




27 DE SEPTIEMBRE DE 2021

INTERFAZ CRUD

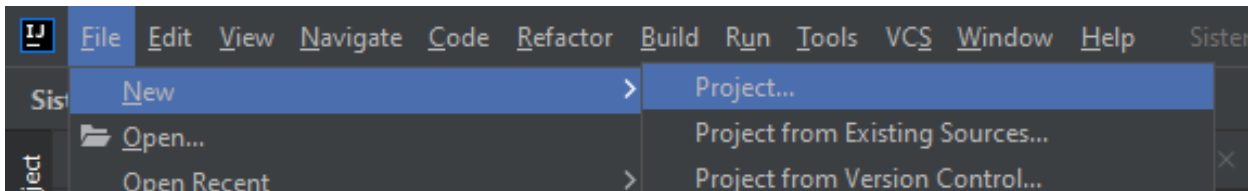
DESARROLLO DE APLICACIONES

DANIEL GARCÍA MORALES
UNIVERSIDAD VERACRUZANA
INGENIERÍA DE SOFTWARE



1. Creando un nuevo proyecto

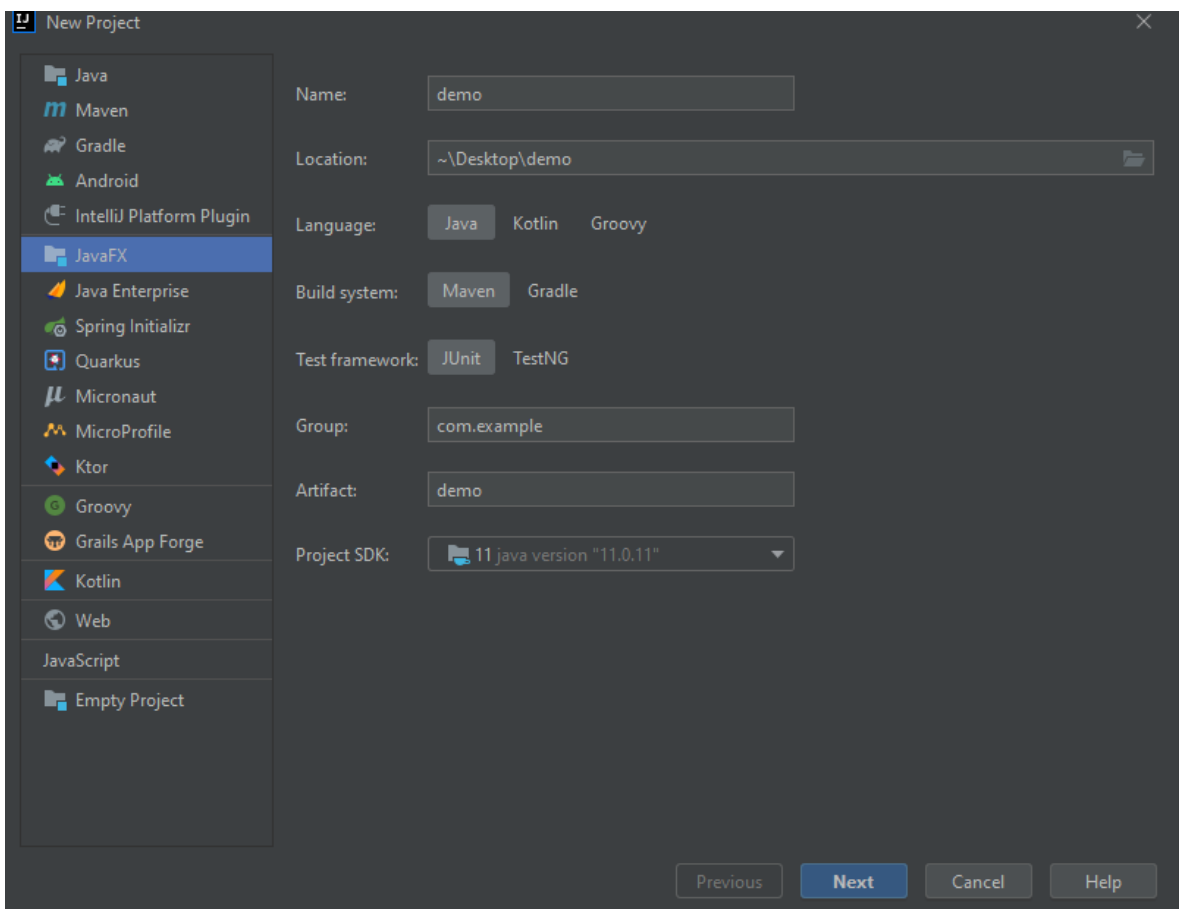
Es el primer paso que debemos de realizar antes de cualquier otra cosa dentro de nuestro editor de código o IDE.



2. Maven

Nuestro proyecto utilizará el framework JavaFX por lo cual deberemos seleccionarlo, darle un nombre a nuestro proyecto, una ruta de guardado y, lo más importante, seleccionar la opción de Maven y la versión 11 de Java.

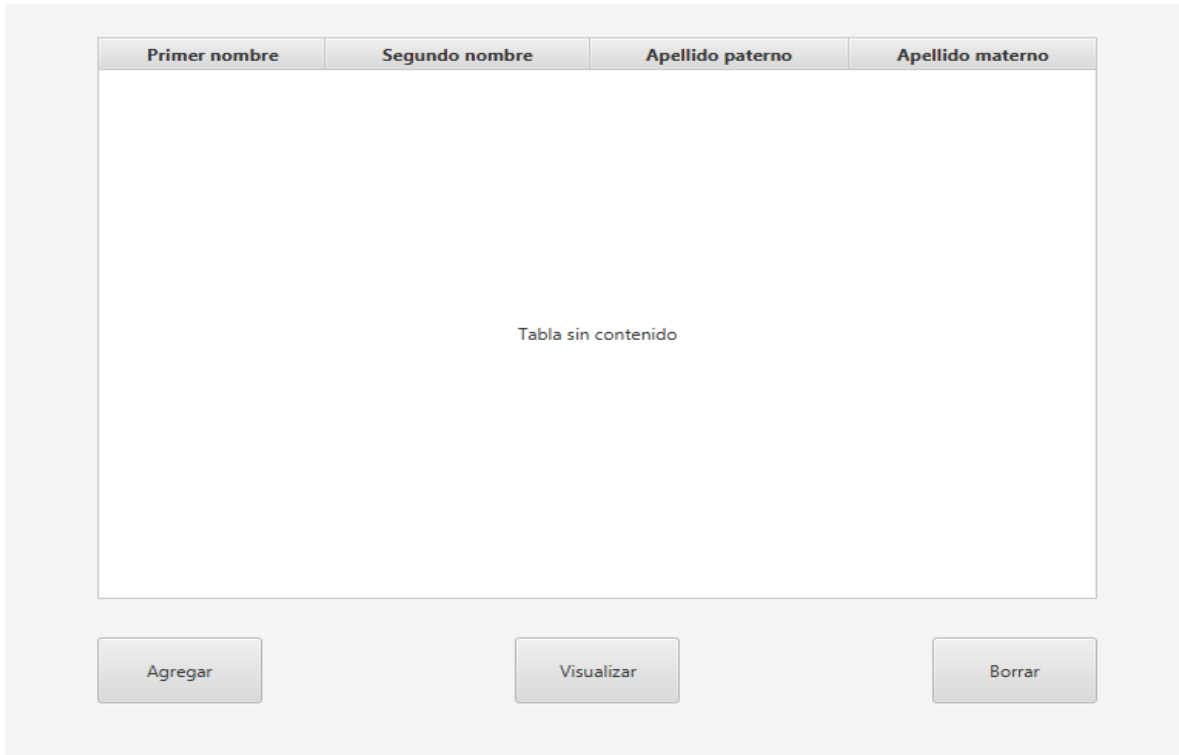
Posteriormente se nos creará un proyecto con Maven.



3. Creando una ventana

Al crear nuestro archivo con JavaFX se nos añadirá una ventana predeterminada que podemos editar nos la herramienta de SceneBuilder.

Nuestra ventana principal mostrará los datos del alumno:



4. Primeras clases

Necesitaremos crear nuestras primeras clases del sistema.

a) Clase main

Nos ayudará con la ejecución del sistema.

```
public class Main extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource("hello-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 768, 574);
        stage.setTitle("Sistema ABC");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch();
    }
}
```

b) Conexión a la base de datos

Nos ayudará a establecer conexión con nuestra base de datos.

```
public class DataBaseConnection {
    public static Connection getConnection() {
        Connection conn = null;
        String url = "jdbc:postgresql://maisonbleue2020.ddns.net";
        String db = "tarea";
        String user = "guest";
        String password = "Tilin123";
        try{
            Class.forName("org.postgresql.Driver");

            conn = DriverManager.getConnection(url + "/" + db, user, password);
            System.out.println("Conexión establecida a PostgreSQL");
        }catch(ClassNotFoundException | SQLException ex){
            System.out.println("Error al abrir Conexión PostgreSQL: " + ex.getMessage());
        }

        return conn;
    }
}
```

c) Clase estudiante

Nos provee los atributos del objeto Estudiante.

```
public class Estudiante {
    private int id;
    private String primerNombre;
    private String segundoNombre;
    private String primerApellido;
    private String segundoApellido;
    private boolean activation;

    public Estudiante(int id, String primerNombre, String segundoNombre, String primerApellido, String
segundoApellido, boolean activation) {
        this.id = id;
        this.primerNombre = primerNombre;
        this.segundoNombre = segundoNombre;
        this.primerApellido = primerApellido;
        this.segundoApellido = segundoApellido;
        this.activation = activation;
    }

    public Estudiante() {
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getPrimerNombre() {
        return primerNombre;
    }

    public void setPrimerNombre(String primerNombre) {
        this.primerNombre = primerNombre;
    }

    public String getSegundoNombre() {
        return segundoNombre;
    }

    public void setSegundoNombre(String segundoNombre) {
        this.segundoNombre = segundoNombre;
    }

    public String getPrimerApellido() {
        return primerApellido;
    }

    public void setPrimerApellido(String primerApellido) {
        this.primerApellido = primerApellido;
    }

    public String getSegundoApellido() {
        return segundoApellido;
    }

    public void setSegundoApellido(String segundoApellido) {
        this.segundoApellido = segundoApellido;
    }

    public boolean isActivation() {
        return activation;
    }

    public void setActivation(boolean activation) {
        this.activation = activation;
    }
}
```

d) Clase Estudiante DAO

Nos provee todos los métodos necesarios para el manejo del objeto Estudiante dentro de la base de datos.

```
public class Estudiante_DAO {
    public static List<Estudiante> getEstudiantesActivos(){
        Connection connection = DataBaseConnection.getConnection();
        List<Estudiante> estudiantes = new ArrayList<>();
        String query = "SELECT * FROM public.\"estudiante\" WHERE activo = true ORDER BY id ASC";
        System.out.println("Conexión establecida");

        try{
            PreparedStatement sentence = connection.prepareStatement(query);
            ResultSet resultSet = sentence.executeQuery();
            Estudiante estudiante;
            while (resultSet.next()){
                estudiante = new Estudiante(
                    resultSet.getInt("id"),
                    resultSet.getString("primer_nom"),
                    resultSet.getString("segundo_nom"),
                    resultSet.getString("primer_ape"),
                    resultSet.getString("segundo_ape"),
                    resultSet.getBoolean("activo"));
                estudiantes.add(estudiante);
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
        return estudiantes;
    }

    public static void guardar(Estudiante estudiante){
        Connection connection = DataBaseConnection.getConnection();
        try {
            String query = "INSERT INTO public.\"estudiante\" (primer_ape, segundo_ape, primer_nom, segundo_nom, activo) VALUES (?, ?, ?, ?, ?)";
            PreparedStatement preparedStatement = connection.prepareStatement(query);
            preparedStatement.setString(1, estudiante.getPrimerApellido());
            preparedStatement.setString(2, estudiante.getSegundoApellido());
            preparedStatement.setString(3, estudiante.getPrimerNombre());
            preparedStatement.setString(4, estudiante.getSegundoNombre());
            preparedStatement.setBoolean(5, estudiante.isActivation());
            preparedStatement.execute();
            preparedStatement.close();
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }

    public static void borrar(Estudiante estudiante){
        Connection connection = DataBaseConnection.getConnection();

        try {
            String query = "UPDATE public.\"estudiante\" SET activo=false WHERE id = ?";
            PreparedStatement preparedStatement = connection.prepareStatement(query);
            preparedStatement.setInt(1, estudiante.getId());
            preparedStatement.execute();
            preparedStatement.close();
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }
}
```

5. Clase extra

Esta clase nos provee métodos para abrir ventanas de manera más fácil.

```
public class Ventana {
    public static final boolean CERRAR = true;
    public static final boolean NO_CERRAR = false;

    public static void iniciarVentana(Button boton, Object controlador, String nombreVentana, boolean cerrarVentana)
    {
        try{
            FXMLLoader loader = new FXMLLoader(Ventana.class.getResource(nombreVentana));
            loader.setController(controlador);
            Stage stage = new Stage();
            Parent parent = (Parent) loader.load();
            stage.setScene(new Scene(parent));
            stage.initModality(Modality.APPLICATION_MODAL);
            stage.show();
            if(cerrarVentana){
                cerrar(boton);
            }
        } catch (IOException ex) {
            Logger.getLogger(Ventana.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public static void cerrar(Button boton){
        Stage stagePrincipal = (Stage) boton.getScene().getWindow();
        stagePrincipal.close();
    }
}
```

6. Creando métodos

En el controlador de nuestra ventana principal deberemos crear un método que nos despliegue los datos almacenados en la DB respectivos al estudiante.

```
public void initialize(URL url, ResourceBundle resourceBundle) {
    try {
        llenarTablaEstudiantes();
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    }
}
```

```
private void llenarTablaEstudiantes() throws SQLException {
    this.clmn_primerNom.setCellValueFactory(new PropertyValueFactory<Estudiante, String>("primerNombre"));
    this.clmn_segundoNom.setCellValueFactory(new PropertyValueFactory<Estudiante, String>("segundoNombre"));
    this.clmn_apellidoPat.setCellValueFactory(new PropertyValueFactory<Estudiante, String>("primerApellido"));
    this.clmn_apellidoMat.setCellValueFactory(new PropertyValueFactory<Estudiante, String>("segundoApellido"));

    Estudiante_DAO estudiante_dao = new Estudiante_DAO();
    List<Estudiante> listaEstudiante = Estudiante_DAO.getEstudiantesActivos();

    observableEstudiante = FXCollections.observableArrayList(listaEstudiante);
    this.tb_Estudiantes.setItems(observableEstudiante);
}
```

7. Primeros resultados

Con todo lo anterior ya tendremos todo lo necesario para desplegar nuestra primera ventana que despliega los datos almacenados en nuestra base de datos.

Sistema ABC

Primer nombre	Segundo nombre	Apellido paterno	Apellido materno
Theresita	Maurie	Bustin	Medlen
Sonnie	Konstantin	Whellams	McConway
Dara	Jonathon	Coupe	Verling
Tammara	Gibby	O'Shevlan	Azam
Artie	Reinwald	Boyan	Shipway
Saundra	Braden	Waldie	Bristowe
Grannie	Avram	Belfitt	Skitch
Ethelda	Latisha	Rathe	Lilywhite
Esma	Cad	Grestie	Compton
Eddy	Meade	Lambdin	Sall
Ariel	Laura	Koppe	Pike
Garner	Brooke	Banville	Shakshaft
Noel	Etti	Sutherby	Ortmann
Olwen	Early	Heathorn	Leipelt
Isabelita	Montgomery	Gaskell	Marzelo
Meggy	Valene	Tinman	Massow

Agregar Visualizar Borrar

8. Nuevas ventanas

Ahora podemos crear las ventanas para visualizar al estudiante y agregar un nuevo estudiante.

Primer nombre:

Segundo nombre:

Primer apellido:

Segundo apellido:

Aceptar Cancelar



9. Dando funcionalidad

Creamos los métodos necesarios para cada uno de los controladores de nuestras nuevas ventanas.

```
public class ConsultarController implements Initializable {
    public Label label_PrimerNombre;
    public Label label_SegundoNombre;
    public Label label_PrimerApellido;
    public Label label_SegundoApellido;
    public Button button_Aceptar;

    Estudiante estudiante;

    public ConsultarController(Estudiante estudiante){
        this.estudiante = estudiante;
    }

    public void clickAceptar(ActionEvent actionEvent) {
        Stage estaVentana = (Stage)button_Aceptar.getScene().getWindow();
        estaVentana.close();
    }

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        label_PrimerNombre.setText(estudiante.getPrimerNombre());
        label_SegundoNombre.setText(estudiante.getSegundoNombre());
        label_PrimerApellido.setText(estudiante.getPrimerApellido());
        label_SegundoApellido.setText(estudiante.getSegundoApellido());
    }
}
```

```

public class AgregarController implements Initializable {
    @FXML
    public Button button_Aceptar;
    @FXML
    public Button button_Cancelar;
    @FXML
    public TextField txtField_PrimerNombre;
    @FXML
    public TextField txtField_SegundoNombre;
    @FXML
    public TextField txtField_PrimerApellido;
    @FXML
    public TextField txtField_SegundoApellido;

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        UnaryOperator<TextFormatter.Change> characterFilter = change -> {
            String newText = change.getControlNewText();
            if (newText.matches("[a-zA-Z ]*")) {
                return change;
            } else {
                return null;
            }
        };

        txtField_PrimerNombre.setTextFormatter(new TextFormatter<String>(characterFilter));
        txtField_SegundoNombre.setTextFormatter(new TextFormatter<String>(characterFilter));
        txtField_PrimerApellido.setTextFormatter(new TextFormatter<String>(characterFilter));
        txtField_SegundoApellido.setTextFormatter(new TextFormatter<String>(characterFilter));
    }

    public void clickAceptar(ActionEvent actionEvent) throws SQLException {
        if(this.txtField_PrimerNombre.getText().isEmpty() || this.txtField_SegundoNombre.getText().isEmpty() ||
            this.txtField_PrimerApellido.getText().isEmpty() ||
            this.txtField_SegundoApellido.getText().isEmpty()){
            Alert alertaConfirmacion = new Alert(Alert.AlertType.WARNING);
            alertaConfirmacion.setHeaderText(null);
            alertaConfirmacion.setTitle("Alerta");
            alertaConfirmacion.setContentText("Aegúrese de llenar todos los campos.");
            alertaConfirmacion.showAndWait();
        }else{
            Estudiante estudiante = new Estudiante();
            estudiante.setPrimerNombre(this.txtField_PrimerNombre.getText());
            estudiante.setSegundoNombre(this.txtField_SegundoNombre.getText());
            estudiante.setPrimerApellido(this.txtField_PrimerApellido.getText());
            estudiante.setSegundoApellido(this.txtField_SegundoApellido.getText());
            estudiante.setActivation(true);

            try{
                Estudiante_DAO estudiante_dao = new Estudiante_DAO();
                estudiante_dao.guardar(estudiante);

                Alert alertaConfirmacion = new Alert(Alert.AlertType.CONFIRMATION);
                alertaConfirmacion.setHeaderText(null);
                alertaConfirmacion.setTitle("Información");
                alertaConfirmacion.setContentText("Registro exitoso.");
                alertaConfirmacion.showAndWait();
            }catch (Exception ex){
                Alert alertaConfirmacion = new Alert(Alert.AlertType.WARNING);
                alertaConfirmacion.setHeaderText(null);
                alertaConfirmacion.setTitle("Alerta");
                alertaConfirmacion.setContentText("Error en la conexión." + ex);
                alertaConfirmacion.showAndWait();
            }

            limpiarCampos();
        }
    }

    public void clickCancelar(ActionEvent actionEvent) {
        Stage estaVentana = (Stage)button_Cancelar.getScene().getWindow();
        estaVentana.close();
    }

    public void limpiarCampos(){
        this.txtField_PrimerNombre.setText("");
        this.txtField_SegundoNombre.setText("");
        this.txtField_PrimerApellido.setText("");
        this.txtField_SegundoApellido.setText("");
    }
}

```

10. Ya casi

Ahora podemos registrar nuevos estudiantes y visualizar sus datos.

Primer nombre:	<input type="text" value="Daniel"/>
Segundo nombre:	<input type="text" value="Daniel"/>
Primer apellido:	<input type="text" value="Garcia"/>
Segundo apellido:	<input type="text" value="Morales"/>
<div><input type="button" value="Aceptar"/><input type="button" value="Cancelar"/></div>	

Primer nombre:	Daniel
Segundo nombre:	Daniel
Primer apellido:	Garcia
Segundo apellido:	Morales
<div><input type="button" value="Aceptar"/></div>	

11. Último método

Para finalizar, crearemos el método para eliminar un Estudiante.

Para esto haremos uso únicamente del DAO y no de una ventana extra, únicamente extraemos el Estudiante seleccionado y lo pasamos como atributo en el método del DAO. El método cambia de estado a Activo = False, por lo cual hará que deje de aparecer en nuestra tabla.

Este método está en el controlador de la ventana principal, ya que en ella es donde podemos seleccionar un estudiante y así obtener ese objeto de manera más rápida.

```
protected void borrar(ActionEvent actionEvent){
    Estudiante estudiante = this.tb_Estudiantes.getSelectionModel().getSelectedItem();
    if(estudiante != null){
        try{
            Estudiante_DAO estudiante_dao = new Estudiante_DAO();
            estudiante_dao.borrar(estudiante);

            Alert alertaConfirmacion = new Alert(Alert.AlertType.CONFIRMATION);
            alertaConfirmacion.setHeaderText(null);
            alertaConfirmacion.setTitle("Información");
            alertaConfirmacion.setContentText("Se ha eliminado el estudiante.");
            alertaConfirmacion.showAndWait();
        }catch (Exception ex){
            Alert alertaConfirmacion = new Alert(Alert.AlertType.WARNING);
            alertaConfirmacion.setHeaderText(null);
            alertaConfirmacion.setTitle("Alerta");
            alertaConfirmacion.setContentText("Error en la conexión." + ex);
            alertaConfirmacion.showAndWait();
        }
    }
}
```

Y listo, con esto ya tendremos todos nuestros CRUDs realizados.