Entregable #6

Presentacion

Trabajo realizado por Jhonatan David Asprilla Arango

> **CC** 1018222341
> jasprilla@unal.edu.co

Punto 1

> El código de este ejercicio también se encuentra disponible en el GitHub.

> **Nota:** El archivo en el cual se almacenan los datos es el documento `./users.txt`, el cual se creara en caso de que no exista.

Codigo

```java
package entregable_seis;

import java.io.File;
import java.io.RandomAccessFile;
import java.util.ArrayList;
import java.util.Arrays;
import java.awt.BorderLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

interface Subscriber {
     void Notify();
}

interface Suscriptable {
     void registerSubscriber(Subscriber subscriber);
     void notifySubscribers();
}

interface Database {
    public void connect();
    public void disconnect();
    public Model query(int query_field, String query_value);
    public Model[] query_all();
    public void update(int query_field, String query_value, int new_field, String
new_value);
    public void delete(int query_field, String query_value);
    public boolean insert(Model model);
}

interface Model {
    public String serialize();
    public void deserialize(String serialized);
}

class User implements Model {
    public String name;
    public Number number;

    public User(String line) {
        this.name = null;
        this.number = null;
        this.deserialize(line);
    }

    public User(String name, Number number) {
        this.name = name;
        this.number = number;
    }

    @Override
    public void deserialize(String serialized) {
        this.name = serialized.split("!")[0];
        this.number = Integer.parseInt(serialized.split("!")[1]);
    }

    @Override
    public String serialize() {
        return this.name + "!" + this.number.toString();
    }
}

class FileDatabase implements Database {
    private File file;
    private RandomAccessFile fileHandle;
    private String filePath;

    public FileDatabase(String filePath) {
        this.filePath = filePath;
    }

    private String read_file() {
        String lines = "";
        try {
            this.fileHandle.seek(0);
            while (this.fileHandle.getFilePointer() < this.fileHandle.length()) {
                try {
                    String line = this.fileHandle.readLine();
                    lines += line + "\n";
                } catch (Exception e) {
                    System.out.println("FileIOHandler.query: reading line, " +
e.getMessage());
                }
            }
        } catch (Exception e) {
            System.out.println("FileIOHandler.query: " + e.getMessage());
        }
        return lines;
    }

    private void write_file(String new_content) {
        try {
            String processed_content = String.join("\n",
Arrays.stream(new_content.split("\n")).filter(line ->
!line.equals("")).toArray(String[]::new));
            fileHandle.seek(0);
            fileHandle.writeBytes(processed_content);
            fileHandle.setLength(processed_content.length());
        } catch (Exception e) {
            System.out.println("FileIOHandler.query: " + e.getMessage());
        }
    }
```

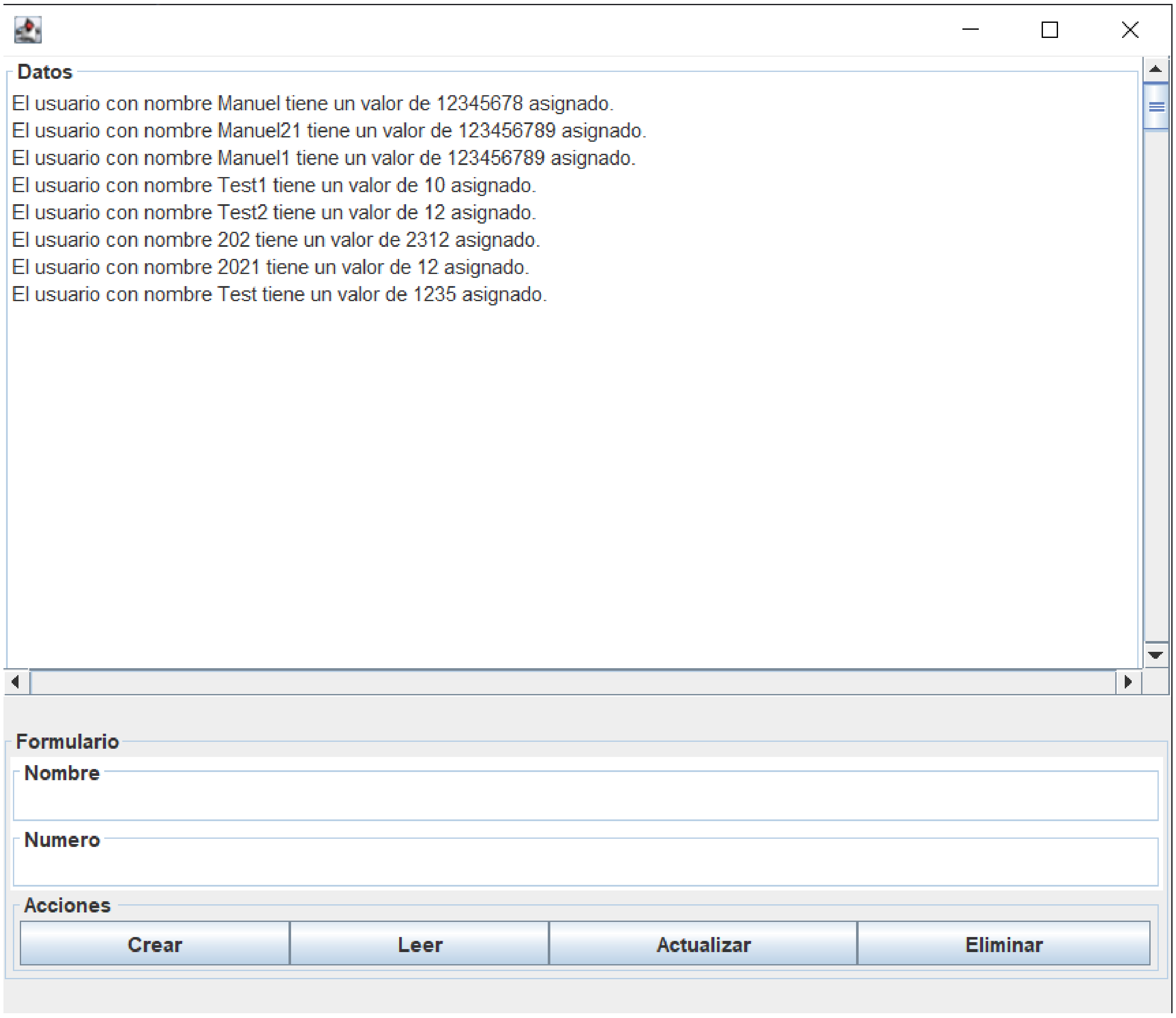
```java
    @Override
    public void connect() {
        try {
            this.file = new File(this.filePath);
            if (!this.file.exists()) {
                this.file.createNewFile();
            }
            this.fileHandle = new RandomAccessFile(this.file, "rw");
        } catch (Exception e) {
            System.out.println("FileIOHandler.open: " + e.getMessage());
        }
    }

    @Override
    public void disconnect() {
        try {
            this.fileHandle.close();
        } catch (Exception e) {
            System.out.println("FileIOHandler.close: " + e.getMessage());
        }
    }

    @Override
    public Model[] query_all() {
        return Arrays.stream(this.read_file().split("\n"))
            .filter(line -> !line.equals(""))
            .map(line -> new User(line))
            .toArray(User[]::new);
    }

    @Override
    public Model query(int query_field, String query_value) {
        String outputLine = null;
        try {
            this.fileHandle.seek(0);
            while (this.fileHandle.getFilePointer() < this.fileHandle.length()) {
                try {
                    String line = this.fileHandle.readLine();
                    if (line.contains("!"))
                        if (line.split("!")[query_field].equals(query_value)) {
                            outputLine = line;
                            break;
                        }
                } catch (Exception e) {
                    System.out.println("FileIOHandler.query: reading line, " +
e.getMessage());
```

```
                }
            }
        } catch (Exception e) {
            System.out.println("FileIOHandler.query: " + e.getMessage());
        }
        User user = new User(outputLine);
        return user;
    }

    @Override
    public void update(int query_field, String query_value, int new_field, String
new_value) {
        try {
            this.fileHandle.seek(0);
            while (this.fileHandle.getFilePointer() < this.fileHandle.length()) {
                try {
                    String line = this.fileHandle.readLine();
                    if (line.split("!")[query_field].equals(query_value)) {
                        User user = new User(line);
                        boolean modified = false;
                        switch (new_field) {
                            case 0:
                                boolean model_doesnt_exists =
Arrays.stream(this.read_file().split("\n"))
                                    .filter(_line -> !_line.equals(""))
                                    .map(_line -> new User(_line).name)
                                    .filter(name -> name.equals((new_value)))
                                    .findFirst()
                                    .isEmpty();
                                if (model_doesnt_exists) {
                                    user.name = new_value;
                                    modified = true;
                                } else {
                                    System.out.println("FileIOHandler.update: user
already exists");
                                }
                                break;
                            case 1:
                                user.number = Integer.parseInt(new_value);
                                modified = true;
                                break;
                            default:
                                break;
                        }
                        if (modified) {
                            String file_content = this.read_file();
                            file_content = file_content.replace(line,
user.serialize());
                            this.write_file(file_content);
                        }
                    }
                } catch (Exception e) {
                    System.out.println("FileIOHandler.query: reading line, " +
e.getMessage());
                }
            }
        } catch (Exception e) {
            System.out.println("FileIOHandler.query: " + e.getMessage());
        }
    }

    @Override
    public void delete(int query_field, String query_value) {
        try {
            this.fileHandle.seek(0);
            while (this.fileHandle.getFilePointer() < this.fileHandle.length()) {
                try {
                    String line = this.fileHandle.readLine();
                    if (line.split("!")[query_field].equals(query_value)) {
                        String file_content = this.read_file();
                        file_content = file_content.replace(line, "");
                        this.write_file(file_content);
                    }
                } catch (Exception e) {
                    System.out.println("FileIOHandler.query: reading line, " +
e.getMessage());
                }
            }
        } catch (Exception e) {
            System.out.println("FileIOHandler.query: " + e.getMessage());
        }
    }

    @Override
    public boolean insert(Model model) {
        try {
            this.fileHandle.seek(this.fileHandle.length());
            boolean model_doesnt_exists =
Arrays.stream(this.read_file().split("\n"))
                .filter(line -> !line.equals(""))
                .map(line -> new User(line).name)
                .filter(name -> name.equals(((User)model).name))
```

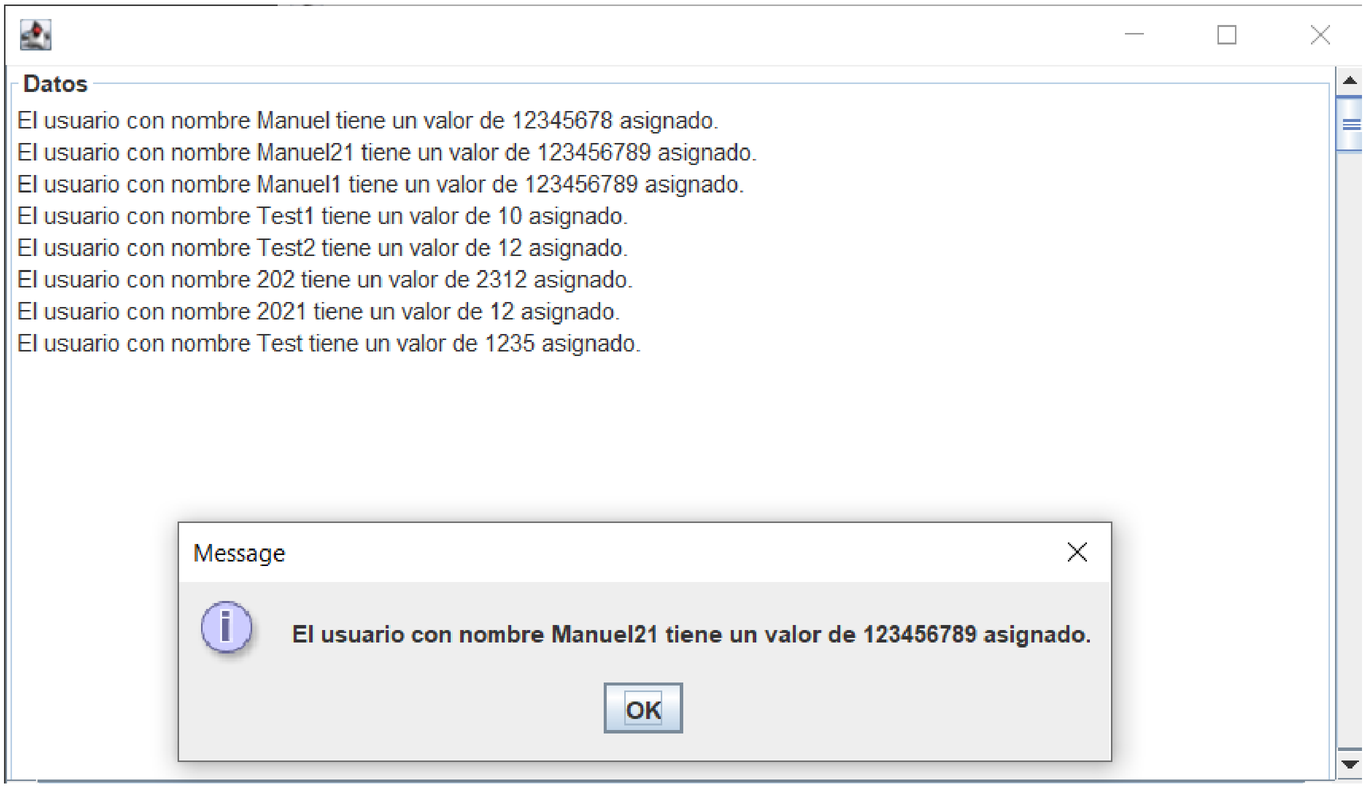
```
                  .findFirst()
                  .isEmpty();

            if (model_doesnt_exists) {
                this.fileHandle.writeBytes("\n" + model.serialize());
                return true;
            } else {
                System.out.println("FileIOHandler.insert: user already exists");
            }
        } catch (Exception e) {
            System.out.println("FileIOHandler.insert: " + e.getMessage());
        }
        return false;
    }
}

class UserRepository implements Suscriptable {
    private Database db;
    public ArrayList<User> users;
    private ArrayList<Subscriber> subscribers;

    public UserRepository(Database db) {
        this.db = db;
        this.users = new ArrayList<User>(Arrays.asList((User[]) db.query_all()));
        this.subscribers = new ArrayList<Subscriber>();
    }

    public String toString() {
        return this.users.stream()
            .map(user -> "El usuario con nombre " + user.name + " tiene un valor de
" + user.number.toString() + " asignado.")
            .reduce("", (acc, user) -> acc + user + "\n");
    }

    public void create(String name, Number number) {
        User user = new User(name, number);
        if (this.db.insert(user))
            users.add(user);
        else
            JOptionPane.showMessageDialog(null, "El usuario ya existe");
        this.notifySubscribers();
    }

    public User read(String name) {
        User user = (User) this.db.query(0, name);
        return user;
```

```java
    }

    public void update(String name, Number number) {
        this.db.update(0, name, 1, number.toString());
        this.users.stream()
            .filter(user -> user.name.equals(name))
            .findFirst()
            .get()
            .number = number;
        this.notifySubscribers();
    }

    public void delete(String name) {
        this.db.delete(0, name);
        this.users.removeIf(user -> user.name.equals(name));
        this.notifySubscribers();
    }

    @Override
    public void notifySubscribers() {
        this.subscribers.forEach(subscriber -> subscriber.Notify());
    }

    @Override
    public void registerSubscriber(Subscriber subscriber) {
        this.subscribers.add(subscriber);
    }
}

class UserForm extends JFrame implements Subscriber, ActionListener {
    private UserRepository userRepository;
    private JTextArea dataField;
    private JScrollPane dataScroll;
    private JPanel rootPanel;
    private JPanel buttonPanel;
    private JPanel formPanel;
    private JButton createButton;
    private JButton readButton;
    private JButton updateButton;
    private JButton deleteButton;
    private JTextField nameField;
    private JTextField numberField;

    public UserForm(UserRepository userRepository) {
        this.userRepository = userRepository;
        this.dataField = new JTextArea(256, 10);
        this.rootPanel = new JPanel(new GridBagLayout());

        this.formPanel = new JPanel(new GridBagLayout());
        this.buttonPanel = new JPanel(new GridBagLayout());

        this.nameField = new JTextField(10);
        this.numberField = new JTextField(10);

        this.createButton = new JButton("Crear");
        this.createButton.setActionCommand("create");
        this.createButton.addActionListener(this);

        this.readButton = new JButton("Leer");
        this.readButton.setActionCommand("read");
        this.readButton.addActionListener(this);

        this.updateButton = new JButton("Actualizar");
        this.updateButton.setActionCommand("update");
        this.updateButton.addActionListener(this);

        this.deleteButton = new JButton("Eliminar");
        this.deleteButton.setActionCommand("delete");
        this.deleteButton.addActionListener(this);

        userRepository.registerSubscriber(this);
    }

    public void init() {
        this.setLayout(new BorderLayout());

        GridBagConstraints constraints = new GridBagConstraints();

        constraints.fill = GridBagConstraints.BOTH;
        constraints.weighty = 1;
        constraints.weightx = 1;
        constraints.gridx = 0;
        constraints.gridy = 0;

        this.dataField.setBorder(BorderFactory.createTitledBorder("Datos"));
        this.dataField.setEditable(false);
        this.dataField.setText(this.userRepository.toString());

        this.dataScroll = new JScrollPane(
            this.dataField,
            JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
            JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS
        );

        rootPanel.add(this.dataScroll, constraints);
        constraints.fill = GridBagConstraints.HORIZONTAL;
        constraints.gridx = 0;
        buttonPanel.add(this.createButton, constraints);
        constraints.gridx = 1;
        buttonPanel.add(this.readButton, constraints);
        constraints.gridx = 2;
        buttonPanel.add(this.updateButton, constraints);
        constraints.gridx = 3;
        buttonPanel.add(this.deleteButton, constraints);

        this.buttonPanel.setBorder(BorderFactory.createTitledBorder("Acciones"));

        constraints.gridx = 0;
        constraints.gridy = 0;

        this.nameField.setBorder(BorderFactory.createTitledBorder("Nombre"));
        formPanel.add(this.nameField, constraints);
        constraints.gridy = 1;
        this.numberField.setBorder(BorderFactory.createTitledBorder("Numero"));
        constraints.gridy = 2;
        formPanel.add(this.numberField, constraints);
        constraints.gridy = 3;
        formPanel.add(this.buttonPanel, constraints);

        formPanel.setBorder(BorderFactory.createTitledBorder("Formulario"));

        constraints.gridx = 0;
        constraints.gridy = 1;
        constraints.fill = GridBagConstraints.HORIZONTAL;
        constraints.weighty = 0.1;

        rootPanel.add(formPanel, constraints);

        this.add(rootPanel, BorderLayout.CENTER);
        this.pack();
        this.setSize(700, 600);
        this.setVisible(true);
    }

    @Override
    public void Notify() {
        this.dataField.setText(this.userRepository.toString());
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        switch (e.getActionCommand()) {
            case "create":
                this.userRepository.create(this.nameField.getText(),
Integer.parseInt(this.numberField.getText()));
                break;
            case "read":
                User user = this.userRepository.read(this.nameField.getText());
                this.nameField.setText(user.name);
                this.numberField.setText(user.number.toString());
                JOptionPane.showMessageDialog(null, "El usuario con nombre " +
user.name +  " tiene un valor de " + user.number + " asignado.");
                break;
            case "update":
                this.userRepository.update(this.nameField.getText(),
Integer.parseInt(this.numberField.getText()));
                JOptionPane.showMessageDialog(null, "Se actualizo el usuario con
nombre " + this.nameField.getText() + " con el valor " +
this.numberField.getText());
                break;
            case "delete":
                this.userRepository.delete(this.nameField.getText());
                JOptionPane.showMessageDialog(null, "Se elimino el usuario con
nombre " + this.nameField.getText());
                break;
            default:
                break;
        }
    }
}

public class Main {
    public static void main(String[] args) {
        FileDatabase db = new FileDatabase("./users.txt");
        db.connect();
        UserForm userForm = new UserForm(new UserRepository(db));
        userForm.init();
        userForm.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                db.disconnect();
                System.exit(0);
            }
        });
    }
}
```

Screenshots


Datos

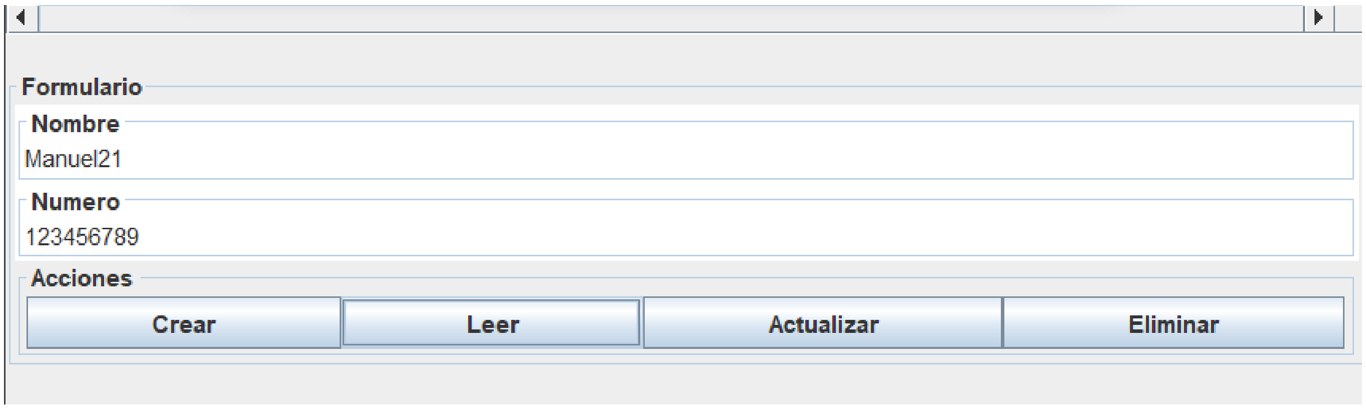
El usuario con nombre Manuel tiene un valor de 12345678 asignado.
El usuario con nombre Manuel21 tiene un valor de 123456789 asignado.
El usuario con nombre Manuel1 tiene un valor de 123456789 asignado.
El usuario con nombre Test1 tiene un valor de 10 asignado.
El usuario con nombre Test2 tiene un valor de 12 asignado.
El usuario con nombre 202 tiene un valor de 2312 asignado.
El usuario con nombre 2021 tiene un valor de 12 asignado.
El usuario con nombre Test tiene un valor de 1235 asignado.


Datos

El usuario con nombre Manuel tiene un valor de 12345678 asignado.
El usuario con nombre Manuel21 tiene un valor de 123456789 asignado.
El usuario con nombre Manuel1 tiene un valor de 123456789 asignado.
El usuario con nombre Test1 tiene un valor de 10 asignado.
El usuario con nombre Test2 tiene un valor de 12 asignado.
El usuario con nombre 202 tiene un valor de 2312 asignado.
El usuario con nombre 2021 tiene un valor de 12 asignado.
El usuario con nombre Test tiene un valor de 1235 asignado.

Message

El usuario con nombre Manuel21 tiene un valor de 123456789 asignado.


Formulario

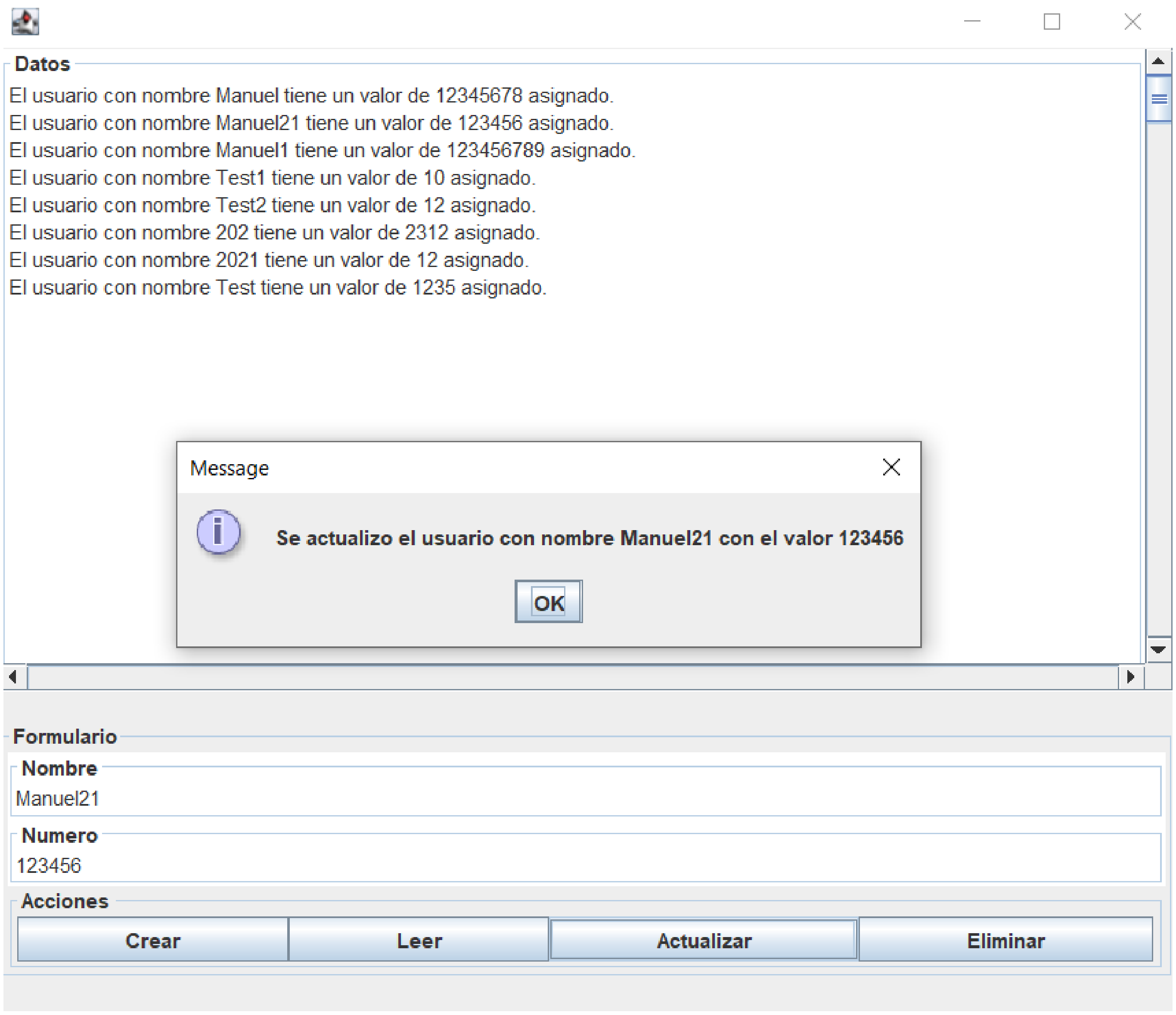
Nombre

Manuel21

Numero

123456789

Acciones


Datos

El usuario con nombre Manuel tiene un valor de 12345678 asignado.
El usuario con nombre Manuel21 tiene un valor de 123456 asignado.
El usuario con nombre Manuel1 tiene un valor de 123456789 asignado.
El usuario con nombre Test1 tiene un valor de 10 asignado.
El usuario con nombre Test2 tiene un valor de 12 asignado.
El usuario con nombre 202 tiene un valor de 2312 asignado.
El usuario con nombre 2021 tiene un valor de 12 asignado.
El usuario con nombre Test tiene un valor de 1235 asignado.

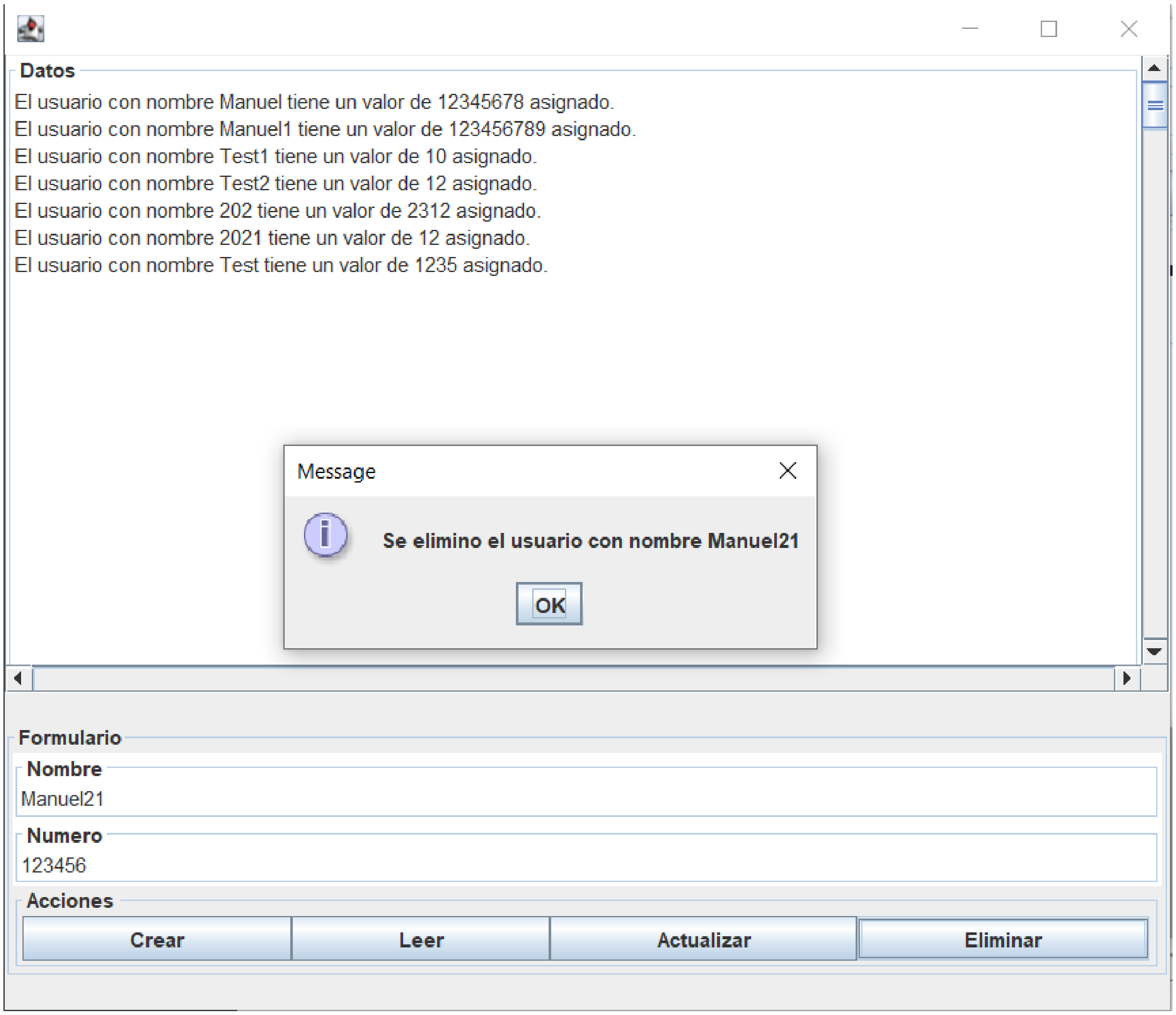
Message

Se actualizo el usuario con nombre Manuel21 con el valor 123456

Formulario

Nombre: Manuel21

Numero: 123456



El usuario con nombre Manuel tiene un valor de 12345678 asignado.
El usuario con nombre Manuel1 tiene un valor de 123456789 asignado.
El usuario con nombre Test1 tiene un valor de 10 asignado.
El usuario con nombre Test2 tiene un valor de 12 asignado.
El usuario con nombre 202 tiene un valor de 2312 asignado.
El usuario con nombre 2021 tiene un valor de 12 asignado.
El usuario con nombre Test tiene un valor de 1235 asignado.

Message

Se elimino el usuario con nombre Manuel21

Nombre: Manuel21

Numero: 123456

Diagrama de clases

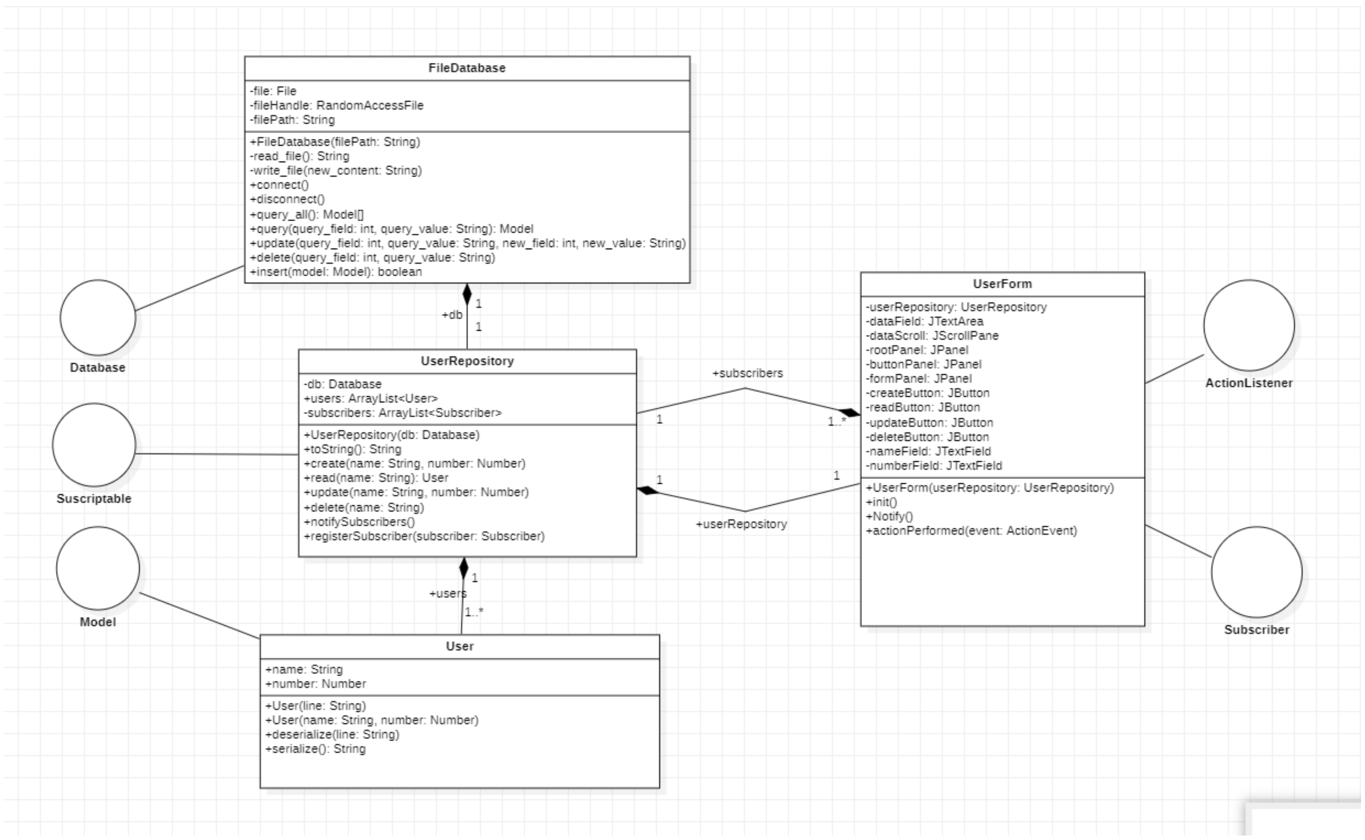
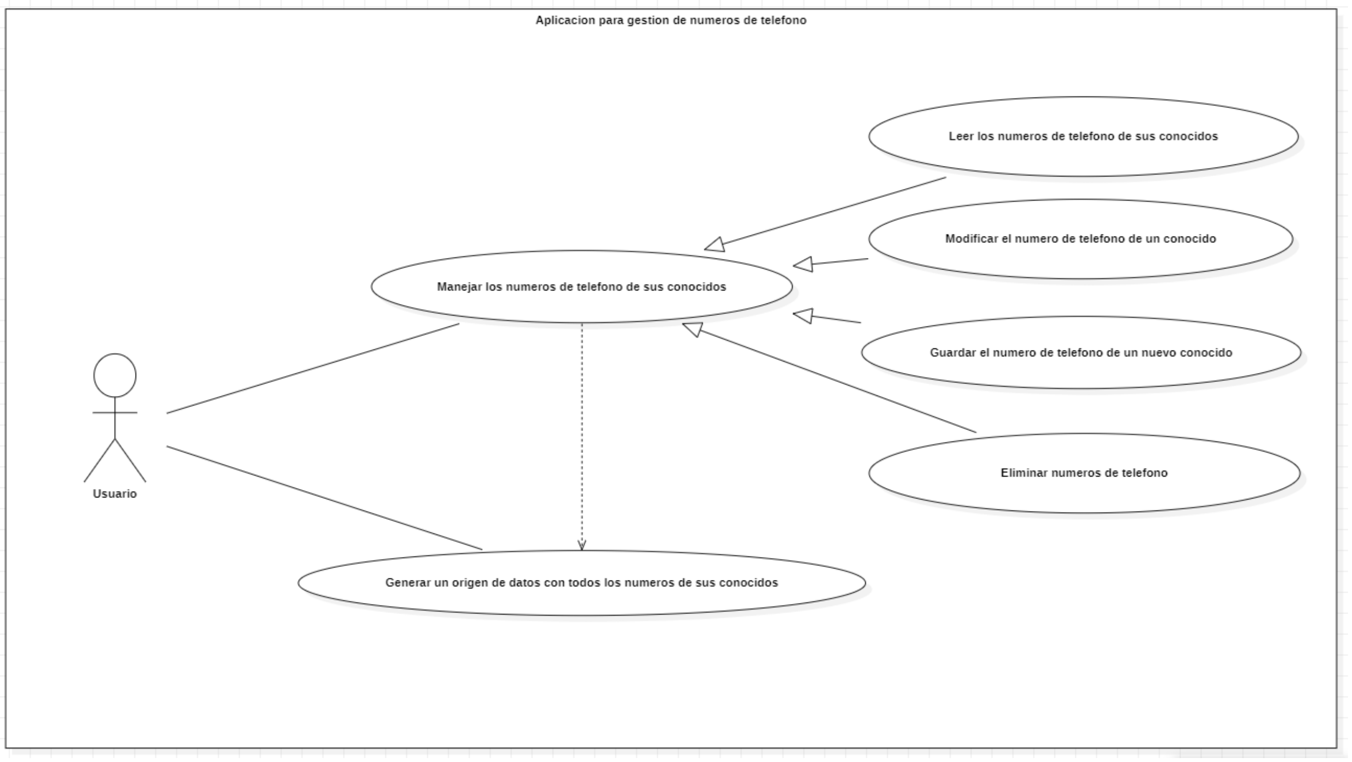

FileDatabase
-file: File
-fileHandle: RandomAccessFile
-filePath: String
+FileDatabase(filePath: String)
-read_file(): String
-write_file(new_content: String)
+connect()
+disconnect()
+query_all(): Model[]
+query(query_field: int, query_value: String): Model
+update(query_field: int, query_value: String, new_field: int, new_value: String)
+delete(query_field: int, query_value: String)
+insert(model: Model): boolean
Database
+db
1
1
UserRepository
-db: Database
+users: ArrayList<User>
-subscribers: ArrayList<Subscriber>
+UserRepository(db: Database)
+toString(): String
+create(name: String, number: Number)
+read(name: String): User
+update(name: String, number: Number)
+delete(name: String)
+notifySubscribers()
+registerSubscriber(subscriber: Subscriber)
Suscriptable
Model
+subscribers
1
1..*
1
1
+userRepository
UserForm
-userRepository: UserRepository
-dataField: JTextArea
-dataScroll: JScrollPane
-rootPanel: JPanel
-buttonPanel: JPanel
-formPanel: JPanel
-createButton: JButton
-readButton: JButton
-updateButton: JButton
-deleteButton: JButton
-nameField: JTextField
-numberField: JTextField
+UserForm(userRepository: UserRepository)
+init()
+Notify()
+actionPerformed(event: ActionEvent)
ActionListener
Subscriber
+users
1
1..*
User
+name: String
+number: Number
+User(line: String)
+User(name: String, number: Number)
+deserialize(line: String)
+serialize(): String

Diagrama de casos de uso


Aplicacion para gestion de numeros de telefono
Leer los numeros de telefono de sus conocidos
Modificar el numero de telefono de un conocido
Manejar los numeros de telefono de sus conocidos
Guardar el numero de telefono de un nuevo conocido
Eliminar numeros de telefono
Usuario
Generar un origen de datos con todos los numeros de sus conocidos