Ibn Tofail University - ENSAK

Program: Computer Engineering

Project Report

Java Application for Internship and Job Search in Morocco

Prepared by:

Rhemimet Meryem

Khaliqi Salma

Supervised by:

Prof. Dr. Moumen Aniss

Academic Year: 2024 – 2025

Acknowledgements

We would like to express our deep gratitude to our supervisor, Professor Doctor Aniss Moumen, for his attentive guidance, valuable advice, and constant support throughout the realization of this project. His scientific rigor, pedagogy, and availability were sources of inspiration and greatly facilitated the achievement of our objectives. We particularly appreciated his ability to guide us patiently, clarify complex concepts, and stimulate our critical thinking, allowing us to progress autonomously while remaining supervised. His enthusiasm for the discipline and academic rigor encouraged us to adopt a methodical and rigorous approach at every stage of the project, from defining objectives to analyzing results. Thanks to his relevant advice and insightful vision, we were able to overcome encountered difficulties and enrich both our technical and intellectual skills, making this experience a true springboard in our academic and professional journey.

Abstract

This project consists of developing a native Java application intended to facilitate the search for internships and job opportunities in Morocco. Faced with the dispersion of offers across several online platforms, the main objective of this application is to centralize professional opportunities and simplify access for students and job seekers. The application is based on a two-part architecture: a user part and an administrator part, accessible via a single login page. Roles are determined from the database: administrators have predefined credentials, while users can freely create their accounts. The user part allows account creation and management, secure authentication, password recovery by sending a code via email, as well as consultation, search, and sorting of offers by city and specialty. It also offers a personalized recommendation feature ("Offers for me") based on the user's profile and CV. In addition, the user can click on an offer to be redirected directly to the announcement website in order to apply. Data collection is ensured by a web scraping mechanism applied to several Moroccan job offer websites. Two sites are scraped using Selenium to handle dynamic content, while two others are processed with Jsoup for data extraction from static pages. The administrator part allows viewing general statistics and restarting the scraping process to update the database with new opportunities. This application thus provides an efficient, centralized, and scalable solution, improving access to professional opportunities and optimizing the job search experience.
Keywords: Java, Web Scraping, Selenium, Jsoup, MySQL, Recommendation, UML.

Introduction

Project context

The search for internships and job opportunities represents a key stage in the academic and professional path of students and young graduates. In Morocco, these opportunities are published on several specialized online platforms, which forces job seekers to consult different websites to find offers matching their profile. This dispersion of information makes the search long, complex, and sometimes inefficient.

Faced with this issue, the centralization of offers and the automation of their collection constitute a relevant solution to facilitate access to professional opportunities and optimize users' time.

Project Objectives

The main objective of this project is to design and develop an application that simplifies the search for internships and job opportunities. The specific objectives are:

- Centralize internship and job offers from multiple websites.
- Facilitate the search process through advanced sorting and search functionalities.
- Provide personalized job offers based on the user's profile and CV.
- Allow the user to click on an offer and be redirected directly to the job posting website to apply.
- Offer a single login page for all users, with role distinction managed from the database:
  - Administrators have predefined credentials inserted manually.
  - Users can freely create their own accounts.
- Automate the updating of job offers using web scraping.
- Provide a simple and intuitive interface for both users and administrators.

General Presentation of the Solution

This project consists of developing a native Java application dedicated to the search for internships and job opportunities. The application allows users to create an account, manage their profile and CV, consult and filter available job offers, and receive personalized recommendations.

- **User part:** consultation and sorting of offers, personalized recommendations, direct access to the job posting website to apply, profile and CV management.
- **Administrator part:** management of offer updates, restarting the scraping process, and consultation of general statistics.

The collection of job offers is carried out from several specialized websites using web scraping techniques with **Selenium** (for dynamic websites) and **Jsoup** (for static websites). This architecture ensures an efficient, centralized, and scalable solution, optimizing user experience and access to professional opportunities.

Contents

List of Figures

List of Tables

Chapter 1: Needs Analysis and Specification

1.1 Problem Statement

The search for internships and job opportunities constitutes an essential stage in the academic and professional path of students and young graduates. In Morocco, these opportunities are dispersed across several specialized websites, each having its own interface and publication criteria. This dispersion forces users to navigate between multiple platforms in order to find offers adapted to their profile and skills, making the search long, complex, and sometimes inefficient. Moreover, existing platforms do not always offer advanced filtering functionalities, intelligent search, or personalized recommendations, which limits users' ability to quickly identify the most relevant opportunities. Faced with this problem, the centralization of offers and the automation of their collection constitute a relevant solution to improve access to professional opportunities and facilitate the search process.

1.2 System Actors

The main actors of the application are:

1. User

The user has access to several functionalities to manage their account and consult offers:

- Create and manage a personal account.
- Log in and log out securely via the single login page.
- Recover a forgotten password via email.
- Consult all available offers.
- Search and sort offers by city and by specialty.
- Receive personalized recommendations based on the profile and CV.
- Manage their profile and CV.
- Click on an offer to be redirected directly to the job posting website in order to apply.

2. Administrator

The administrator has privileges to manage the database and the offer collection process:

- Log in via the single login page (predefined credentials manually inserted into the database).
- Launch the web scraping process to retrieve new offers.
- Update the database with new offers.
- Consult general statistics on users and offers.

1.3 Functional Requirements

Functional requirements describe what the system must accomplish to meet the expectations of users and the administrator.

For the user

The system must allow the user to:

- Create an account and log in securely.
- Recover a forgotten password via email.
- Consult all available offers.
- Sort offers by city and by specialty.
- Search for an offer by keyword.
- View recommended offers ("Offers for me").
- Manage their profile (change password, CV).
- Log out.
- Access the details of an offer and be redirected to the job posting website to apply.

For the administrator

The system must allow the administrator to:

- Launch scraping to retrieve new offers.
- Update the database with recent offers.
- Consult global statistics on offers and users.

1.4 Non-Functional Requirements

Non-functional requirements describe how the system must operate to ensure quality, security, and performance.

- **Security:** protection of users' personal information (password, email, CV).
- **Reliability:** functionalities must remain operational even in case of changes to scraped websites.
- **Performance:** search, sorting, and recommendations must be fast and responsive.
- **Usability:** the application must be intuitive and easy to use for all user profiles.
- **Scalability:** ability to easily add new websites for scraping or new functionalities.

1.5 Constraints

Constraints represent the technical, organizational, and legal limitations of the project.

Technical

The project must comply with the following technical constraints:

- Development in Java SE.
- Use of Selenium to scrape dynamic websites and Jsoup for static websites.
- Integration of a database to store users, CVs, and offers.
- Sending emails via SMTP for password recovery.

Organizational

The organizational constraints are:

- The project must be delivered within the allotted timeframe.
- The interface must be simple, intuitive, and suitable for users and administrators.

Legal

Legal constraints include:

- Compliance with the terms of use of scraped websites.
- Protection of users' personal data.

This needs analysis constitutes the basis for the design and UML modeling of the system, presented in the following chapter.

Chapter 2: UML Modeling of the System

2.1 Chapter Introduction

In this chapter, we present the design and modeling phase of the application using the UML (Unified Modeling Language). This phase aims to translate the functional and technical requirements identified in the previous chapter into clear and structured graphical models, thus facilitating the understanding of the system's operation. UML modeling makes it possible to represent both the static structure and the dynamic behavior of the application. It facilitates communication between the different stakeholders of the project and serves as a basis for Java implementation. In this project, several UML diagrams were created in order to describe the system functionalities, interactions between actors and components, as well as the internal organization of classes. The diagrams presented in this chapter include: the use case diagram, the sequence diagram, the activity diagram, and the class diagram.

2.2 Use Case Diagram

The use case diagram represents the main functionalities offered by the system and the roles of the involved actors: the user and the administrator. It provides a global view of the system by illustrating the actions that each actor can perform.

For the user

The functionalities accessible to the user are as follows:

- Account creation and management.
- Secure login and logout.
- Forgotten password recovery.
- Consultation, sorting, and searching of offers.
- Profile and CV management.
- Application via redirection to the external job posting website.

For the administrator

The functionalities accessible to the administrator are:

- Launching the scraping process to retrieve new offers.
- Consulting general statistics on users and offers.

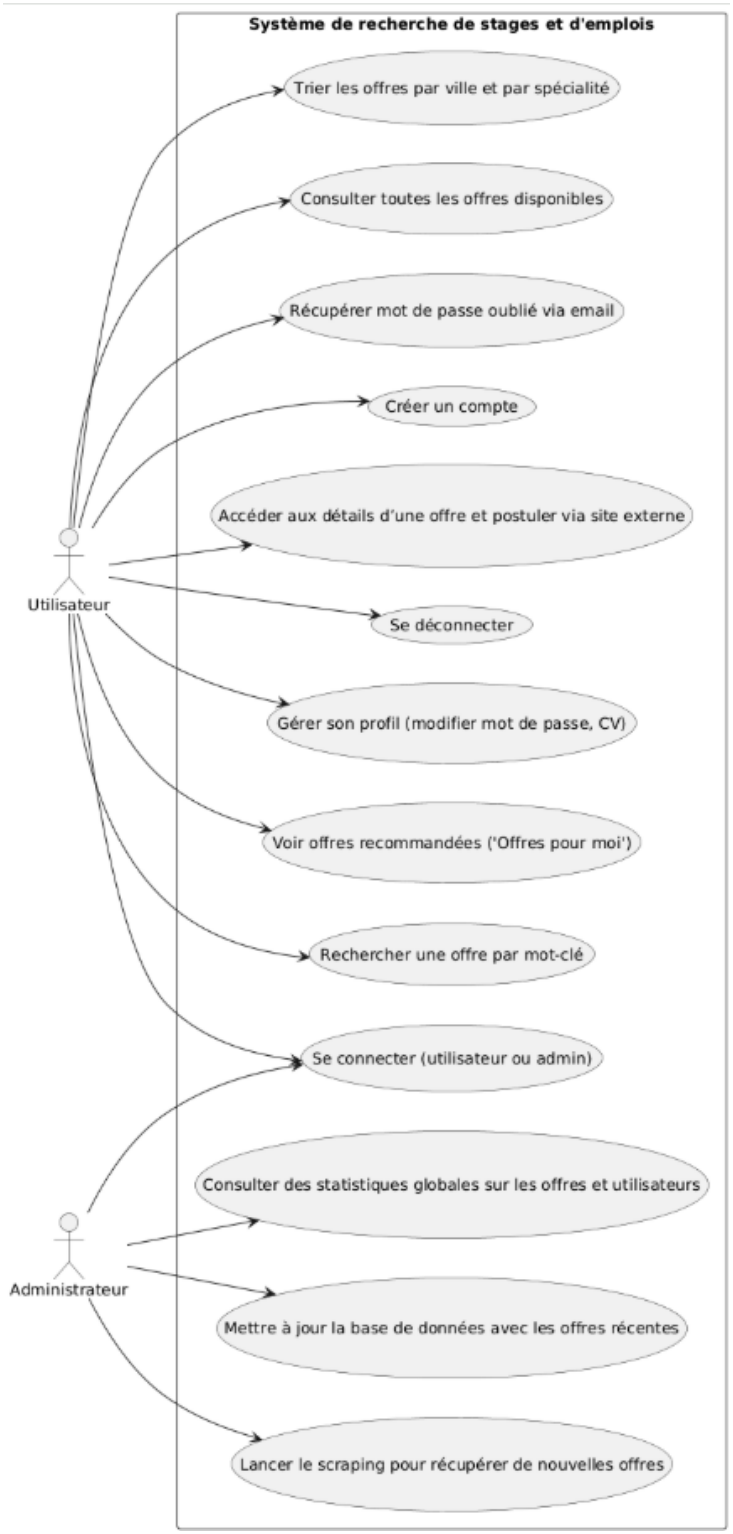


Figure 2.1: Use Case Diagram

2.3 Class Diagram

The class diagram represents the static structure of the application, by highlighting the classes, their attributes, their methods, and the relationships between them. In this project, this diagram illustrates the main entities: users, job offers, CVs, as well as authentication, scraping, and email sending services. It ensures that the design matches the Java implementation and facilitates maintenance and future evolution of the system.

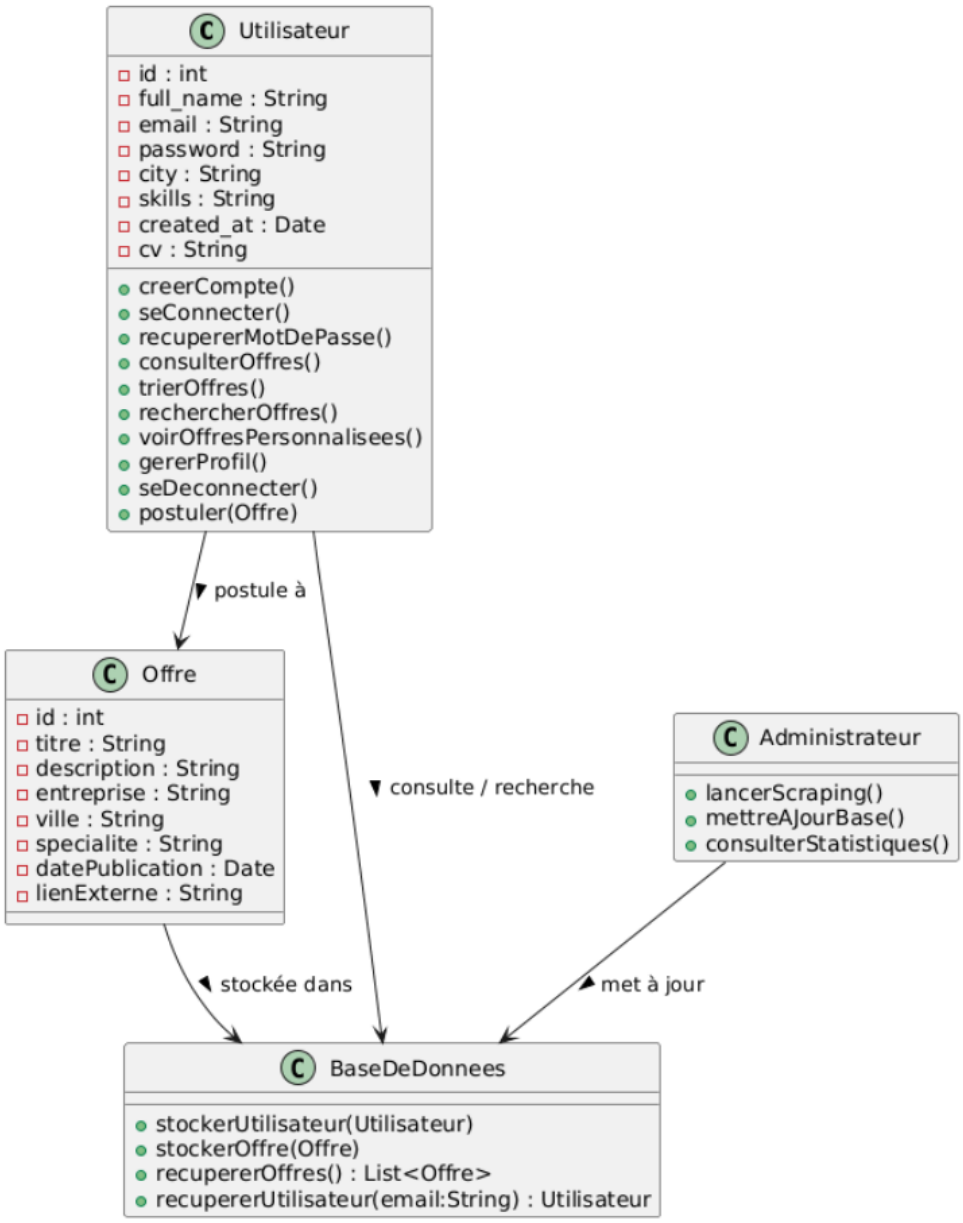


Figure 2.2: Class Diagram

2.4 Sequence Diagram

The sequence diagram describes the temporal flow of interactions between the different system components during the execution of a specific scenario. It highlights the order in which messages are exchanged between actors, the user interface, business logic, and external services, such as email sending.

Within the scope of this project, the sequence diagrams notably illustrate the following scenarios:

- User or administrator login.
- Password recovery via email.
- Consultation of an offer by the user and redirection to the external website to apply.
- Launching the scraping process and updating the database by the administrator.

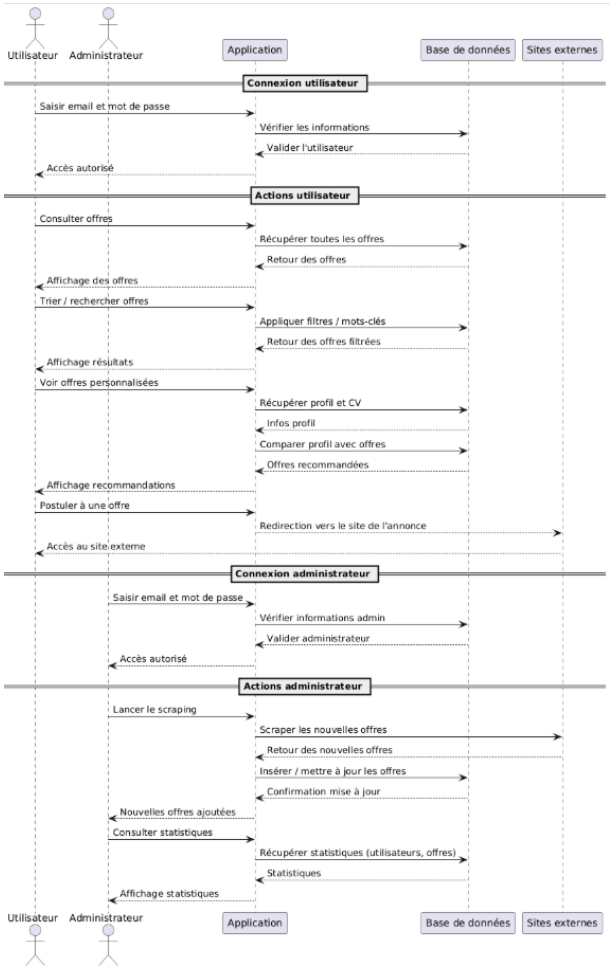


Figure 2.3: Sequence Diagram

2.5 Activity Diagram

The activity diagram models processing flows and the sequence of actions within the system. It allows clear representation of business processes, from the triggering of an action to its completion.

Within the scope of this project, the activity diagram notably illustrates the following processes:

- The process of searching, sorting, and consulting offers by the user.
- Consultation of offer details and redirection to the external website to apply.
- Authentication management and password recovery.
- The scraping process and offer updating by the administrator.

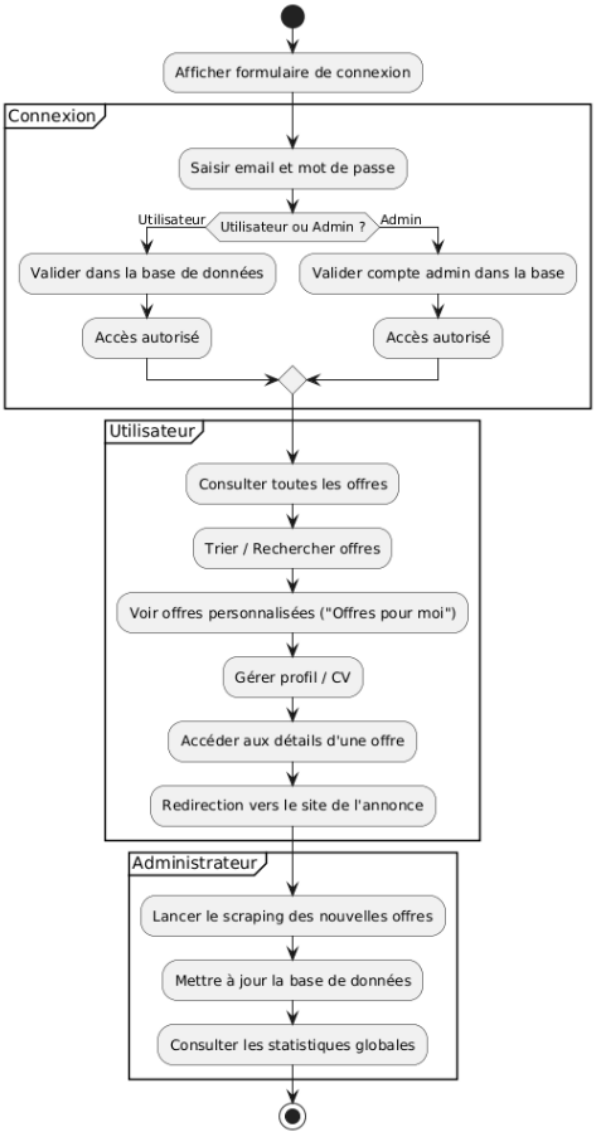


Figure 2.4: Activity Diagram

Chapter 3: Application Implementation

3.1 Technologies Used

In this project, several technologies and libraries were used to develop the application:
Note: FTAltaf was used to improve the appearance of the interface, by adding visual elements and modern styles so that the application is more intuitive and pleasant for the user.

```
http://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation with catalog)
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example.jobs</groupId>
    <artifactId>maven_exemple</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
    </properties>

    <dependencies>
        <!-- JUnit pour tests -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>

        <!-- Selenium Java -->
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>3.141.59</version>
        </dependency>
```

Figure 3.1: pom.xml

Technology / Library	**Usage in the project**
Java SE	Main language for developing the native application
Eclipse + Maven	IDE and dependency management of the project
Selenium	Scraping dynamic websites generated by JavaScript
Jsoup	Scraping static websites (HTML)
MySQL	Storage of users, CVs, and job offers
JavaMail (SMTP)	Sending codes for password recovery
Apache PDFBox / Apache POI	Reading and manipulation of PDF and Word files (CVs)
Logback	Logging and tracking of application events
Weka	For recommendation functionalities based on profile and skills
FTAltaf	Decoration and improvement of the graphical interface to make the application more attractive and ergonomic

Table 3.1: Technologies and libraries used in the project

3.2 Web Scraping

Web scraping is the process of automatically collecting data from websites. In this project, it is used to retrieve job offers from different Moroccan websites.

3.2.1 Dynamic and static websites

- **Dynamic websites:** their content is generated by JavaScript. To scrape them, **Selenium** is used to simulate a real browser and retrieve the content after rendering.
- **Static websites:** the content is directly in the HTML. For these websites, **Jsoup** is used to parse and extract information.

3.2.2 Duplicate management

Before inserting a job offer into the database, the scraper checks whether the link already exists in order to avoid duplicates and ensure data consistency.

3.2.3 General scraping flow

The general scraping flow in this project is carried out in several steps:

1. Launching scraping.
2. Browsing pages and retrieving job offer links.
3. Extracting job offer details (title, company, location, description, dates, etc.).
4. Data cleaning (filling missing values with default values).
5. Duplicate verification.
6. Inserting offers into the database.

3.2.4 Scraping code example

Example 1: MarocAnnonces (Jsoup)

```java
package com.example.jobs.maven_exemple;

import org.jsoup.Jsoup;

public class MarocAnnoncesScraper {

    private static final String BASE_URL =
            "https://www.marocannonces.com/categorie/309/Emploi/Offres-emploi.html";
    private static final int MAX_PAGES = 20; // Ajustable

    public static void main(String[] args) {

        Set<String> visitedLinks = new HashSet<>();

        try {
            for (int page = 1; page <= MAX_PAGES; page++) {

                String url = page == 1
                        ? BASE_URL
                        : BASE_URL.replace(".html", "/" + page + ".html");

                System.out.println("🔍 Scraping page MarocAnnonces: " + url);

                Document doc = Jsoup.connect(url)
                        .userAgent("Mozilla/5.0")
                        .timeout(10000)
                        .get();

                Elements jobLinks = doc.select("a[href*='/Offres-emploi/annonce/']");

                if (jobLinks.isEmpty()) {
                    System.out.println("⛔ Page vide → arrêt");
                    break;
                }

                for (Element linkEl : jobLinks) {
```

Figure 3.2: Maroc Annonces

Example 2: Rekrute.com (Selenium)

```java
package com.example.jobs.maven_exemple;

import org.openqa.selenium.*;

public class RekruteScraper {

    // ===== UTILITAIRE =====
    private static String valeurParDefaut(String value, String defaut) {
        return (value == null || value.trim().isEmpty()) ? defaut : value.trim();
    }

    public static void main(String[] args) {

        System.setProperty("webdriver.chrome.driver",
                "C:\\Users\\SALMA.KH\\Downloads\\chromedriver.exe");

        WebDriver driver = new ChromeDriver();
        WebDriverWait wait = new WebDriverWait(driver, 15);

        try {
            int page = 1;
            int maxPages = 158;

            while (page <= maxPages) {

                driver.get("https://www.rekrute.com/offres-emploi-maroc.html?p=" + page);
                wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("post-data")));

                List<WebElement> jobs =
                        driver.findElements(By.cssSelector("#post-data li.post-id"));

                for (WebElement job : jobs) {

                    try {
                        // ===== TITRE + LIEN =====
                        WebElement aTag =
                                job.findElement(By.cssSelector("h2 a.titreJob"));
```

Figure 3.3: Rekrute

3.2.5 Best practices

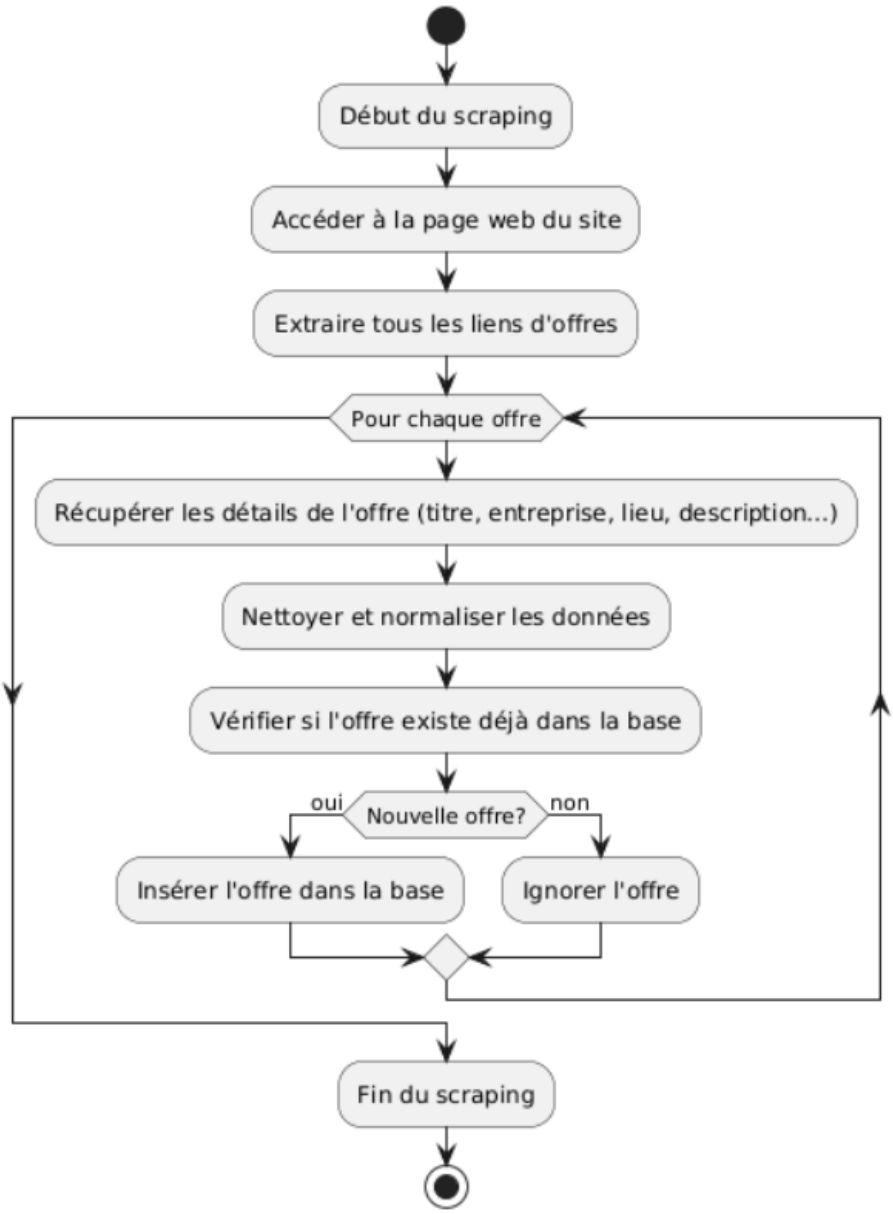
- Add pauses (`Thread.sleep`) to avoid overloading the website and prevent IP blocking.
- Handle exceptions to ignore invalid offers or connection errors.
- Normalize data and fill empty fields with default values to maintain information quality.

3.2.6 Scraping activity diagram

The activity diagram below illustrates the general scraping flow:

- Access to web pages
- Extraction of job offer links
- Browsing offers one by one
- Retrieval of details (title, company, location...)

- Data cleaning and normalization
- Duplicate verification in the database
- Database insertion if the offer is new
- End of the process



3.3 User Part

3.3.1 General description

The user part allows a candidate to manage their account, consult and apply for job or internship offers, and receive personalized recommendations. The interface is implemented

using **Java Swing** and communicates with the database through the `Database` class.

3.3.2 Account creation

- **Class:** `RegisterFrame`

```java
package application;

import javax.swing.*;

public class RegisterFrame extends JFrame {

    private JTextField txtFullName, txtEmail, txtCity, txtSkills;
    private JPasswordField txtPassword;
    private JButton btnRegister, btnLogin, btnImportCV;
    private byte[] cvBytes = null; // Contenu du CV pour la BDD

    public RegisterFrame() {
        setTitle("Créer un compte");
        setSize(400, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        // Titre
        JLabel lblTitle = new JLabel("Créer un compte");
        lblTitle.setFont(new Font("Segoe UI", Font.BOLD, 24));
        lblTitle.setBounds(100, 30, 250, 40);
        add(lblTitle);

        // Champs texte
        txtFullName = new JTextField();
        txtFullName.setBounds(50, 90, 300, 40);
        txtFullName.setBorder(BorderFactory.createTitledBorder("Nom complet"));
        add(txtFullName);

        txtEmail = new JTextField();
        txtEmail.setBounds(50, 150, 300, 40);
        txtEmail.setBorder(BorderFactory.createTitledBorder("Email"));
        add(txtEmail);

        txtPassword = new JPasswordField();
```

Figure 3.4: RegisterFrame

3.3.3 Account creation

- **Class:** `RegisterFrame`
- **Functionalities:**
  - Fill in the form with full name, email, password, city, skills
  - Import a CV in PDF or Word (`cvBytes`)
  - Check whether the email already exists

  - Registration in the database via `Database.registerUser()`

- **Interface:** text fields + "Sign up" and "Login" buttons

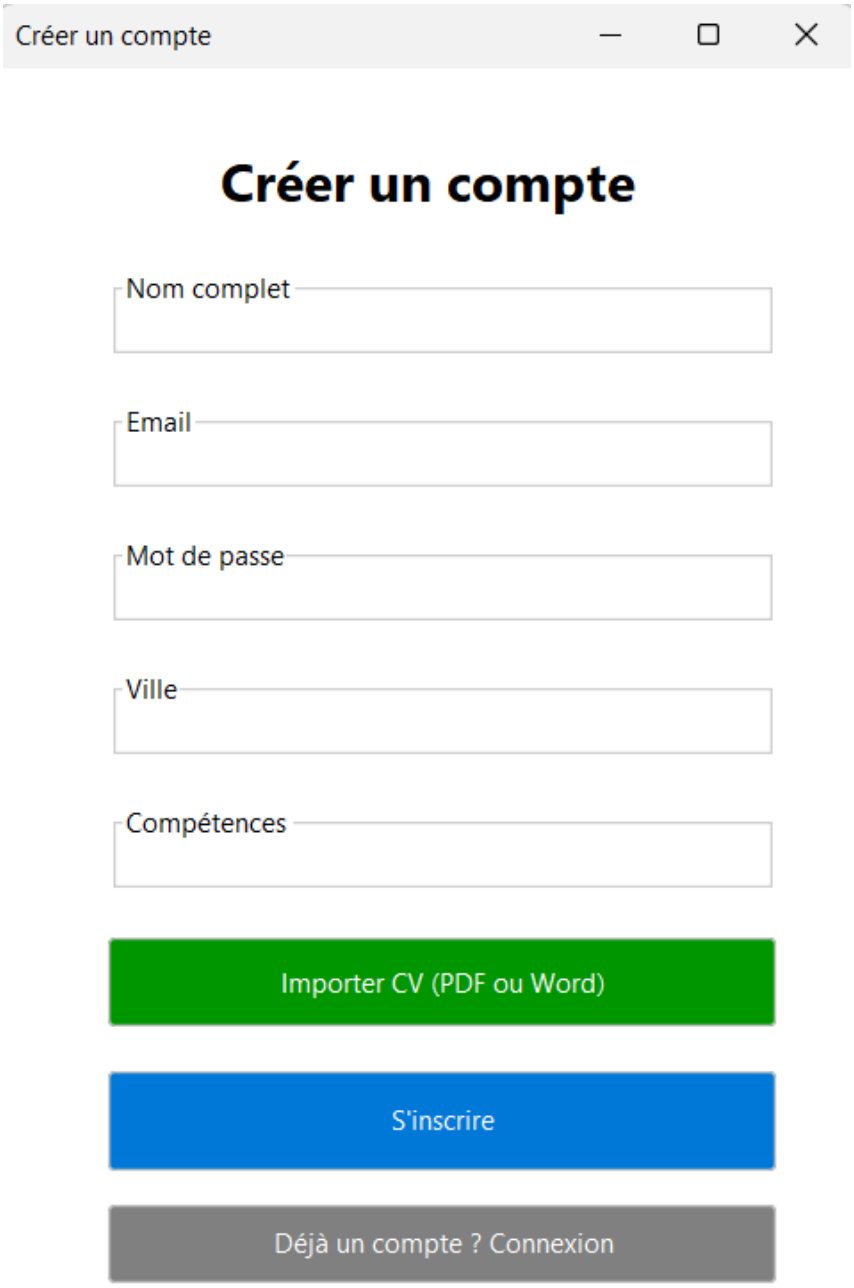


Figure 3.5: Create an account

Technical explanation

The CV is converted into `byte[]` in order to be stored in the database as a **BLOB**.

3.3.4 Login / logout

- **Class:** `LoginFrame`

```java
package application;

import javax.swing.*;

public class LoginFrame extends JFrame {

    private JTextField txtEmail;
    private JPasswordField txtPassword;
    private JButton btnLogin, btnRegister, btnForgot;

    public LoginFrame() {
        setTitle("Connexion");
        setSize(400, 450);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        JLabel lblTitle = new JLabel("Bienvenue");
        lblTitle.setFont(new Font("Segoe UI", Font.BOLD, 28));
        lblTitle.setBounds(120, 30, 200, 40);
        add(lblTitle);

        txtEmail = new JTextField();
        txtEmail.setBounds(50, 100, 300, 40);
        txtEmail.setFont(new Font("Segoe UI", Font.PLAIN, 16));
        txtEmail.setBorder(BorderFactory.createTitledBorder("Email"));
        add(txtEmail);

        txtPassword = new JPasswordField();
        txtPassword.setBounds(50, 160, 300, 40);
        txtPassword.setFont(new Font("Segoe UI", Font.PLAIN, 16));
        txtPassword.setBorder(BorderFactory.createTitledBorder("Mot de passe"));
        add(txtPassword);

        btnLogin = new JButton("Se connecter");
        btnLogin.setBounds(50, 220, 300, 45);
```

Figure 3.6: LoginFrame

- **Functionalities:**
  - Enter email and password
  - Verification with `Database.loginUser()`
  - Redirection to `HomeFrame` if login is successful
  - Buttons to create an account or recover password
- **Interface:** email and password fields + “Log in”, “Create an account”, “Forgot password?” buttons

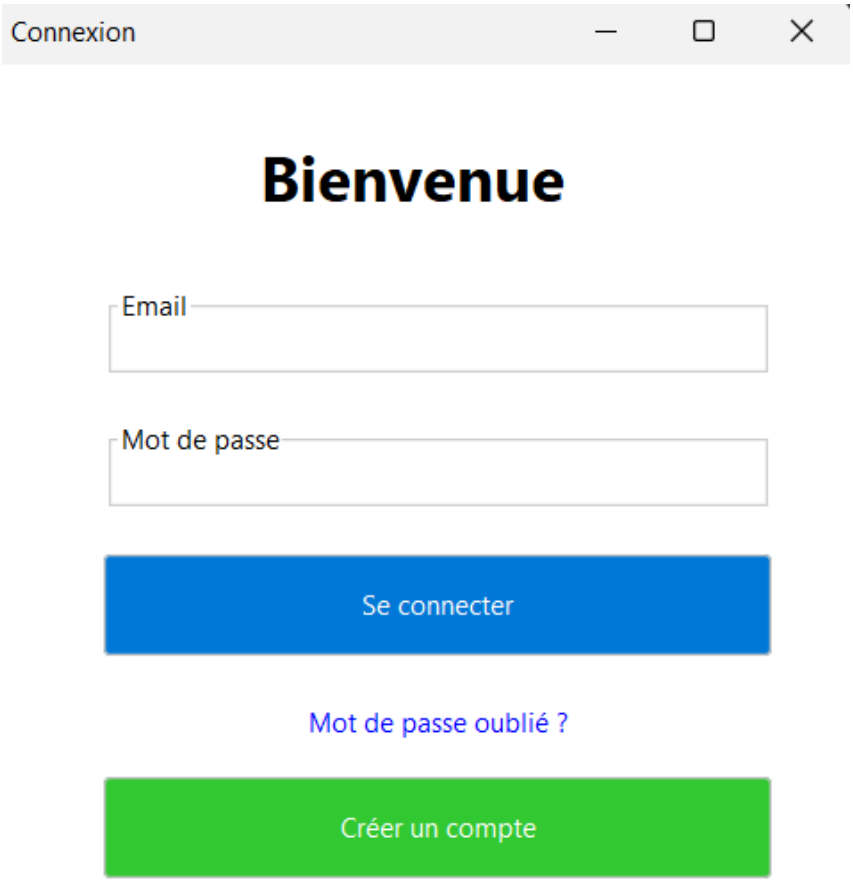


Figure 3.7: Login / logout

3.3.5 Password recovery

- **Involved classes:**
  - `ForgotPasswordFrame`: email input

```java
package application;

import javax.swing.*;

public class ForgotPasswordFrame extends JFrame {
    private JTextField txtEmail;
    private JButton btnSendCode, btnLogin;

    public ForgotPasswordFrame() {
        setTitle("Mot de passe oublié");
        setSize(400, 250);
        setLayout(null);
        setLocationRelativeTo(null);
        getContentPane().setBackground(Color.WHITE);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel lblTitle = new JLabel("Mot de passe oublié");
        lblTitle.setFont(new Font("Segoe UI", Font.BOLD, 20));
        lblTitle.setBounds(80, 20, 300, 30);
        add(lblTitle);

        txtEmail = new JTextField();
        txtEmail.setBounds(50, 70, 300, 40);
        txtEmail.setBorder(BorderFactory.createTitledBorder("Email"));
        add(txtEmail);

        btnSendCode = new JButton("Envoyer le code");
        btnSendCode.setBounds(50, 120, 300, 40);
        btnSendCode.setBackground(new Color(0, 120, 215));
        btnSendCode.setForeground(Color.WHITE);
        btnSendCode.setFocusPainted(false);
        add(btnSendCode);

        btnLogin = new JButton("Retour à la connexion");
        btnLogin.setBounds(50, 170, 300, 30);
        btnLogin.setBorderPainted(false);
        btnLogin.setContentAreaFilled(false);
```

Figure 3.8: ForgotPasswordFrame

- ForgotPasswordCodeFrame : input of the code received by email

```java
package application;

import javax.swing.*;

public class ForgotPasswordCodeFrame extends JFrame {
    private JTextField txtCode;
    private JButton btnValidate, btnLogin;
    private String email;

    public ForgotPasswordCodeFrame(String email) {
        this.email = email;
        setTitle("Entrer le code");
        setSize(400, 250);
        setLayout(null);
        setLocationRelativeTo(null);
        getContentPane().setBackground(Color.WHITE);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel lblTitle = new JLabel("Entrer le code reçu");
        lblTitle.setFont(new Font("Segoe UI", Font.BOLD, 20));
        lblTitle.setBounds(80, 20, 300, 30);
        add(lblTitle);

        txtCode = new JTextField();
        txtCode.setBounds(50, 70, 300, 40);
        txtCode.setBorder(BorderFactory.createTitledBorder("Code"));
        add(txtCode);

        btnValidate = new JButton("Valider");
        btnValidate.setBounds(50, 120, 300, 40);
        btnValidate.setBackground(new Color(50, 200, 50));
        btnValidate.setForeground(Color.WHITE);
        btnValidate.setFocusPainted(false);
        add(btnValidate);

        btnLogin = new JButton("Retour à la connexion");
        btnLogin.setBounds(50, 170, 300, 30);
```

Figure 3.9: ForgotPasswordCodeFrame

- ForgotPasswordNewPasswordFrame : new password

```java
package application;

import javax.swing.*;

public class ForgotPasswordNewPasswordFrame extends JFrame {
    private JPasswordField txtNewPassword;
    private JButton btnSubmit, btnLogin;
    private String email;

    public ForgotPasswordNewPasswordFrame(String email) {
        this.email = email;
        setTitle("Nouveau mot de passe");
        setSize(400, 250);
        setLayout(null);
        setLocationRelativeTo(null);
        getContentPane().setBackground(Color.WHITE);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel lblTitle = new JLabel("Entrer un nouveau mot de passe");
        lblTitle.setFont(new Font("Segoe UI", Font.BOLD, 18));
        lblTitle.setBounds(50, 20, 300, 30);
        add(lblTitle);

        txtNewPassword = new JPasswordField();
        txtNewPassword.setBounds(50, 70, 300, 40);
        txtNewPassword.setBorder(BorderFactory.createTitledBorder("Nouveau mot de passe"));
        add(txtNewPassword);

        btnSubmit = new JButton("Modifier");
        btnSubmit.setBounds(50, 120, 300, 40);
        btnSubmit.setBackground(new Color(50, 200, 50));
        btnSubmit.setForeground(Color.WHITE);
        btnSubmit.setFocusPainted(false);
        add(btnSubmit);

        btnLogin = new JButton("Retour à la connexion");
        btnLogin.setBounds(50, 170, 300, 30);
```

Figure 3.10: ForgotPasswordNewPasswordFrame

3.3.6 Password recovery

- **Involved classes:**
  - `ForgotPasswordFrame`: email input
- **Functionalities:**
  - Generation of a reset code (`Database.generateResetCode`)
  - Code verification (`Database.verifyResetCode`)
  - Password update (`Database.updatePassword`)
  - Simulated email sending via `EmailSender` (SMTP)
- **Technical explanation:**
  - Secure flow to avoid changing the password without verification
  - Simple interface with Swing and clear navigation buttons

To better illustrate the functioning of password recovery, the following screenshots present the different stages of the user interface, from email input to defining a new password.

Step 1: Email input

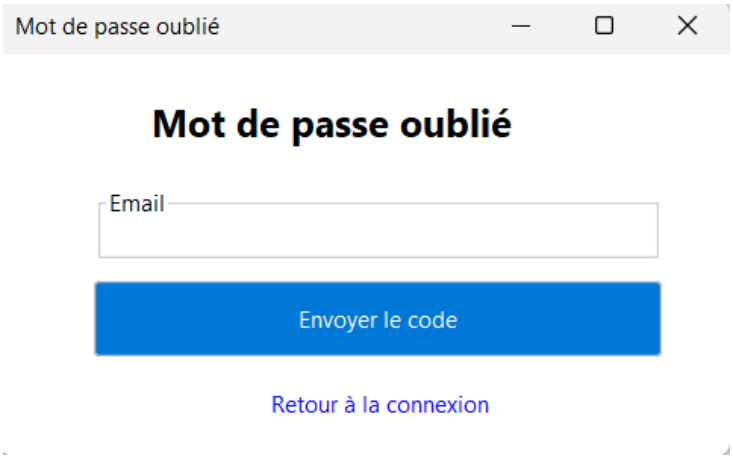


Figure 3.11: Email input

Step 2: Code input

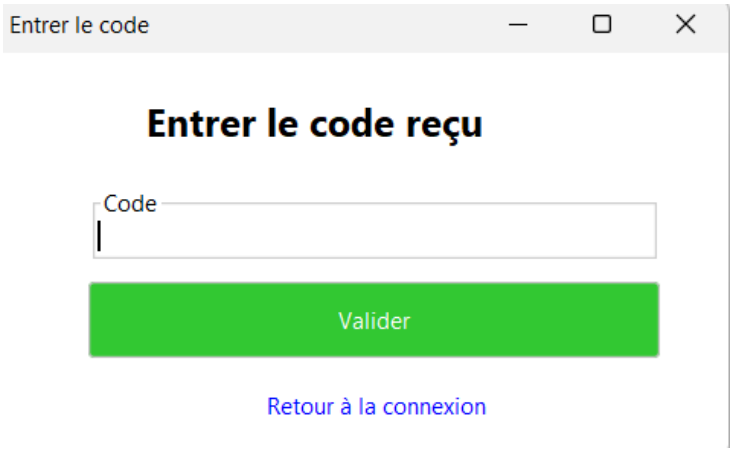


Figure 3.12: Code input

Step 3: New password

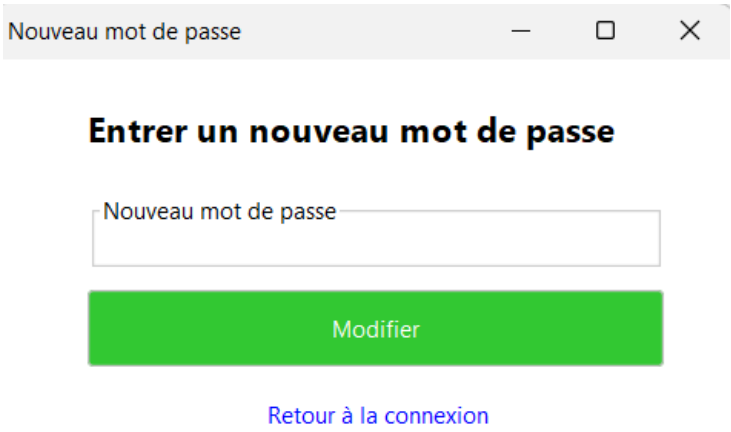


Figure 3.13: New password

3.3.7 Profile management

- **Class:** `ProfileFrame`

```java
package application;

import javax.swing.*;

public class ProfileFrame extends JFrame {

    private JTextField txtFullName, txtCity, txtSkills;
    private JLabel lblEmail, lblCV;
    private JButton btnSave, btnChangePassword, btnImportCV;
    private String email;
    private byte[] cvBytes = null; // stocker le CV en BLOB

    public ProfileFrame(String email) {
        this.email = email;

        setTitle("Mon profil");
        setSize(400, 550);
        setLocationRelativeTo(null);
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        JLabel lblTitle = new JLabel("Mon profil");
        lblTitle.setFont(new Font("Segoe UI", Font.BOLD, 24));
        lblTitle.setBounds(120, 20, 200, 40);
        add(lblTitle);

        // Récupérer infos utilisateur
        String fullName = Database.getUserName(email);
        String city = Database.getUserCity(email);
        String skills = Database.getUserSkills(email);
        cvBytes = Database.getUserCV(email);

        lblEmail = new JLabel("Email : " + email);
        lblEmail.setBounds(50, 80, 300, 25);
        add(lblEmail);

        txtFullName = new JTextField(fullName);
```

Figure 3.14: ProfileFrame

- **Functionalities:**
  - Display and modify: full name, city, skills
  - Import or replace the CV
  - Change the password directly from the profile
  - Save changes in the database (`Database.updateUserProfile`)
- **Interface:** text fields, labels for email and CV, “Import CV”, “Change password”, “Save” buttons

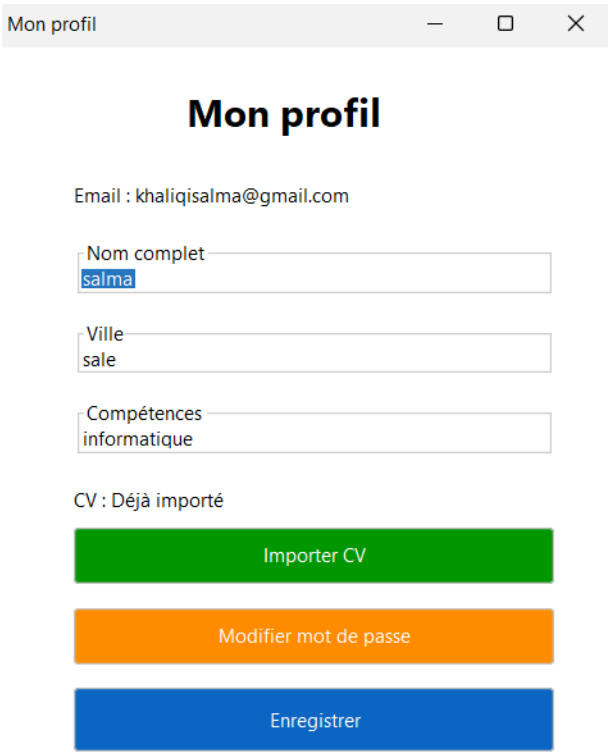


Figure 3.15: My profile

3.3.8 Consultation and search of offers

- **Class:** `HomeFrame`

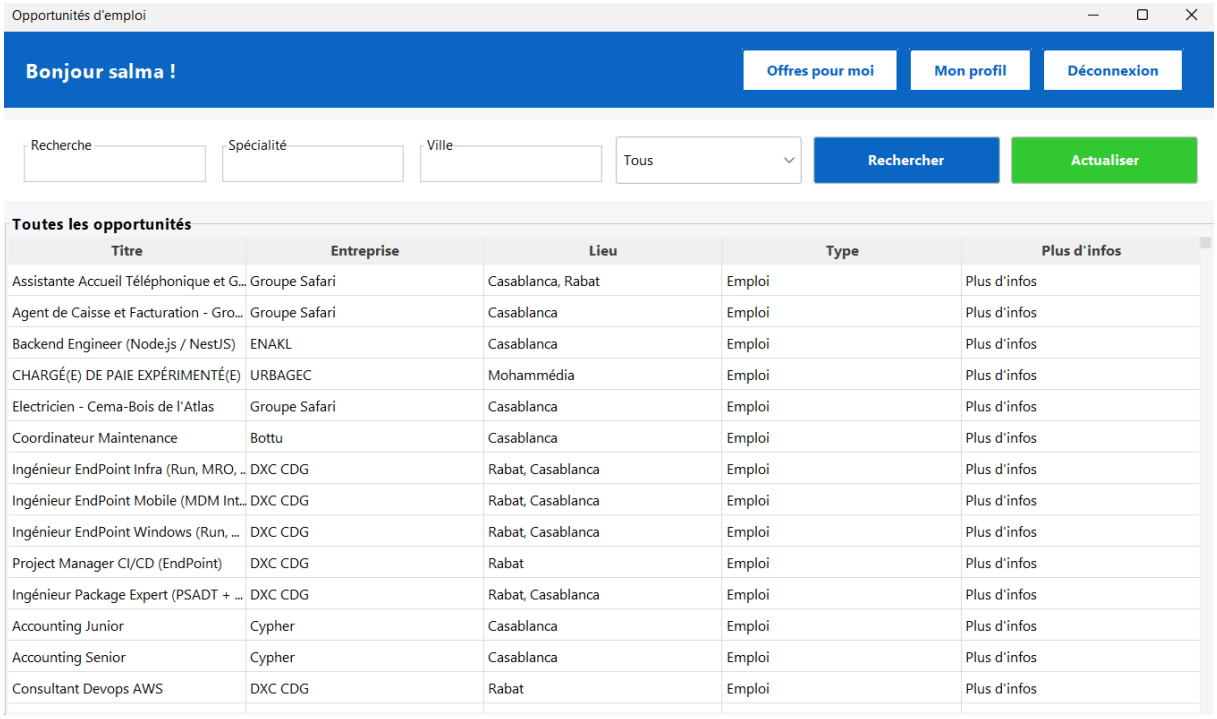
```
1  package application;
2
3  import javax.swing.*;
15
16 public class HomeFrame extends JFrame {
17
18     private JTextField txtVille, txtSpecialite, txtSearch;
19     private JTable table;
20     private DefaultTableModel model;
21     private JComboBox<String> cmbType;
22     private Map<Integer, String> liensMap = new HashMap<>();
23
24     public HomeFrame(String email) {
25
26         setTitle("Opportunités d'emploi");
27         setSize(1100, 650);
28         setLocationRelativeTo(null);
29         setDefaultCloseOperation(EXIT_ON_CLOSE);
30         setLayout(new BorderLayout(10,10));
31         getContentPane().setBackground(new Color(245,246,248));
32
33         /* ================ HEADER ================ */
34       // Bouton Offres pour moi
35         JButton btnOpportunities = new JButton("Offres pour moi");
36         btnOpportunities.setFocusPainted(false);
37         btnOpportunities.setBackground(Color.WHITE);
38         btnOpportunities.setForeground(new Color(10,102,194));
39         btnOpportunities.setFont(new Font("Segoe UI", Font.BOLD, 13));
40         btnOpportunities.setCursor(new Cursor(Cursor.HAND_CURSOR));
41         btnOpportunities.setBorder(BorderFactory.createCompoundBorder(
42                 new LineBorder(new Color(10,102,194), 2, true),
43                 BorderFactory.createEmptyBorder(8, 20, 8, 20)
```

Figure 3.16: HomeFrame

3.3.9 Consultation and search of offers

- **Class:** `HomeFrame`
- **Functionalities:**
  - Display of offers in table form with columns: title, company, location, type
  - Search by city, specialty, and type (internship/job)
  - Filters using `JTextField` and `JComboBox`
  - Refresh with "Refresh" button
  - Clicking on "More info" opens the offer link in the browser
- **Technical explanation:**
  - Offers are retrieved via `Database.searchAllJobs()`
  - Application-side filtering by city, specialty, and type
  - Simple interface with `JTable` and clear interaction buttons

To illustrate the consultation and search interface for offers, the following screenshots show the offers table, available filters, and interaction with selected offers.



3.3.10 Recommended offers ("Offers for me")

- **Class:** `OpportunitiesFrame`

```java
package application;

import com.example.jobs.maven_exemple.*;

public class OpportunitiesFrame extends JFrame {

    private static final double THRESHOLD = 0.35;

    public OpportunitiesFrame(String email) throws Exception {

        setTitle("Opportunités ML");
        setSize(1000, 600); // un peu plus grand pour les filtres
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        // Panel principal
        JPanel mainPanel = new JPanel(new BorderLayout(10,10));
        add(mainPanel);

        // Panel filtres et boutons
        JPanel filterPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 10, 5));

        filterPanel.add(new JLabel("Ville:"));
        JTextField cityField = new JTextField(15);
        filterPanel.add(cityField);

        filterPanel.add(new JLabel("Type:"));
        JComboBox<String> typeCombo = new JComboBox<>(new String[]{"Tous", "Stage", "Emploi"});
        filterPanel.add(typeCombo);

        JButton searchBtn = new JButton("Rechercher");
        searchBtn.setBackground(new Color(10,102,194));
        searchBtn.setForeground(Color.WHITE);
        searchBtn.setFocusPainted(false);
        filterPanel.add(searchBtn);

        JButton refreshBtn = new JButton("Actualiser");
```

Figure 3.17: OpportunitiesFrame

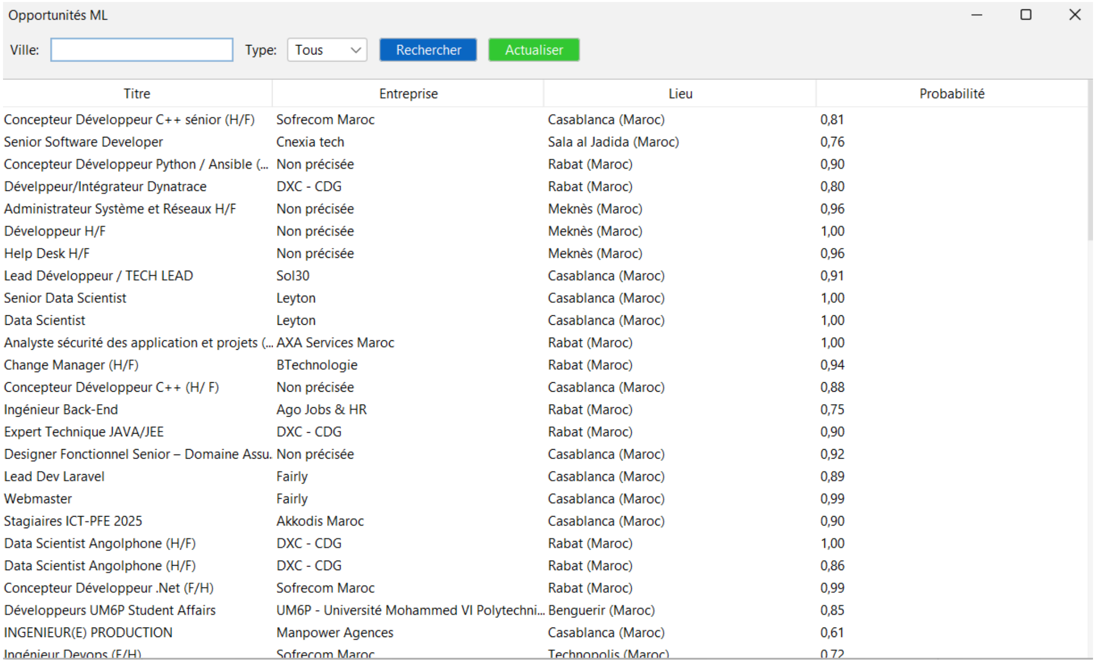
3.3.11 Recommended Offers (“Offers for Me”)

- **Class:** `OpportunitiesFrame`
- **Features:**
  - Machine learning–based recommendations (Weka model)
  - Extraction of candidate and job offer skills (`JobMatcher.extractSkills`)
  - Calculation of the matching probability for each offer
  - Only offers with a probability ≥ 0.35 are displayed
  - Clicking on a row opens the job offer link
- **Technical explanation:**
  - For each user and each offer, a feature vector is created by comparing user skills with job offer skills (`JobMatcher.buildFeatureVector`)

  - The pre-trained Weka model (`model.model`) predicts compatibility and computes the matching probability
  - The interface is dynamically updated using `JTable` to display only relevant offers

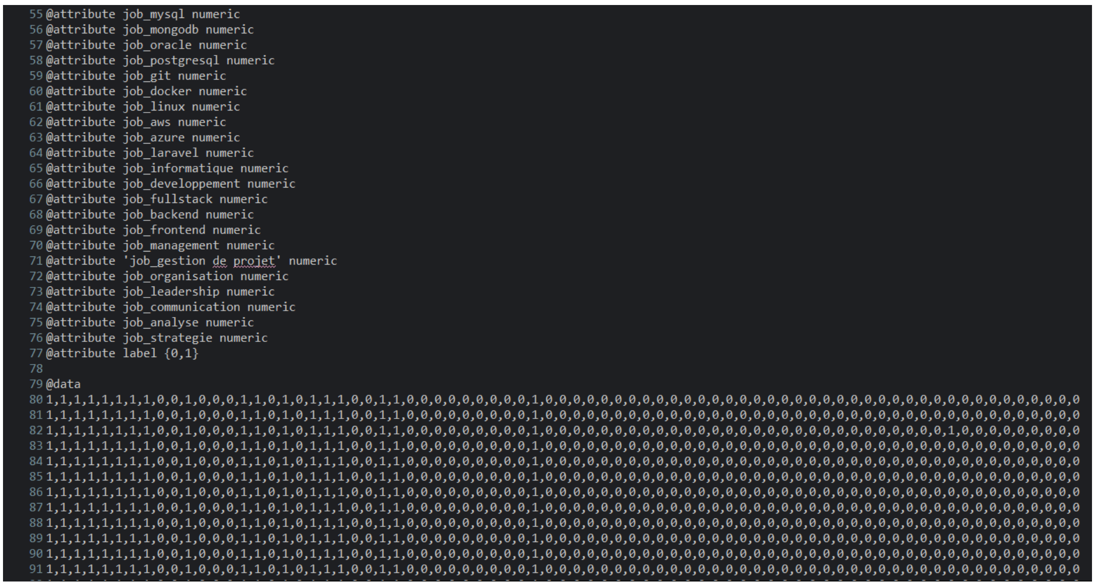
Interface illustration: To illustrate this feature, the following screenshots present:

- the table of recommended offers, filtered according to the compatibility probability computed by the model,
- user interaction allowing a click on an offer to directly access the corresponding link.



Titre	Entreprise	Lieu	Probabilité
Concepteur Développeur C++ sénior (H/F)	Sofrecom Maroc	Casablanca (Maroc)	0,81
Senior Software Developer	Cnexia tech	Sala al Jadida (Maroc)	0,76
Concepteur Développeur Python / Ansible (...	Non précisée	Rabat (Maroc)	0,90
Développeur/Intégrateur Dynatrace	DXC - CDG	Rabat (Maroc)	0,80
Administrateur Système et Réseaux H/F	Non précisée	Meknès (Maroc)	0,96
Développeur H/F	Non précisée	Meknès (Maroc)	1,00
Help Desk H/F	Non précisée	Meknès (Maroc)	0,96
Lead Développeur / TECH LEAD	Sol30	Casablanca (Maroc)	0,91
Senior Data Scientist	Leyton	Casablanca (Maroc)	1,00
Data Scientist	Leyton	Casablanca (Maroc)	1,00
Analyste sécurité des application et projets (...	AXA Services Maroc	Rabat (Maroc)	1,00
Change Manager (H/F)	BTechnologie	Rabat (Maroc)	0,94
Concepteur Développeur C++ (H/ F)	Non précisée	Casablanca (Maroc)	0,88
Ingénieur Back-End	Ago Jobs & HR	Rabat (Maroc)	0,75
Expert Technique JAVA/JEE	DXC - CDG	Rabat (Maroc)	0,90
Designer Fonctionnel Senior – Domaine Assu...	Non précisée	Casablanca (Maroc)	0,92
Lead Dev Laravel	Fairly	Casablanca (Maroc)	0,89
Webmaster	Fairly	Casablanca (Maroc)	0,99
Stagiaires ICT-PFE 2025	Akkodis Maroc	Casablanca (Maroc)	0,90
Data Scientist Angolphone (H/F)	DXC - CDG	Rabat (Maroc)	1,00
Data Scientist Angolphone (H/F)	DXC - CDG	Rabat (Maroc)	0,86
Concepteur Développeur .Net (F/H)	Sofrecom Maroc	Rabat (Maroc)	0,99
Développeurs UM6P Student Affairs	UM6P - Université Mohammed VI Polytechni...	Benguerir (Maroc)	0,85
INGENIEUR(E) PRODUCTION	Manpower Agences	Casablanca (Maroc)	0,61
Ingénieur Devops (F/H)	Sofrecom Maroc	Technopolis (Maroc)	0,72

In addition, the `dataset.arff` file used for training the Weka model can be presented to show how user and job offer skills are represented as binary vectors.



```
@attribute job_mysql numeric
@attribute job_mongodb numeric
@attribute job_oracle numeric
@attribute job_postgresql numeric
@attribute job_git numeric
@attribute job_docker numeric
@attribute job_linux numeric
@attribute job_aws numeric
@attribute job_azure numeric
@attribute job_laravel numeric
@attribute job_informatique numeric
@attribute job_developpement numeric
@attribute job_fullstack numeric
@attribute job_backend numeric
@attribute job_frontend numeric
@attribute job_management numeric
@attribute 'job_gestion de projet' numeric
@attribute job_organisation numeric
@attribute job_leadership numeric
@attribute job_communication numeric
@attribute job_analyse numeric
@attribute job_strategie numeric
@attribute label {0,1}

@data
1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,1,0,1,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,1,0,1,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,1,0,1,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0
1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,1,0,1,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,1,0,1,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,1,0,1,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,1,0,1,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,1,0,1,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,1,0,1,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,1,0,1,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,1,0,1,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,1,0,1,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Skill extraction from the CV

To enrich the user profile and improve recommendation accuracy, the system also exploits the candidate's CV, stored in the database as a BLOB. The `CVExtractor` class is used to automatically extract raw text from PDF-format CVs using the Apache PDFBox library.

```java
package com.example.jobs.maven_exemple;

import org.apache.pdfbox.pdmodel.PDDocument;

public class CVExtractor {
    public static String extractText(byte[] cvBytes) {
        if (cvBytes == null) return "";
        try (PDDocument doc = PDDocument.load(cvBytes)) {
            return new PDFTextStripper().getText(doc);
        } catch (Exception e) {
            return "";
        }
    }
}
```

The extracted text is then analyzed by the `JobMatcher.extractSkills` method, which identifies technical and general skills based on a predefined list of keywords (programming languages, technologies, tools, and professional skills). The detected skills are combined with those manually entered by the user to build a complete skill profile, which is used to compute compatibility with job offers.

During execution, the skills extracted from the CV are displayed in the console for verification and debugging purposes, as illustrated below:

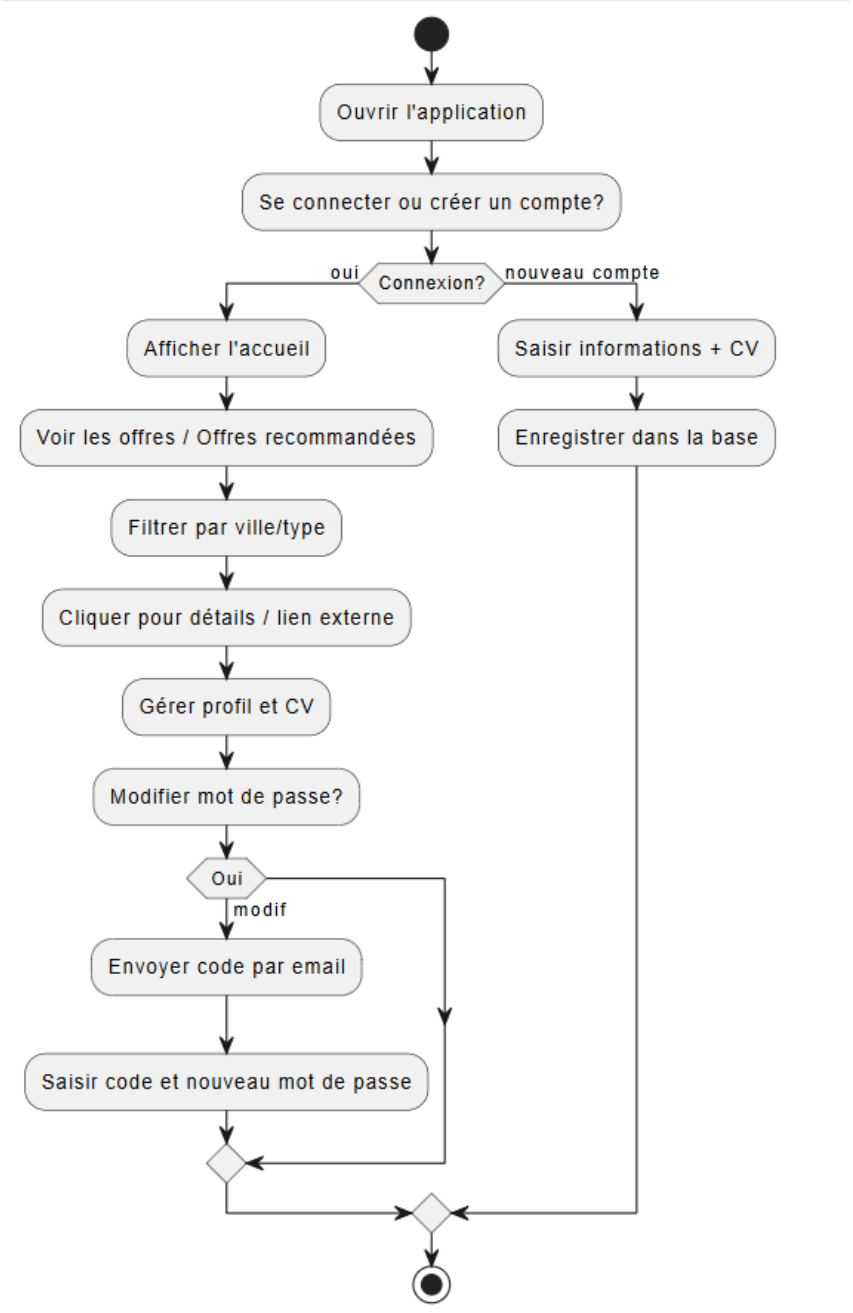
```
USER : khaliqisalma@gmail.com
Skills extraits du CV : c++ | python | css | laravel | c | oracle | vue | analyse | informatique | javascript | sql | docker | java | git | linux | php | html | mysql
```

3.3.12 Sorting by City / Specialty and Access to Details

- **Features:**
  - City-based filtering: input in `cityField` → dynamic table filtering
  - Filtering by type (internship/job): selection in `typeCombo`
  - Access to details: double-clicking a table row opens the offer link in the browser (`Desktop.getDesktop().browse(new URI(link))`)
- **Technical summary:**
  - The front-end is entirely developed using Java Swing

- Database communication is handled via the `Database` class
- CVs are stored as BLOBs; password management is functional, with an architecture allowing future integration of hashing mechanisms
- Recommended offers rely on a machine learning model to filter and display relevant opportunities
- User actions are secured and guided using popup messages (`JOptionPane`) for errors and confirmations

To visualize the flow of user interactions with the application, the following activity diagram illustrates the main steps from launching the application to browsing offers, managing the profile, and recovering a password.



3.4 Administrator Section

The administrator section allows full management of the application and enables actions reserved for users with an admin role.

- **Features:**
  - Launch scraping to update job offers
  - Database updates
  - Consultation of user and job offer statistics

3.4.1 Administrator Main Menu

The main menu allows the administrator to access essential functionalities:

- Access statistics
- Launch job scraping
- Logout

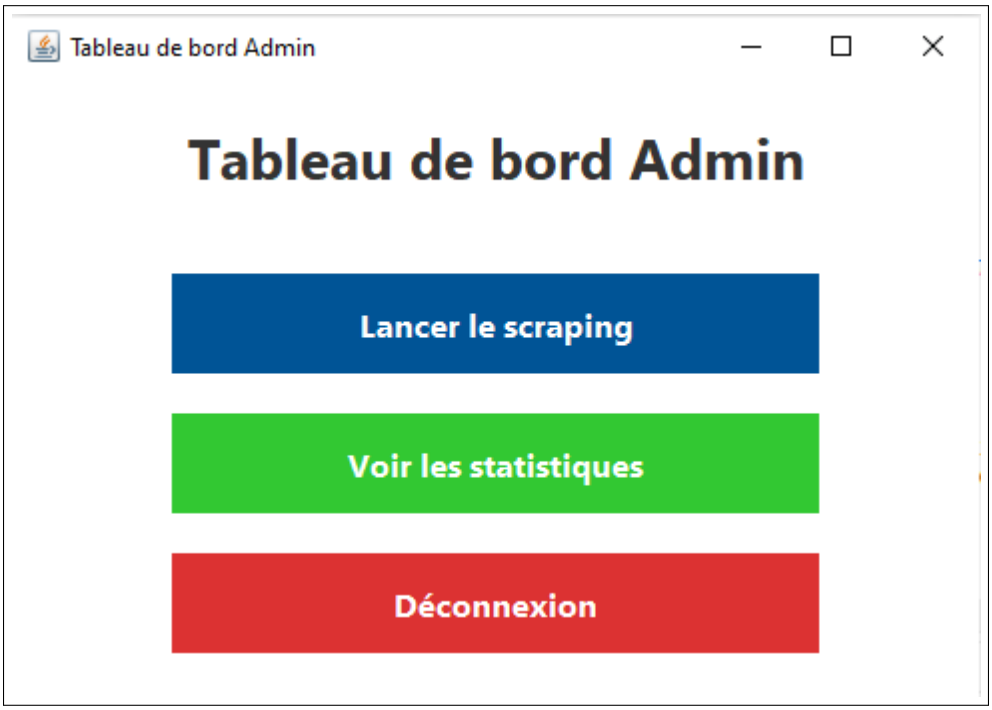


Figure 3.18: Administrator main menu

3.4.2 Statistics Menu

This menu centralizes access to available statistical visualizations and serves as the entry point to analytical charts.

Proposed features:

- Visualization of offers by platform
- Visualization of offers by city
- Analysis of the most active companies

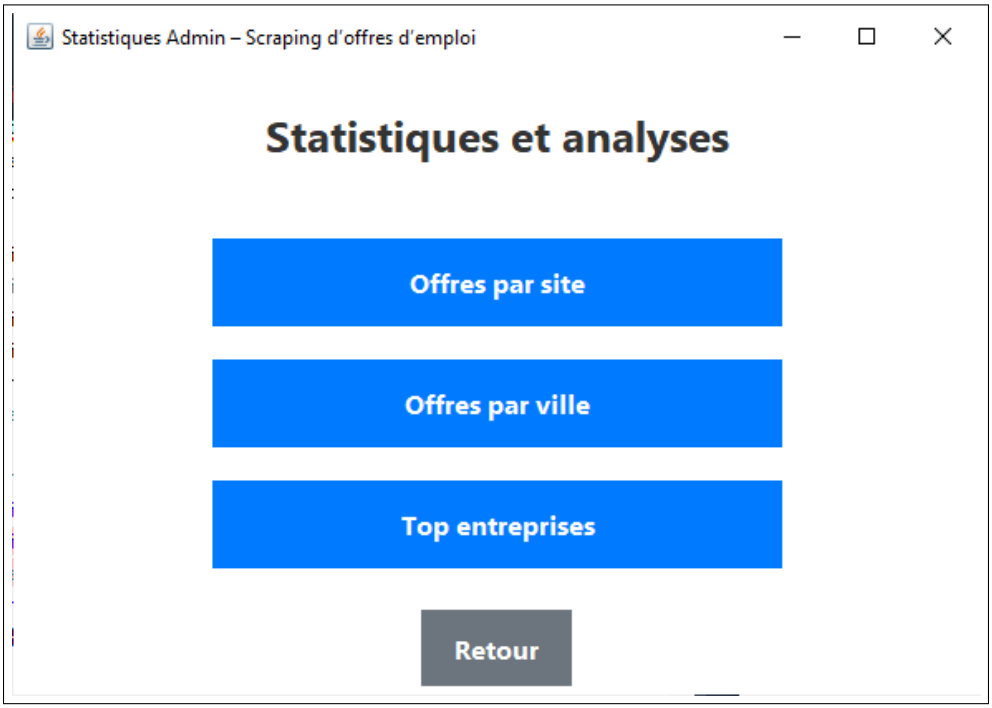


Figure 3.19: Statistics menu

3.4.3 Job Offers by Platform Visualization

This feature presents the distribution of job offers across different platforms (Jobzyn, Rekrute, Emploi.ma, MarocAnnonces).

A pie chart is used to provide a clear and immediate view of each platform's contribution.

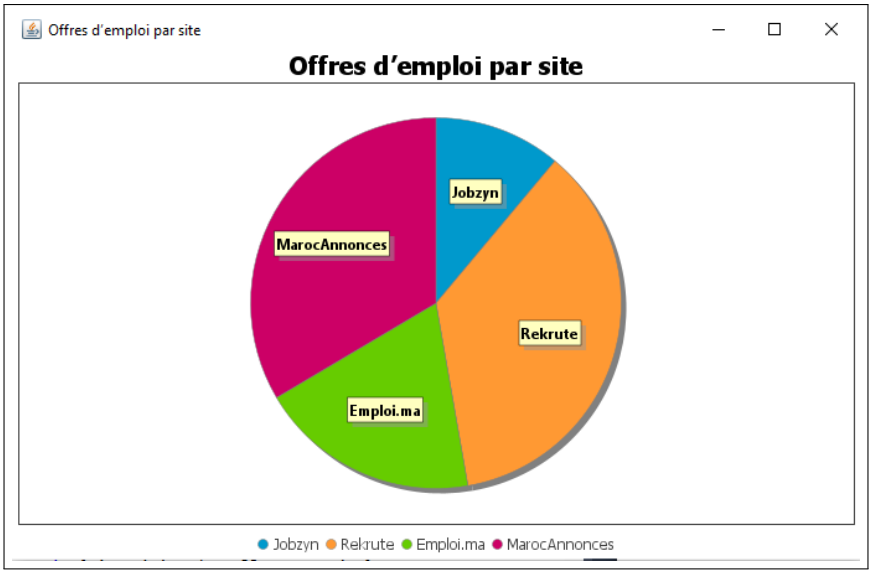


Figure 3.20: Job offers distribution by platform

3.4.4 Job Offers by City Visualization

This chart analyzes the geographical distribution of job offers based on cities.

A bar chart highlights cities with the highest concentration of job opportunities.

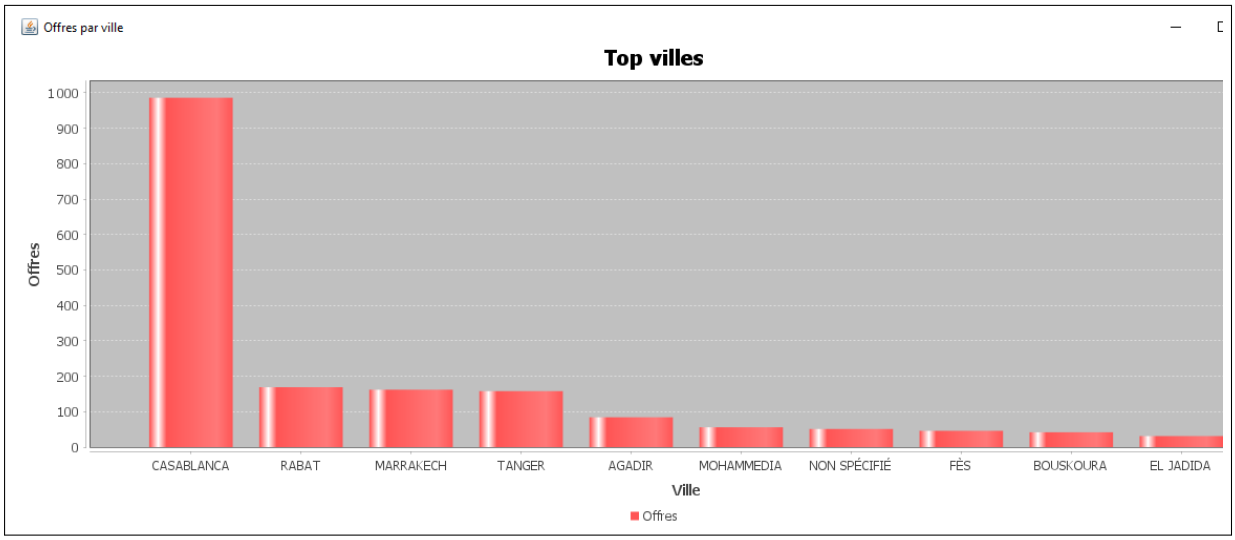


Figure 3.21: Job offers distribution by city

3.4.5 Analysis of the Most Active Companies

This feature displays companies offering the highest number of job opportunities. Irrelevant values such as *"Not specified"* or *"Anonymous"* are automatically excluded to ensure result reliability.

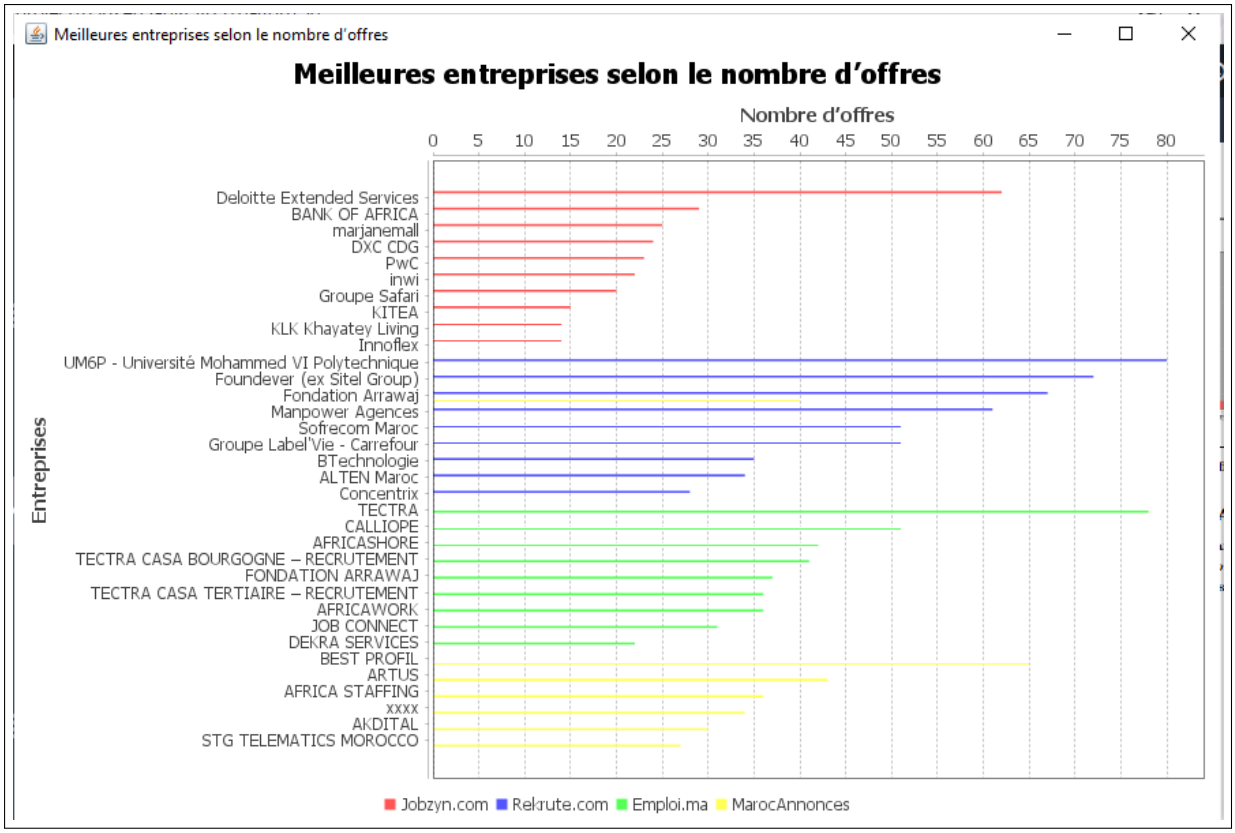


Figure 3.22: Top companies by number of offers

3.4.6 Scraping Execution

The administrator has a dedicated interface to manually launch scraping for different platforms.

This feature allows:

- Data updates
- Addition of new job offers
- Monitoring of scraper execution

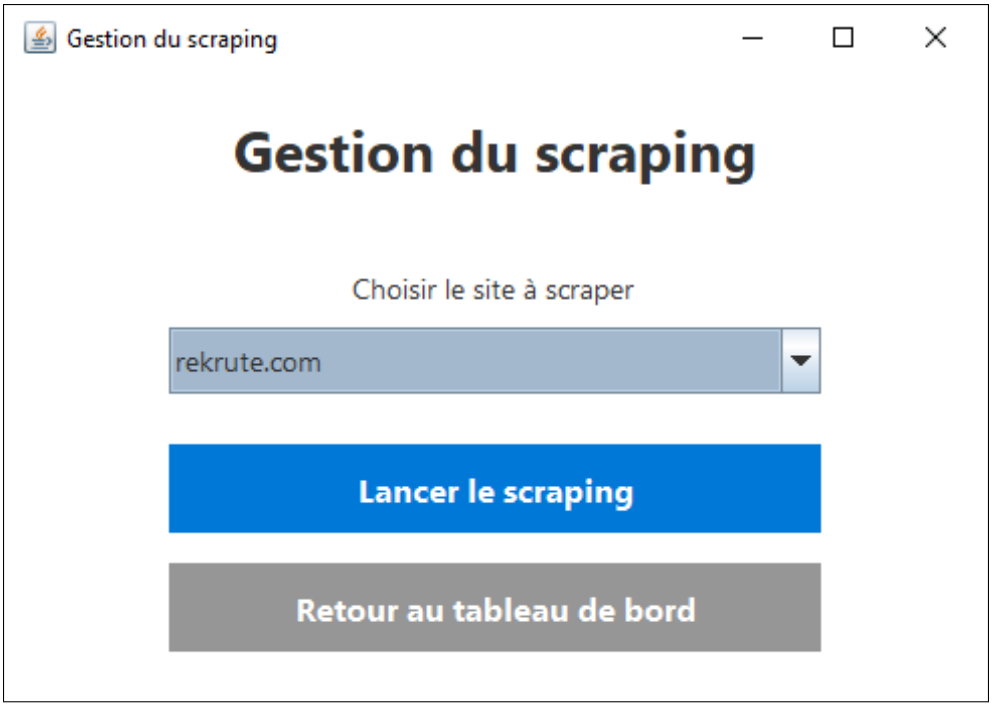


Figure 3.23: Scraping execution interface

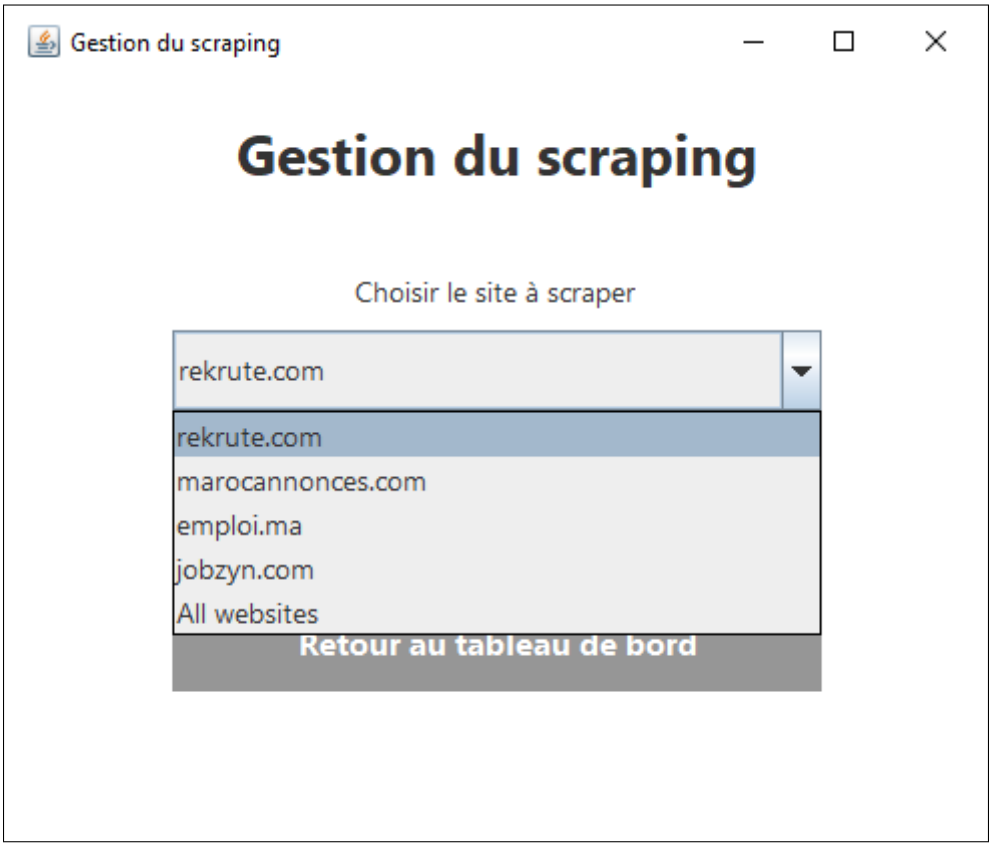


Figure 3.24: Scraping execution interface

3.4.7 Administrator Services

Business logic is managed by dedicated services:

- **StatisticsService**: statistics calculation and aggregation
- **ScrapingService**: orchestration of scraping across multiple platforms

This separation improves maintainability, readability, and scalability of the application.

```java
package service.admin;

import com.example.jobs.maven_exemple.Database;

public class StatisticsService {
    public static Map<String, Integer> topCompaniesOverall(int limit) {
        Map<String, Integer> result = new LinkedHashMap<>();

        try (Connection conn = Database.getConnection()) {

            // All tables
            String[][] tables = {
                    {"jobzyn_offres", "entreprise", null},
                    {"rekrute_offres", "entreprise", null},
                    {"job_offers", "company", "emploi.ma"},
                    {"job_offers", "company", "MarocAnnonces"}
            };

            for (String[] t : tables) {
                String table = t[0];
                String column = t[1];
                String source = t[2];

                String query;
                if (source == null) {
                    query = "SELECT " + column + " AS company, COUNT(*) AS count FROM " + table +
                            " WHERE " + column + " IS NOT NULL AND TRIM(" + column + ") != '' " +
                            "GROUP BY " + column;
                } else {
                    query = "SELECT " + column + " AS company, COUNT(*) AS count FROM " + table +
                            " WHERE source = ? AND " + column + " IS NOT NULL AND TRIM(" + column + ") != '' " +
                            "GROUP BY " + column;
                }
```

Figure 3.25: StatisticsService

```java
package service.admin;

import com.example.jobs.maven_exemple.Emploimascrapper;

public class AdminScrapingService {

    public static void startScraping(String site) {
        switch (site.toLowerCase()) {
            case "rekrute.com":
                scrapeRekrute();
                break;

            case "marocannonces.com":
                scrapeMarocAnnonces();
                break;

            case "emploi.ma":
                scrapeEmploi();
                break;

            case "jobzyn.com":
                scrapeJobzyn();
                break;

            case "all websites":
                int total = 0;
                total += scrapeRekrute();
                total += scrapeMarocAnnonces();
                total += scrapeEmploi();
                total += scrapeJobzyn();
                System.out.println("🎉 Total new offers added: " + total);
                break;

            default:
```

Figure 3.26: ScrapingService

General Conclusion

This project consists of an application for managing and consulting job and internship offers, developed in Java with a Swing graphical interface and a MySQL database. The application allows users to create an account, manage their profile and CV, browse all available offers, search and filter based on various criteria (city, contract type, keywords), and directly access job offer links.

It also integrates a recommendation system based on machine learning (Weka model), which compares user skills with those required by job offers and displays only those meeting a probability threshold. Security is ensured, particularly for password reset via SMTP email.

The interface is decorated and ergonomic thanks to Ftaltaf, with interactive buttons, colors, and tables enhancing user experience. The database centralizes all information about users and job offers, enabling reliable insertion, update, and consultation operations.

Overall, the project provides a complete solution for job opportunity search and recommendation, combining data management, machine learning, security, and a user-friendly interface.