus horas acumuladas y sus horas faltantes del proyecto. |
| **Autor** | Emilio Antonio Alarcón Santos |
| **Precondiciones** | El actor ingresado es identificado como ESTUDIANTE.  El ESTUDIANTE debe estar registrado en el sistema.  El ESTUDIANTE debe estar validado en el sistema. |
| **Flujo normal** | 1. El sistema muestra la ventana “Consultar progreso”, recupera el nombre y la matricula del ESTUDIANTE, así como el nombre de la organización y del PROYECTO al que está vinculado el ESTUDIANTE, muestra una tabla que recupera el título, fecha de entrega y estado de los ARCHIVOS vinculados al ESTUDIANTE, muestra una tabla que recupera el título, ruta, fecha de entrega, estado y horas de los REPORTES realizados por el ESTUDIANTE, dos campos de texto deshabilitados donde uno muestre las horas cubiertas y otro que muestre las horas por cubrir y un botón de salir. EX1 2. El actor da clic en “Salir”. 3. Termina caso de uso. |
| **Flujo alterno** | No hay flujos alternos |
| **Excepciones** | EX1 Conexión BD  1.1 El sistema manda un mensaje “Ocurrió un error con la conexión”.  1.2 Termina el caso de uso. |
| **Postcondiciones** | Consulta de progreso cargada exitosamente. |
| **Extensiones** | Ninguna. |
| **Inclusiones** | Ninguna. |



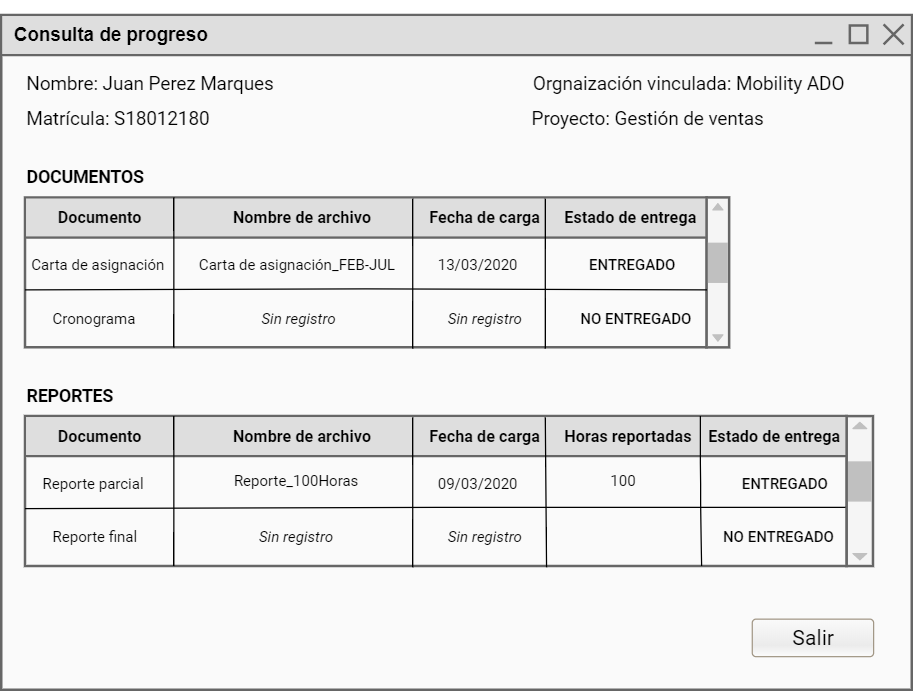
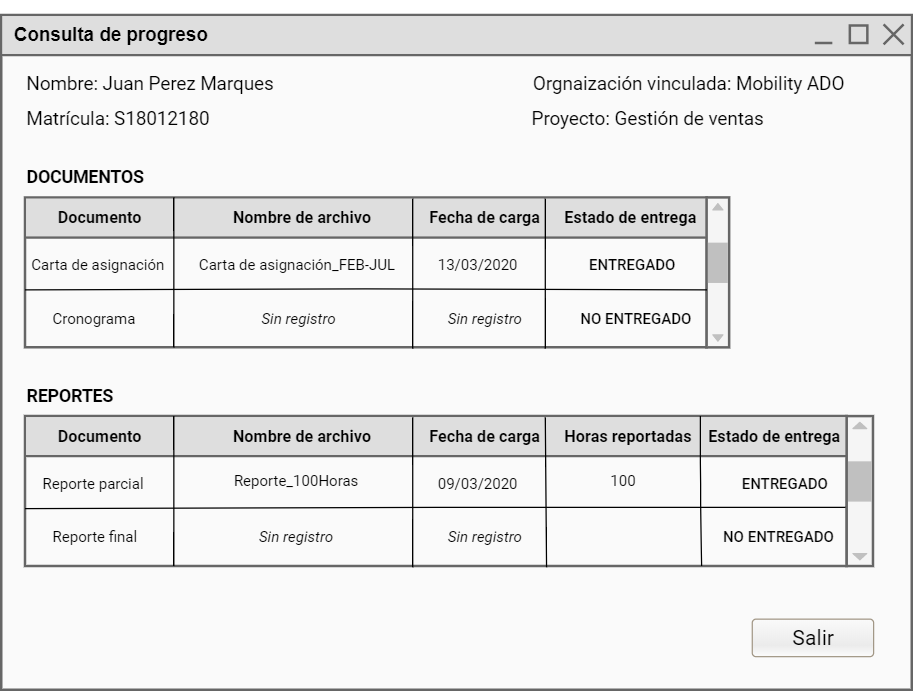
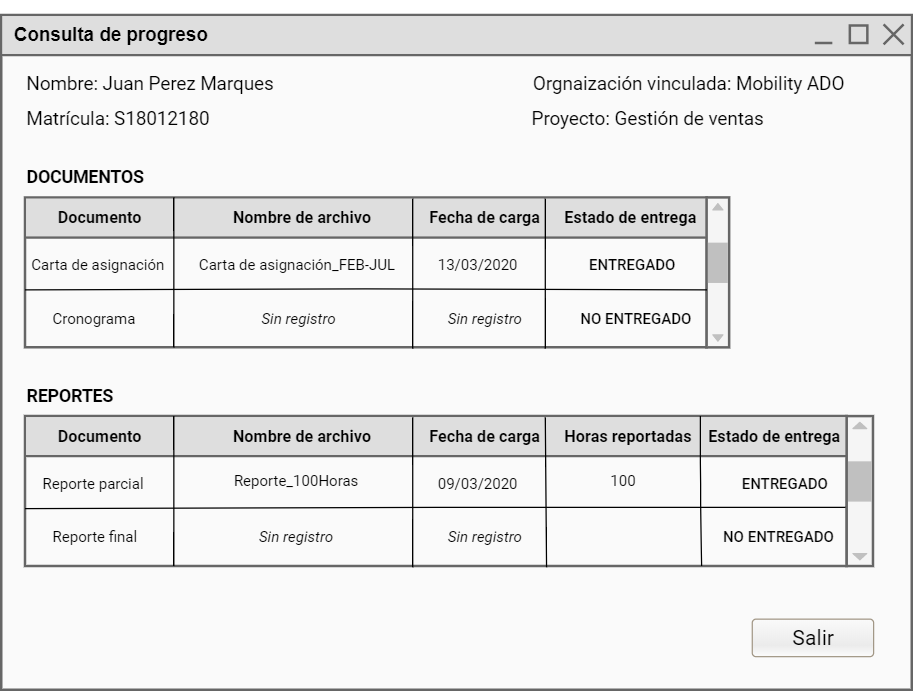


Ilustración 4 Prototipo interfaz CU09 Consultar progreso – 1

## CU10 Entregar reporte

|  |  |
| --- | --- |
| **ID** | CU10 |
| **Nombre** | Entregar reporte |
| **Descripción** | Permite al estudiante subir sus reportes al sistema. |
| **Autor** | Ruíz Alarcón Ricardo |
| **Actor** | Estudiante |
| **Precondiciones** | El usuario ingresado es identificado como ESTUDIANTE.  El usuario debe estar inscrito en la experiencia educativa.  El usuario debe tener un PROYECTO asignado.  El usuario no adeuda EE de periodos anteriores. |
| **Flujo normal** | 1. El sistema despliega la ventana “Entregar reporte” donde recupera la fecha actual del sistema junto a un cuadro de texto para las horas a reportar, un combo box para tipo de reporte, un botón de examinar, un botón de cancelar y un botón de aceptar.  2. El usuario llena los campos y da clic en examinar. F1  3. El sistema abre una ventana donde se muestran los archivos en la computadora para seleccionar y un botón de aceptar y cancelar.  4. El usuario selecciona un archivo .docx o .pdf y da clic en aceptar. F2  5. El sistema verifica los campos y el tipo del archivo, registra el documento en la base de datos y muestra un mensaje “Archivo cargado en el sistema” y un botón de aceptar. EX1 F3  6. El usuario hace clic en Aceptar.  7. El sistema cierra las ventanas y termina CU. |
| **Flujo alterno** | F1. Clic en cancelar.  1.1 Termina caso de uso.  F2. Clic cancelar (Explorador de archivos).  2.1 El sistema cierra el explorador de archivos.  2.2 El sistema regresa al paso 1 del flujo normal.  F3. Campos incompletos o erróneos.  3.1. El sistema manda un mensaje “ERROR. Campos incompletos o inválidos.”  3.2. El sistema regresa al paso 1 |
| **Excepciones** | EX1 Perdió conexión  1.1 El sistema manda un mensaje “Ocurrió un error con la conexión”  2.1 Termina el caso de uso. |
| **Postcondiciones** | El sistema registra las opciones seleccionadas del alumno en la BD y las envía al coordinador para la asignación de proyecto. |
| **Extensiones** | Ninguna. |
| **Inclusiones** | Ninguna. |



Ilustración 5 Prototipo interfaz CU10 Entregar reporte - 1

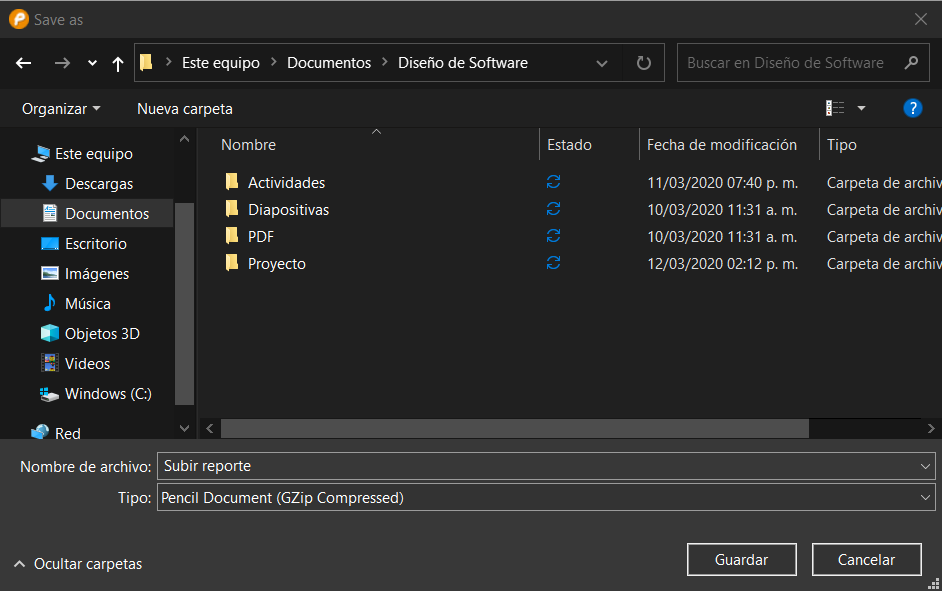


Ilustración 6 Prototipo interfaz CU10 Entregar reporte - 2

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración 7 Prototipo interfaz CU10 Entregar reporte - 3

# Modelo de datos

A continuación, se presentan los siguientes diagramas los cuales representan el modelo de dominio con el diagrama de clases y un modelo de datos que representa como se conforma dicho dominio en la base de datos.



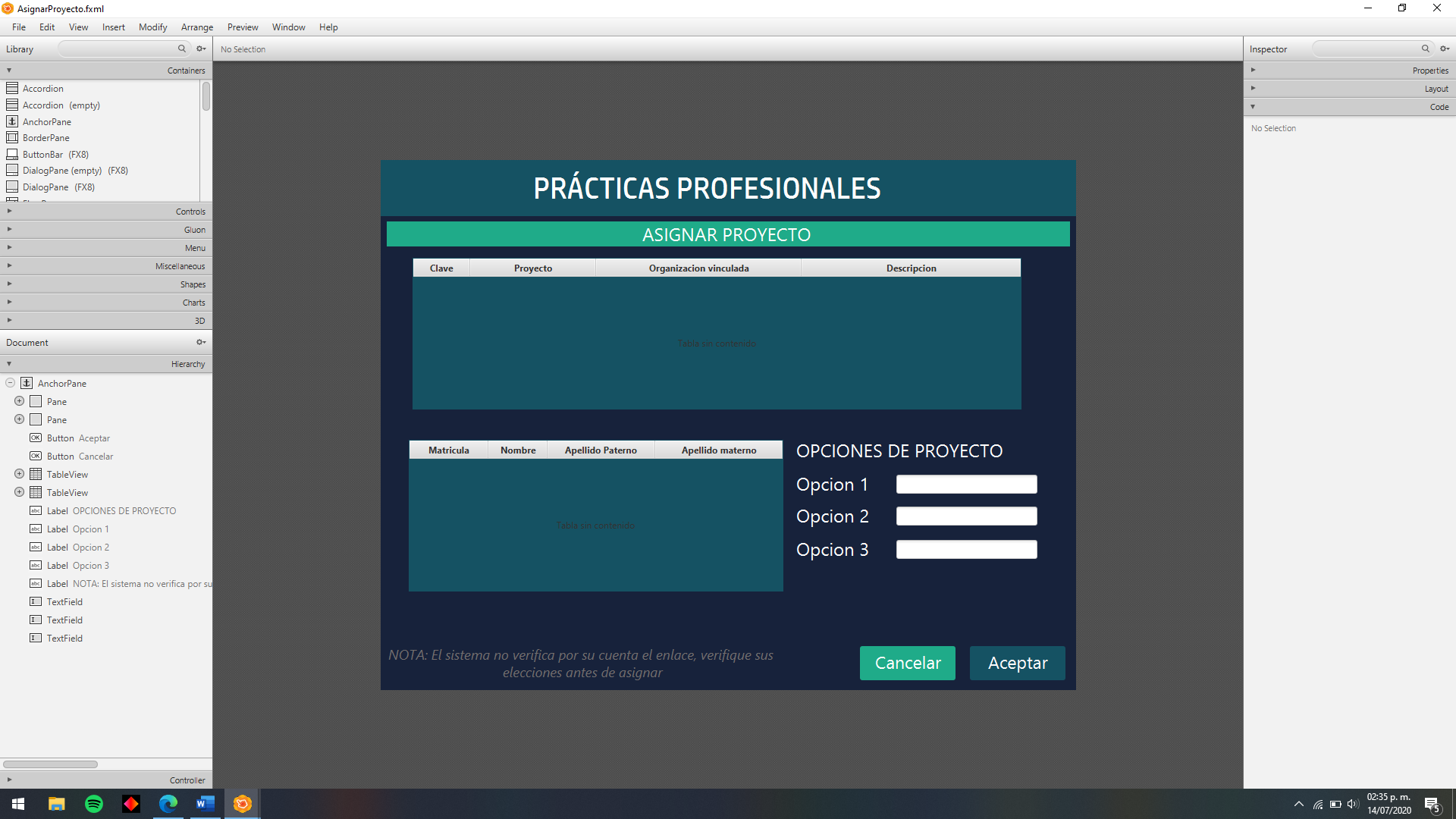
Ilustración 8 Diagrama de clases

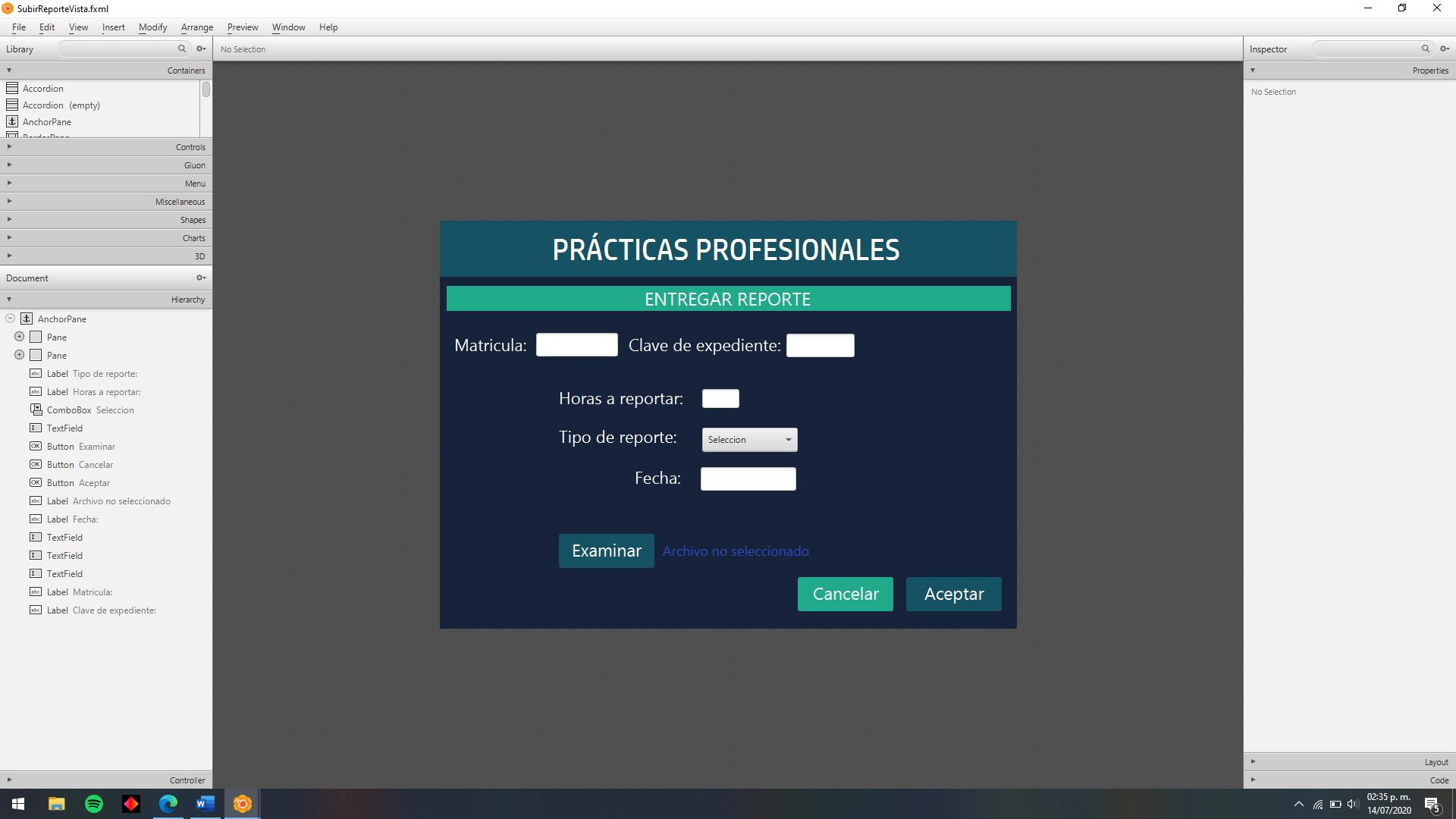


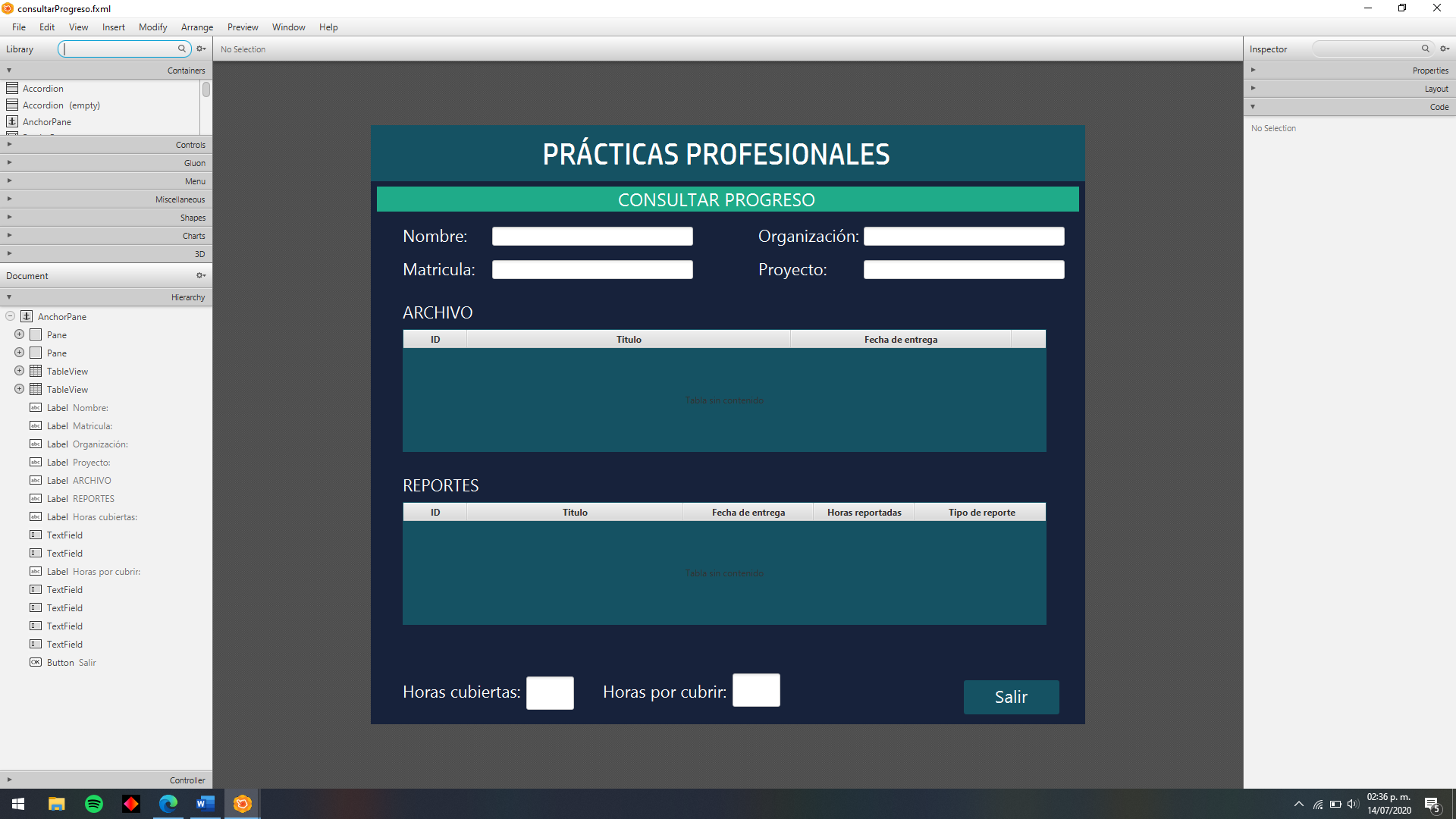
Ilustración 9 Modelo de datos

# Capa de persistencia de datos

Las siguientes imágenes representan las interfaces que forman parte de la implementación, las cuales están apegadas al estándar y cumplen de acuerdo con el diseño de prototipos de CU.







# Revisión de código apegada a prácticas de programación defensiva

Esta sección del documento muestra un fragmento del código presentado en el proyecto, el cual sostiene los principios que se usan en las prácticas de programación defensiva. Cabe destacar que este es solo un método ejemplo, ya que todos los métodos cuentan con la misma clase de prácticas.

Como se puede notar, al final tiene un cierre de recursos de rs, stm y con con el fin de verificar y atrapar excepciones.

/\*\*

\* Recupera de la base de datos la lista de SELECCIONPROYECTO de un

\* ESTUDIANTE especificado registrado en la BD.

\*

\* @param matricula Matricula del ESTUDIANTE del cual se recuperaran las

\* SELECCIONPROYECTO.

\* @return obs Lista contenedora de las SELECCIONPROYECTO.

\*/

public ArrayList<SeleccionProyectoPOJO> getSelecciones(String matricula) throws Exception{

Connection con = null;

Statement stm = null;

ResultSet rs = null;

String sql = "select claveproyecto, fecha, periodo from "

+ "seleccionproyecto where matricula = '" + matricula + "';";

ArrayList<SeleccionProyectoPOJO> obs

= new ArrayList<SeleccionProyectoPOJO>();

try {

con = new ConexionDB().conectarMySQL();

stm = con.createStatement();

rs = stm.executeQuery(sql);

while (rs.next()) {

int claveproyecto = rs.getInt("claveproyecto");

LocalDate fecha = LocalDate.parse(rs.getString("fecha"),

DateTimeFormatter.ofPattern("yyyy-MM-dd"));

String periodo = rs.getString("periodo");

SeleccionProyectoPOJO c = new SeleccionProyectoPOJO(

claveproyecto, fecha, periodo);

obs.add(c);

}

} catch (SQLException e) {

throw new Exception("Error en Clase EstudianteDAO, método "

+ "getSelecciones: " + e.getMessage());

}finally{

try { if (rs != null) rs.close(); } catch (Exception e) {};

try { if (stm != null) stm.close(); } catch (Exception e) {};

try { if (con!= null) con.close(); } catch (Exception e) {};

}

return obs;

}

# Definición de pruebas unitarias del modulo

A continuación, se muestra el código que forma parte de las pruebas unitarias que demuestran las funcionalidades de todos los rubros del sistema que conforma el proyecto.

package modelo;

import javafx.collections.ObservableList;

import org.junit.Test;

import static org.junit.Assert.\*;

import org.junit.Before;

public class ArchivoDAO\_Test {

ArchivoDAO archivoDAO = null;

ArchivoPOJO archivo = null;

@Before

public void before(){

archivoDAO = new ArchivoDAO();

}

@Test

public void recuperarArchivosTest(){

ObservableList<ArchivoPOJO> archivos = archivoDAO.getArchivos("S18012183");

assertTrue(archivos.isEmpty());

}

@Test

public void recuerarMaxIDTest(){

int clave = archivoDAO.obtenerClaveArchivo();

assertTrue(clave == 3);

}

}

-----------------------------------------------------------------------------

package modelo;

import java.util.ArrayList;

import javafx.collections.ObservableList;

import org.junit.Test;

import static org.junit.Assert.\*;

import org.junit.Before;

public class EstudianteDAO\_Test {

EstudianteDAO estudianteDAO = null;

EstudiantePOJO estudiante = null;

@Before

public void before(){

estudianteDAO = new EstudianteDAO();

}

@Test

public void recuperarEstudianteTest(){

estudiante = estudianteDAO.recuperar("S18012180");

assertTrue(estudiante.getNombre().equalsIgnoreCase("Omar"));

}

@Test

public void recuperarNombreEstudianteTest(){

String nombre = estudianteDAO.recuperarNombreEstudiante("S18012180");

assertTrue(nombre.equalsIgnoreCase("Omar"));

}

@Test

public void recuperarNombreOrganizacionTest(){

String nombreOrganizacion = estudianteDAO.recuperarNombreOrganizacion("S18012183");

assertTrue(nombreOrganizacion.equalsIgnoreCase("Telmex"));

}

@Test

public void recuperarNombreProyectoTest(){

String nombreProyecto = estudianteDAO.recuperarNombreProyecto("S18012183");

assertTrue(nombreProyecto.equalsIgnoreCase("PagoMex"));

}

@Test

public void recuperarClaveProyectoTest(){

int claveProyecto = estudianteDAO.recuperarClaveProyecto("S18012183");

assertTrue(claveProyecto == 912);

}

@Test

public void rcuperarClaveExpedienteTest(){

int claveExpediente = estudianteDAO.recuperaClaveExpediente("S18012183");

assertTrue(claveExpediente == 1);

}

@Test

public void asignarProyectoTest(){

estudianteDAO.asginarProyecto("S18012180", 811);

int claveExpediente = estudianteDAO.recuperaClaveExpediente("S18012180");

assertTrue(claveExpediente == 3);

}

@Test

public void recuperarEstudiantesTest(){

ObservableList<EstudiantePOJO> estudiantes = estudianteDAO.getEstudiantes();

estudiante = estudiantes.get(0);

assertTrue(estudiante.getNombre().equalsIgnoreCase("Emilio Antonio"));

}

@Test

public void reuperarselecciones(){

ArrayList<SeleccionProyectoPOJO> selecciones = estudianteDAO.getSelecciones("S18012180");

SeleccionProyectoPOJO seleccion = selecciones.get(0);

assertTrue(seleccion.getClaveProyecto() == 711);

}

}

-----------------------------------------------------------------------------package modelo;

import javafx.collections.ObservableList;

import static org.junit.Assert.assertTrue;

import org.junit.Before;

import org.junit.Test;

public class ProyectoDAO\_Test{

ProyectoDAO proyectoDAO;

ProyectoPOJO proyecto;

@Before

public void before(){

proyectoDAO = new ProyectoDAO();

}

@Test

public void testRecuperarNombre(){

String nombre = proyectoDAO.recuperarNombre(712);

assertTrue(nombre.equalsIgnoreCase("SecureServ"));

}

@Test

public void testRecuperarNombreOrganizacion(){

String nombreOrganizacion = proyectoDAO.recuperarNombreOrganizacion(712);

assertTrue(nombreOrganizacion.equalsIgnoreCase("Softeck"));

}

@Test

public void recuperarProyectosTest(){

ObservableList<ProyectoPOJO> proyectos = proyectoDAO.getProyectos();

proyecto = proyectos.get(0);

assertTrue(proyecto.getNombre().equalsIgnoreCase("ProdutivityService"));

}

}

-----------------------------------------------------------------------------

package modelo;

import javafx.collections.ObservableList;

import org.junit.Test;

import static org.junit.Assert.\*;

import org.junit.Before;

public class ReporteDAO\_Test {

ReporteDAO reporteDAO = null;

ReportePOJO reporte = null;

@Before

public void before(){

reporteDAO = new ReporteDAO();

}

@Test

public void recuperarReportesTest(){

ObservableList<ReportePOJO> reportes = reporteDAO.getReportes("S18012183");

reporte = reportes.get(0);

assertTrue(reporte.getTipoReporte().equalsIgnoreCase("Inicial"));

}

@Test

public void recuperarHorasTest(){

int horasAcomuladas = reporteDAO.recuperarHoras("S18012183");

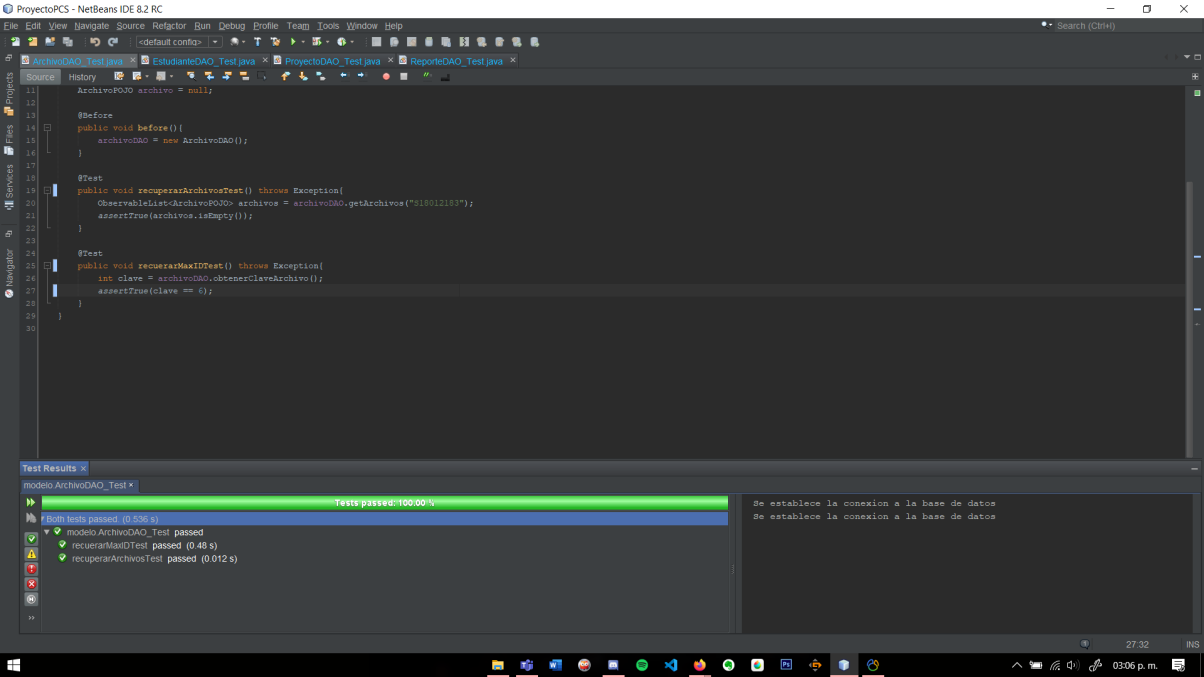
assertTrue(horasAcomuladas == 52);

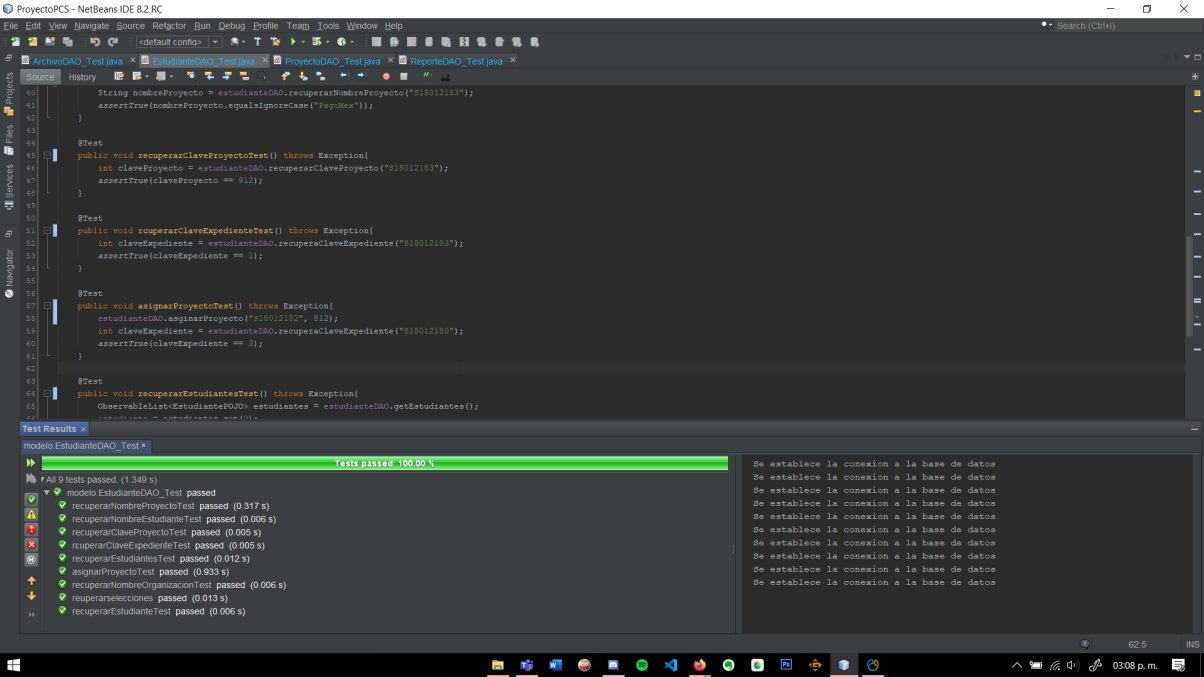
}

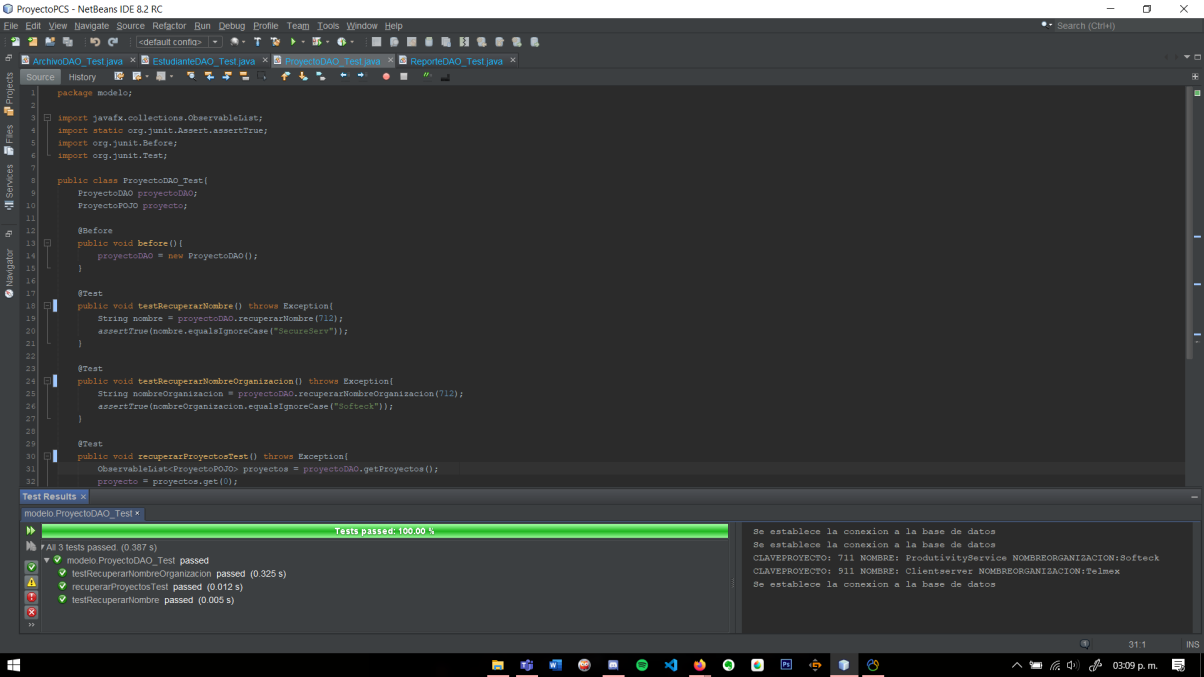
}

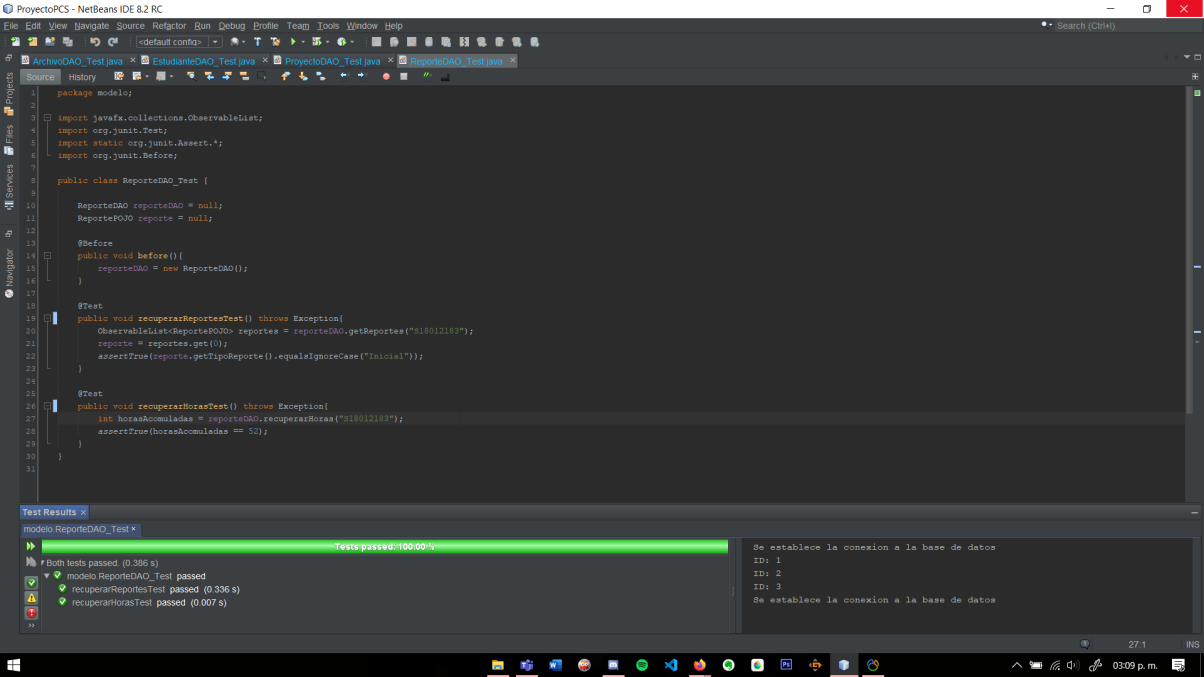
# Reporte de pruebas unitarias

Las siguientes imágenes muestran los reportes de las diferentes pruebas unitarias, las cuales muestran coherencia y como se puede ver, todas pasan. De requerir verlas mas a detalle se encuentran en el código.









# Documentación del api generada apegada al estándar

En este apartado se muestra la documentación de API generada apegada al estándar, el cual incluye la documentación de las clases del proyecto y sus métodos. Sin embargo, con el fin de no hacer este documento extremadamente largo solo se agrega una parte de la documentación la cual es:

* Documentación de paquetes
* Documentación del paquete controlador
* Documentación de la clase EstudianteDAO

El resto de la documentación se encuentra en la carpeta donde se entregó el proyecto.

|  |  |
| --- | --- |
| **Package** | **Description** |
| [**controlador**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\controlador\package-summary.html) | Define las clases que serviran para comunicar la capa de modelo con la vista negociando entre los paquetes la informacion y realizar los procesos para aplicar los cambios a los datos, funcionando como intermediario entre el modeo y la vista administrando el movimiento de informacion entre las capas. |
| [**modelo**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\package-summary.html) | Este paquete conteiene los modelos de las clases que interactuan con la base de datos, en ellas se encuentran los atributos asi como sus getters, setters y sus demas métodos particulares. |
| [**vista**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\vista\package-summary.html) | Contiene los FXML de las vistas de las distintas ventanas con las que interactua de forma directa el usuario, de igual forma contiene los componentes que utiliza el controlador en sus operaciones. |

|  |  |
| --- | --- |
| **Class** | **Description** |
| [**AsignarProyectoController**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\controlador\AsignarProyectoController.html) | Clase controlador de la vista de AsignarProyecto, pantalla que tiene como objetivo realizar la asignación de un ESTUDIANTE registrado en la BD con un PROYECTO que no tenga un ESTUDIANTE asociado. |
| [**consultarProgresoController**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\controlador\consultarProgresoController.html) | Clase controlador de la vista del consultarProgreso, pantalla que tiene como objetivo mostrar la consulta del progreso del ESTUDIANTE que fue recuperado de la pantalla anterior (preConsultarProgreso) por medio de una matricula. |
| [**Main**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\controlador\Main.html) | Clase Main con la que inicia la aplicación. |
| [**MenuController**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\controlador\MenuController.html) | Clase controlador de la vista del Menu, la cual es la primera pantalla de la aplicación. |
| [**preConsultarProgresoController**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\controlador\preConsultarProgresoController.html) | Clase controlador de la vista del preConsultarProgreso, pantalla que tiene como objetivo recuperar una matricula para poder cargar la consulta en la vista consultarProgreso. |
| [**preSubirReporteController**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\controlador\preSubirReporteController.html) | Clase controlador de la vista del preSubirReporte, pantalla que tiene como objetivo recuperar una matricula para poder cargar la entrega en la vista SubirReporte. |
| [**SubirReporteController**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\controlador\SubirReporteController.html) | Clase controlador de la vista del SubirReporte, pantalla que tiene como objetivo mostrar subir un archivo al expediente del ESTUDIANTE que fue recuperado de la pantalla anterior (preSubirReporte) por medio de una matricula. |

|  |  |
| --- | --- |
| **Class** | **Description** |
| [**ArchivoDAO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\ArchivoDAO.html) | Clase que encargada de realizar todas las operaciones de los objetos ARCHIVO manejados dentro del sistema con los registrados en la BD. |
| [**ArchivoPOJO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\ArchivoPOJO.html) | Clase que representa como objeto los ARCHIVOS entregados por los ESTUDIANTES registrados en la BD. |
| [**ConexionDB**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\ConexionDB.html) | Clase que realiza la conexión con la base de datos. |
| [**EstudianteDAO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudianteDAO.html) | Clase que encargada de realizar todas las operaciones de los objetos ESTUDIANTE manejados dentro del sistema con los registrados en la BD. |
| [**EstudiantePOJO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudiantePOJO.html) | Clase que representa como objeto los ESTUDIANTES registrados en la BD. |
| [**ExpedientePOJO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\ExpedientePOJO.html) | Clase que representa como objeto el EXPEDIENTE de los ESTUDIANTES registrados en la BD. |
| [**Inscripción**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\Inscripci%C3%B3n.html) | Clase que representa como objeto la INSCRIPCIÓN de los ESTUDIANTES registrados en la BD. |
| [**OrganizacionDAO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\OrganizacionDAO.html) | Clase que encargada de realizar todas las operaciones de los objetos ORGANIZACIÓN manejados dentro del sistema con los registrados en la BD. |
| [**OrganizacionPOJO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\OrganizacionPOJO.html) | Clase que representa como objeto las ORGANIZACIONES registradas en la BD. |
| [**ProfesorPOJO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\ProfesorPOJO.html) | Clase que representa como objeto los PROFESORES registrados en la BD. |
| [**ProyectoDAO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\ProyectoDAO.html) | Clase que encargada de realizar todas las operaciones de los objetos PROYECTO manejados dentro del sistema con los registrados en la BD. |
| [**ProyectoPOJO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\ProyectoPOJO.html) | Clase que representa como objeto los PROYECTOS pertenecientes a las ORGANIZACIONES registradas en la BD. |
| [**ReporteDAO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\ReporteDAO.html) | Clase que encargada de realizar todas las operaciones de los objetos REPORTE manejados dentro del sistema con los registrados en la BD. |
| [**ReportePOJO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\ReportePOJO.html) | Clase que representa como objeto los REPORTES entregados por los ESTUDIANTES registrados en la BD. |
| [**SeleccionProyectoPOJO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\SeleccionProyectoPOJO.html) | Clase que representa como objeto la SELECCIONPROYECTO de los ESTUDIANTES registrados en la BD. |

public class EstudianteDAO

extends java.lang.Object

Clase que encargada de realizar todas las operaciones de los objetos ESTUDIANTE manejados dentro del sistema con los registrados en la BD.

* + ***Constructor Summary***

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [EstudianteDAO](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudianteDAO.html#EstudianteDAO--)() |

* + ***Method Summary***

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| void | [asginarProyecto](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudianteDAO.html#asginarProyecto-java.lang.String-int-)(java.lang.String matriculaEstudianteElegido, int claveProyectoElegido)  Realiza la asignación de un ESTUDIANTE y un PROYECTO registrados en la BD registrando la INSCRIPCION y abriendo el EXPEDIENTE con las respectivas claves de ambos parametros. |
| javafx.collections.ObservableList<**[EstudiantePOJO](file:///C:\\Users\\ricar\\Documents\\Git\\ProyectoPCS\\ProyectoPCS\\dist\\javadoc\\modelo\\EstudiantePOJO.html" \o "class in modelo)**> | [getEstudiantes](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudianteDAO.html#getEstudiantes--)()  Recupera de la base de datos la lista de ESTUDIANTES registrados en la BD. |
| java.util.ArrayList<**[SeleccionProyectoPOJO](file:///C:\\Users\\ricar\\Documents\\Git\\ProyectoPCS\\ProyectoPCS\\dist\\javadoc\\modelo\\SeleccionProyectoPOJO.html" \o "class in modelo)**> | [getSelecciones](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudianteDAO.html#getSelecciones-java.lang.String-)(java.lang.String matricula)  Recupera de la base de datos la lista de SELECCIONPROYECTO de un ESTUDIANTE especificado registrado en la BD. |
| int | [recuperaClaveExpediente](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudianteDAO.html#recuperaClaveExpediente-java.lang.String-)(java.lang.String matricula)  Recupera la clave del EXPEDIENTE perteneciente a un ESTUDIANTE especificado. |
| [**EstudiantePOJO**](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudiantePOJO.html) | [recuperar](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudianteDAO.html#recuperar-java.lang.String-)(java.lang.String matricula)  Recupera un objeto ESTUDIANTE por medio de una matricula especifica. |
| int | [recuperarClaveProyecto](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudianteDAO.html#recuperarClaveProyecto-java.lang.String-)(java.lang.String matricula)  Recupera la clave del PROYECTO asignado a un ESTUDIANTE especificado. |
| java.lang.String | [recuperarNombreEstudiante](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudianteDAO.html#recuperarNombreEstudiante-java.lang.String-)(java.lang.String matricula)  Recupera el nombre de un ESTUDIANTE especificado. |
| java.lang.String | [recuperarNombreOrganizacion](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudianteDAO.html#recuperarNombreOrganizacion-java.lang.String-)(java.lang.String matricula)  Recupera el nombre de la ORGANIZACION perteneciente al PROYECTO asignado a un ESTUDIANTE especificado. |
| java.lang.String | [recuperarNombreProyecto](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudianteDAO.html#recuperarNombreProyecto-java.lang.String-)(java.lang.String matricula)  Recupera el nombre del PROYECTO asignado a un ESTUDIANTE especificado. |

* + - **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

* + ***Constructor Detail***
    - **EstudianteDAO**

public EstudianteDAO()

* + ***Method Detail***
    - **recuperar**
    - public [EstudiantePOJO](file:///C:\\Users\\ricar\\Documents\\Git\\ProyectoPCS\\ProyectoPCS\\dist\\javadoc\\modelo\\EstudiantePOJO.html" \o "class in modelo) recuperar(java.lang.String matricula)

throws java.lang.Exception

Recupera un objeto ESTUDIANTE por medio de una matricula especifica.

Parameters:

matricula - Matricula del ESTUDIANTE a recuperar.

Returns:

EstudiantePOJO Objeto ESTUDIANTE.

Throws:

java.lang.Exception

* + - **recuperarNombreEstudiante**
    - public java.lang.String recuperarNombreEstudiante(java.lang.String matricula)

throws java.lang.Exception

Recupera el nombre de un ESTUDIANTE especificado.

Parameters:

matricula - Matricula del ESTUDIANTE a recuperar.

Returns:

nombreEstudiante Nombre del ESTUDIAIANTE especificado.

Throws:

java.lang.Exception

* + - **recuperarNombreOrganizacion**
    - public java.lang.String recuperarNombreOrganizacion(java.lang.String matricula)

throws java.lang.Exception

Recupera el nombre de la ORGANIZACION perteneciente al PROYECTO asignado a un ESTUDIANTE especificado.

Parameters:

matricula - Matricula del ESTUDIANTE a recuperar.

Returns:

nombreOrganizacion Nombre de la ORGANIZACION especificado.

Throws:

java.lang.Exception

* + - **recuperarNombreProyecto**
    - public java.lang.String recuperarNombreProyecto(java.lang.String matricula)

throws java.lang.Exception

Recupera el nombre del PROYECTO asignado a un ESTUDIANTE especificado.

Parameters:

matricula - Matricula del ESTUDIANTE a recuperar.

Returns:

nombreProyecto Nombre del PROYECTO especificado.

Throws:

java.lang.Exception

* + - **recuperarClaveProyecto**
    - public int recuperarClaveProyecto(java.lang.String matricula)

throws java.lang.Exception

Recupera la clave del PROYECTO asignado a un ESTUDIANTE especificado.

Parameters:

matricula - Matricula del ESTUDIANTE a recuperar.

Returns:

claveProyecto Clave del PROYECTO especificado.

Throws:

java.lang.Exception

* + - **recuperaClaveExpediente**
    - public int recuperaClaveExpediente(java.lang.String matricula)

throws java.lang.Exception

Recupera la clave del EXPEDIENTE perteneciente a un ESTUDIANTE especificado.

Parameters:

matricula - Matricula del ESTUDIANTE a recuperar.

Returns:

claveExpediente Clave del EXPEDIENTE especificado.

Throws:

java.lang.Exception

* + - **getEstudiantes**
    - public javafx.collections.ObservableList<[EstudiantePOJO](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\EstudiantePOJO.html)> getEstudiantes()

throws java.lang.Exception

Recupera de la base de datos la lista de ESTUDIANTES registrados en la BD.

Returns:

obs Lista contenedora de los ESTUDIANTES.

Throws:

java.lang.Exception

* + - **getSelecciones**
    - public java.util.ArrayList<[SeleccionProyectoPOJO](file:///C:\Users\ricar\Documents\Git\ProyectoPCS\ProyectoPCS\dist\javadoc\modelo\SeleccionProyectoPOJO.html)> getSelecciones(java.lang.String matricula)

throws java.lang.Exception

Recupera de la base de datos la lista de SELECCIONPROYECTO de un ESTUDIANTE especificado registrado en la BD.

Parameters:

matricula - Matricula del ESTUDIANTE del cual se recuperaran las SELECCIONPROYECTO.

Returns:

obs Lista contenedora de las SELECCIONPROYECTO.

Throws:

java.lang.Exception

* + - **asginarProyecto**
    - public void asginarProyecto(java.lang.String matriculaEstudianteElegido,
    - int claveProyectoElegido)

throws java.lang.Exception

Realiza la asignación de un ESTUDIANTE y un PROYECTO registrados en la BD registrando la INSCRIPCION y abriendo el EXPEDIENTE con las respectivas claves de ambos parametros.

Parameters:

matriculaEstudianteElegido - Matricula del ESTUDIANTE a asignar.

claveProyectoElegido - Clave del PROYECTO a asignar.

Throws:

java.lang.Exception

# Definición de la UI

A continuación, se puede apreciar un fragmento de código en el cual la UI comprende una separación adecuada con validaciones y completamente funcional de acuerdo con el diseño. Cabe destacar que este es solo un método ejemplo, ya que todos los métodos cuentan con la misma clase de prácticas.

/\*\*

\* Acción realizada al dar clic en el botón: Aceptar, el cual recupera las

\* cadenas escritas en los textfield requeridos asi como recupera el archivo

\* seleccionado en el filechooser y realiza un registro en la base de datos

\* acorde al ESTUDIANTE seleccionado en la pantalla anterior.

\*

\* @param event El clic del botón.

\* @throws FileNotFoundException Sin selección de archivo.

\*/

@FXML

private void aceptar(ActionEvent event) throws FileNotFoundException, Exception {

ArchivoDAO arch = new ArchivoDAO();

ArchivoPOJO archP = new ArchivoPOJO();

ReporteDAO rep = new ReporteDAO();

ReportePOJO repP = new ReportePOJO();

if(this.validarCampos()){

archP.setTitulo(file.getName());

String fecha = txtFecha.getText();

archP.setFechaEntrega(LocalDate.parse(fecha,

DateTimeFormatter.ofPattern("yyyy-MM-dd")));

String horas = txtHoras.getText();

if(horas.matches("[0-9]\*")){

repP.setHorasReportadas(Integer.parseInt(txtHoras.getText()));

repP.setTipoReporte(combxTipo.getValue());

if(this.validarArchivo()){

try {

byte[] doc = new byte[(int) file.length()];

InputStream input = new FileInputStream(file);

input.read(doc);

archP.setArchivo(doc);

} catch (IOException ex) {

archP.setArchivo(null);

}

arch.subirArchivo(archP, Integer.parseInt(txtClaveExp.getText()));

int idArch = arch.obtenerClaveArchivo();

rep.subirReporte(repP, idArch);

AlertaFXML alerta = new AlertaFXML((Stage)this.btnCancelar.getScene().getWindow());

alerta.alertaInformacion("Exito", "Archivo subido exitosamente",

"El archivo se ha cargado correctamente al sistema");

this.closeWindows();

}

}else{

AlertaFXML alerta = new AlertaFXML((Stage)this.btnCancelar.getScene().getWindow());

alerta.alertaInformacion("Error", "Horas a registrar incorrectas",

"Las horas son aracteres numericos");

}

}

}

# Conclusiones

Por medio de este trabajo logramos aterrizar los conocimientos aprendidos durante el curso de Principios de Construcción de Software que van de la mano con los aprendidos en el curso de Principios de Diseño de Software, en el cual vemos un claro ejemplo del aprendizaje dado por el esfuerzo y empeño a lo largo de este semestre en línea. Por primera vez vemos una parte expuesta de un sistema creado a partir de nuestra percepción el cual nos llevó a mejorar y pulir el mismo con los debidos conocimientos y aplicaciones de Java, JavaFX, JUnit, etc.

La base del buen desarrollo de este trabajo fue el trabajo en equipo el cual, gracias a la coordinación y empeño de cada uno de los integrantes, y por supuesto una buena comunicación, lograron el desarrollo en tiempo y forma por lo que la entrega que se realiza es la versión final y completa de dicho trabajo.

Inclusive, en nuestra percepción, aunque el trabajo inicio con un ligero retraso por la desinformación de cómo se debía empezar, logramos sacar adelante el trabajo pese a la adversidad del semestre en línea y la poca practica de las herramientas que necesitábamos, dicho es el caso de GitHub y JavaFX, ya que en estas dos herramientas desconocíamos el funcionamiento y aplicación correcta debido a que los primeros ejemplos fueron poco en comparación de lo requerido. Sin embargo, con un poco de esfuerzo y autoaprendizaje cada integrante logro resolver sus dudas y compartir soluciones con los demás.

Lo último en realizar fue la documentación y algunas validaciones, debido a que esto fue para pulir el código y poder lograr un mejor funcionamiento de la implementación final del proyecto.

# Referencias

1. Topley, K. (2010). *JavaFX developer's guide*. Pearson Education.
2. Clarke, J., Connors, J., & Bruno, E. J. (2009). *JavaFX: Developing Rich Internet Applications*. Pearson Education.
3. Dea, C., Heckler, M., Grunwald, G., Pereda, J., & Phillips, S. M. (2014). *Javafx 8: Introduction by example*. New York City, USA: Apress.
4. Latte, B., Henning, S., & Wojcieszak, M. (2019). Clean code: On the use of practices and tools to produce maintainable code for long-living.
5. Deitel, P. J., & Deitel, H. M. (2008). *Java: como programar*. Pearson educacion.
6. Massol, V., & Husted, T. (2003). *JUnit in action*. Manning Publications Co..
7. Gamma, E., & Beck, K. (2006). JUnit.
8. Moore, I. (2001). Jester-a JUnit test tester. *Proc. of 2nd XP*, 84-87.
9. Kramer, D. (1999, October). API documentation from source code comments: a case study of Javadoc. In *Proceedings of the 17th annual international conference on Computer documentation* (pp. 147-153).
10. Rantanen, P. (2017). REST API example generation using Javadoc. *Computer Science and Information Systems*, *14*(2), 447-466.
11. Anderson, G., & Anderson, P. (2009). *Essential JavaFX*. Pearson Education.