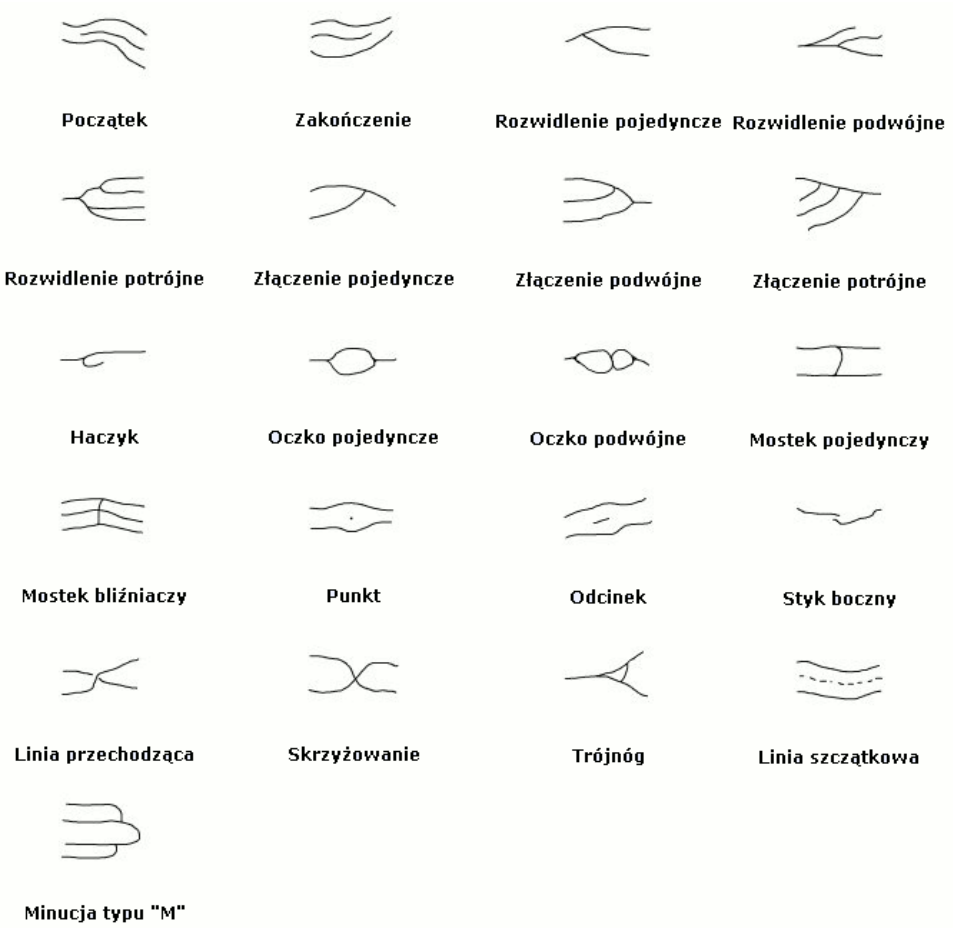
WSTĘP

Wyjaśnienie pojęcia

Biometria to dziedzina nauki zajmująca się identyfikacją i weryfikacją tożsamości jednostek na podstawie ich unikalnych cech biologicznych lub behawioralnych. Cechy biometryczne mogą obejmować takie elementy jak rozmiar dłoni, odciski palców, siatkówkę oka, wielkość ust czy całościową strukturę twarzy. Cechy behawioralne obejmują między innymi sposób chodzenia, charakterystyczny sposób pisania czy mówienia. Biometria znajduje zastosowanie w wielu dziedzinach życia, w tym w systemach bezpieczeństwa, monitorowaniu i identyfikacji osób, uwierzytelniania w technologiach mobilnych, bankowości czy kryminalistyce. Jej główną zaletą jest wyjątkowy poziom niezawodności, który opiera się na unikalnych cechach biometrycznych jednostek. Jednakże, ze względu na wrażliwość i prywatność danych osobowych, konieczne jest staranne i bezpieczne przechowywanie tych informacji.

Najpopularniejszą cechą biometryczną każdego człowieka są linie papilarne. Te charakterystyczne wzory skórne, występują głównie na opuszkach palców, choć mogą być również obecne na dłoniach i stopach. Formują się w okresie życia płodowego i uważane są za cechy odziedziczone genetycznie. Linie papilarne składają się z rowków, bruzd i guzków na powierzchni skóry, tworzących różnorodne kształty, takie jak pętle, rozwidlenia, oczka czy odcinki. Te charakterystyczne wzory nazywane są minucjami. Klasyfikacja typów minucji według C. Grzeszyka, wyróżnia ich około 23. Wzajemny układ minucji jednoznacznie identyfikuje daną osobę. Do uznania dwóch śladów linii papilarnych za tożsame wystarczy od kilku do kilkunastu cech wspólnych (ta sama minucja występująca w tym samym miejscu). W Polsce przyjmuje się, że wystarczające jest 12 zgodnych minucji do pewnego określenia tożsamości, choć niektóre minucje występujące rzadziej zmniejszają tę liczbę. Do pozyskania odcisków palca wykorzystuje się czytniki linii papilarnych, które za pomocą różnych technik, pozyskują zdjęcie odcisku i zapisują dane w postaci elektronicznej.



Rysunek 1. Typy minucji

Źródło: https://kcir.pwr.edu.pl/~witold/aiarr/2008_projekty/odciski/

Rysunek 2. Przykładowe minucje na odcisku

Rodzaje urządzeń

Wyróżnia się cztery rodzaje czytników linii papilarnych:

Optyczny czytnik linii papilarnych

Czytniki te działają na zasadzie tworzenia kserokopii odcisku poprzez matrycę CCD, zwykle wykorzystywaną w aparatach cyfrowych i kamerach. Elementy światłoczułe w tej matrycy generują sygnał elektryczny proporcjonalny do ilości światła padającego na nie, co pozwala na utworzenie obrazu odcisku z jasnych i ciemnych pikseli.

Pojemnościowy czytnik linii papilarnych

Ten rodzaj czytnika, do stworzenia obrazu odcisku palca wykorzystuje niewielkie kondensatory. Gdy położymy palec na czytniku, urządzenie mierzy ładunek – grzbiety wykazują zmianę pojemności, podczas gdy doliny nie powodują praktycznie żadnej zmiany. Są one uważane za bardziej precyzyjne i mniej podatne na fałszerstwo.

Ultradźwiękowy czytnik linii papilarnych

Działanie tego skanera polega na budowaniu obrazu odcisku palca dzięki dyfrakcji, rozproszeniu oraz odbiciu dźwięku podczas kontaktu powierzchni opuszka palca z powierzchnią czytnika. Grzbiety i doliny linii papilarnych odbijają dźwięk inaczej, więc czytniki mogą stworzyć szczegółową mapę 3D wzorów minucji.

Termiczny czytnik linii papilarnych

Ten typ skanera wykorzystuje sensory termiczne, które wychwytują minimalne różnice temperatur pomiędzy liniami palca. Grzbiety posiadają wyższą temperaturę niż doliny i na tej podstawie generuje jego obraz termiczny.

W niniejszym projekcie, do pobierania odcisków palca, będzie wykorzystywany skaner optyczny FS80H firmy Futronic.

Rysunek 3. Skaner optyczny Futronic FS80H

Źródło: https://www.futronic-tech.com/pro-detail.php?pro_id=1543

Futronic model FS80 jest elektronicznym skanerem optycznym którego czas potrzebny na przeprowadzenie skanu to około 100 milisekund. Skan odbywa się poprzez przyłożenie palca do szkła ochronnego znajdującego się na górnej części obudowy. Ponadto skaner ten jest wyposażony w technologię umożliwiającą rozróżnianie czy skanowany obiekt jest rzeczywistym ludzkim palcem co znacząco utrudnia próbę fałszerstwa. Dodatkowo urządzenie jest w stanie pracować w ciemności dzięki zestawowi diod LED których zadaniem jest podświetlanie przyłożonego opuszka palca.

Parametry skanera :

- 45 mm szerokości, 60 mm długości oraz 25 mm wysokości,
- pobierany skan opuszka palca w rozdzielczości 480 x 320 pixeli,
- obraz pozyskiwany w 8 bitach (256 odcieniach szarości),
- interfejs USB w technologii 2.0.

Silnik modułu do skanowania odcisków palców jest w stanie dopasować zrolowane czy spłaszczone odciski palców z wysokim stopniem dokładności. Jest on zaprojektowany tak aby był odporny na przesunięcia palca na skanerze, obrót czy nawet deformację odcisku.

Sposoby analizy danych

Rozróżniamy dwa główne cele analizy danych: weryfikację i identyfikację.

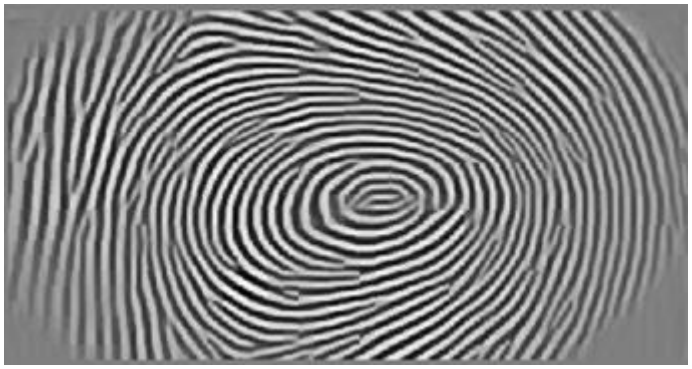
Proces weryfikacji polega na potwierdzeniu tożsamości osoby na podstawie odcisku palca poprzez porównanie go z odciskiem palca przechowywanym w bazie danych dla tej osoby.

Proces identyfikacji polega na porównaniu odcisku palca z zapisanymi w bazie danych odciskami w celu ustalenia tożsamości osoby. System przeszukuje całą bazę danych w celu znalezienia dopasowania dla danego odcisku palca.

Pierwszym krokiem jest pobranie obrazu odcisku palca za pomocą skanera, co pozwala na uzyskanie cyfrowej wersji odcisku. Po zeskanowaniu palca należy wyekstrahować linie papilarne.

Rysunek 4. Pierwotny obraz odcisku palca.

W przypadku występowania braków w odcisku palca, można zastosować różne techniki, takie jak transformatę Fouriera, która może pomóc w uzupełnieniu brakujących informacji poprzez analizę częstotliwościową obrazu.

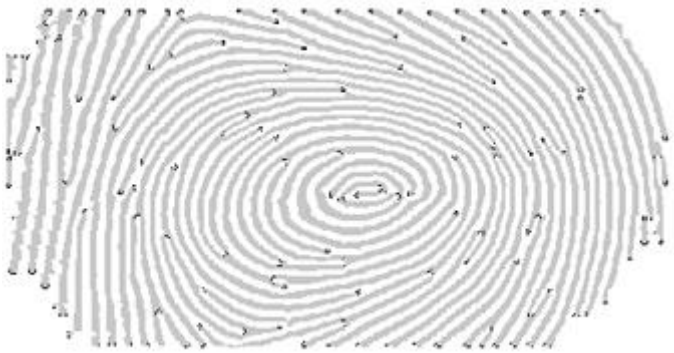


Rysunek 5. Wzbogacony obraz

Następnie obraz taki poddaje się progowaniu, w którym przekształca się obraz wejściowy do formy binarnej, gdzie piksele są przyporządkowane do jednej z dwóch klas (czarny lub biały, 1 lub 0), na podstawie pewnego ustalonego progu. Zastosowanie np. algorytmu krawędziowego pozwala na wyłuskanie linii papilarnych.

Rysunek 6. Sprogowany obraz

Dopiero na takim obrazie wykorzystuje się algorytmy wyróżniania minucji. Jeżeli występują w tych samych miejscach na obu odciskach, to przyjmuje się, że znajdując 12 takich cech, można stwierdzić z wysokim prawdopodobieństwem, że są to odciski należące do tej samej osoby.



Rysunek 7. Wyróżnione minucje

Źródło: https://kcir.pwr.edu.pl/~witold/aiarr/2007_projekty/odciski2/

Znaczenie metody oraz jej zastosowanie

Metoda identyfikacji biometrycznej poprzez odcisk palca jest powszechnie stosowana w wielu obszarach, szczególnie tam, gdzie istnieje potrzeba zabezpieczenia dostępu przed osobami trzecimi. Najczęściej spotyka się ją przy kontrolowanych wejściach do budynków o wysokim stopniu zabezpieczeń, umożliwiając tym samym monitorowanie obecności osób w konkretnych jego częściach. Równie popularne jest jej wykorzystanie do autoryzacji transakcji finansowych przez użytkownika. W obecnych czasach, gdzie większość osób posiada smartfony z czytnikami linii papilarnych, umożliwia to wykonywanie operacji w aplikacjach bankowych oraz bezpieczne logowanie się do nich.

Wyjaśnienie celu projektu

Celem projektu jest opracowanie systemu kontroli dostępu do pomieszczeń na terenie firmy. Podczas zatrudnienia, do systemu wprowadzane są skany czterech pierwszych palców ręki nowego pracownika, licząc od kciuka. Pracownik zostaje następnie przypisany do określonych pomieszczeń na terenie firmy. Od tego momentu ma do nich dostęp. Gdy pracownik chce wejść do danego pomieszczenia, musi losowo zeskanować jeden z czterech palców (palec jest losowany przez system). Następnie skan ten jest porównywany ze wszystkimi skanami istniejącymi w bazie dla danego typu palca, na przykład kciuka. Po czym następuje proces identyfikacji i znajdowany jest pracownik z najwyższym wynikiem zgodności dla danego odcisku, następnie weryfikowany jest jego dostęp do danego pomieszczenia. Jeśli pracownik ma dostęp do danego pokoju drzwi zostają otwarte, w przeciwnym wypadku pozostają zamknięte. Pozwoli to na kontrolowanie obecności pracowników, na terenie firmy ale również może przyczynić się do poprawy bezpieczeństwa poprzez zapewnienie, że tylko uprawnieni pracownicy mają dostęp do określonych pomieszczeń. Posiadanie takich danych może być przydatne w przypadku audytów bezpieczeństwa lub dochodzeń wewnętrznych. Takie rozwiązanie pozwala również na szybkie reagowanie w sytuacjach awaryjnych, gdy konieczne jest natychmiastowe ograniczenie dostępu do pewnych obszarów firmy.

CZĘŚĆ DOŚWIADCZALNA

Opis:

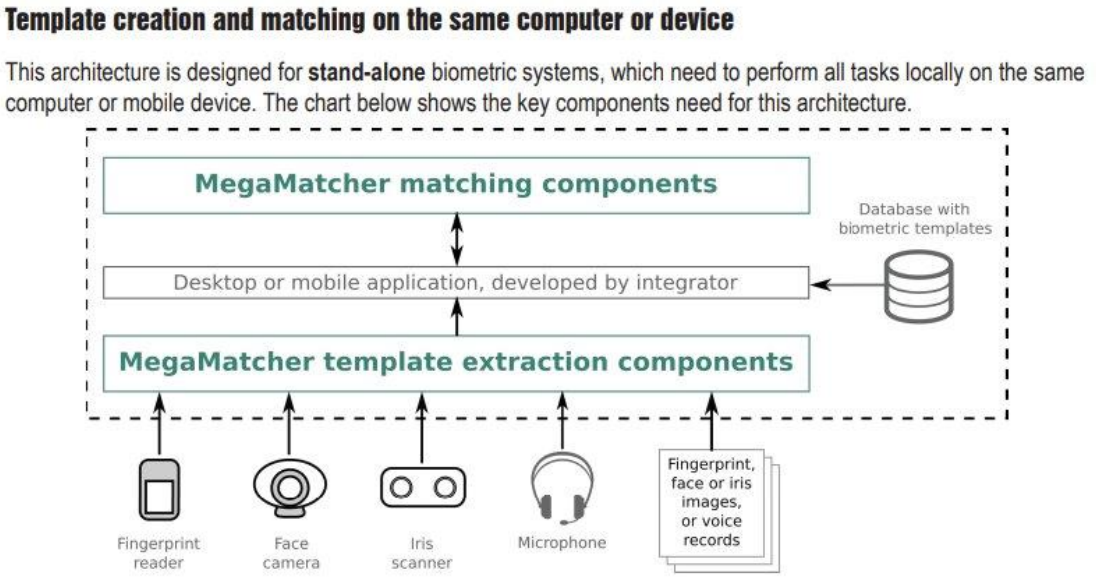
W doświadczeniu zasymulujemy system zabezpieczający dostęp do pomieszczeń w budynku oparty o biometrię palca. Aby tego dokonać będą nam potrzebne skanery odcisków palca, które umożliwią pobranie danych biometrycznych użytkowników oraz aplikacja połączona z bazą danych, która będzie przetwarzała odciski palców.

Doświadczenie będzie składało się z następujących części:

1. Stworzenie aplikacji do pobierania i analizowania danych biometrycznych użytkowników wraz z bazą danych, w której dane te będą przechowywane
2. Zebranie danych z uczelnianych skanerów
3. Testy aplikacji

SDK

System kontroli dostępu opiera się o zaawansowane SDK od Neurotechnology, znane jako VeriFinger. Jest ono kluczowym narzędziem zapewniającym skuteczną identyfikację i weryfikację odcisków palców. VeriFinger został specjalnie dostosowany i zintegrowany z aplikacją, aby spełnić wymagania systemu kontroli dostępu



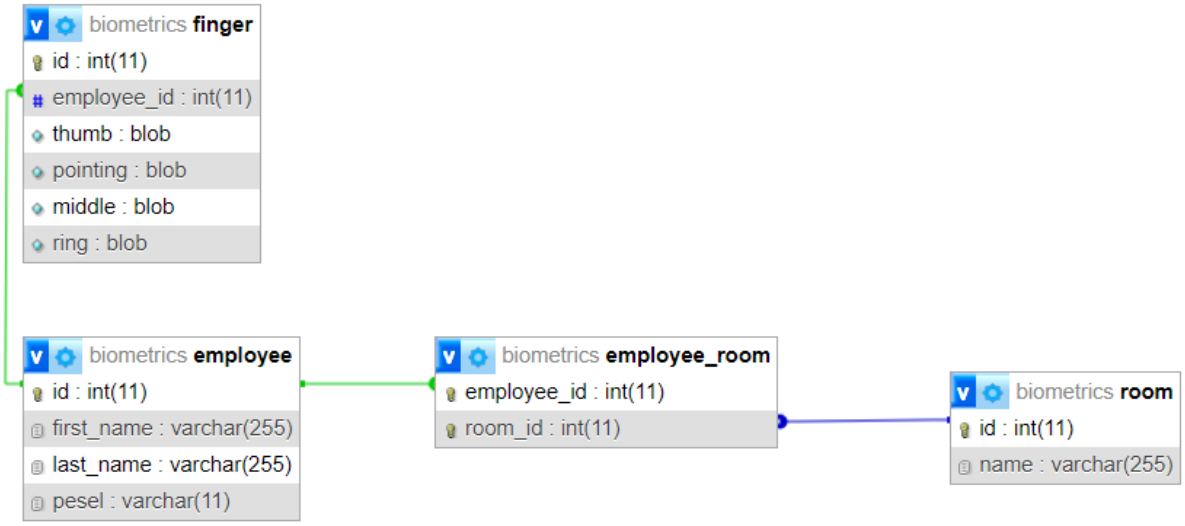
Rysunek 8. Diagram architektury

Przy projektowaniu architektury systemu sugerowaliśmy się powyższym diagramem z dokumentacji MegaMatcher SDK.

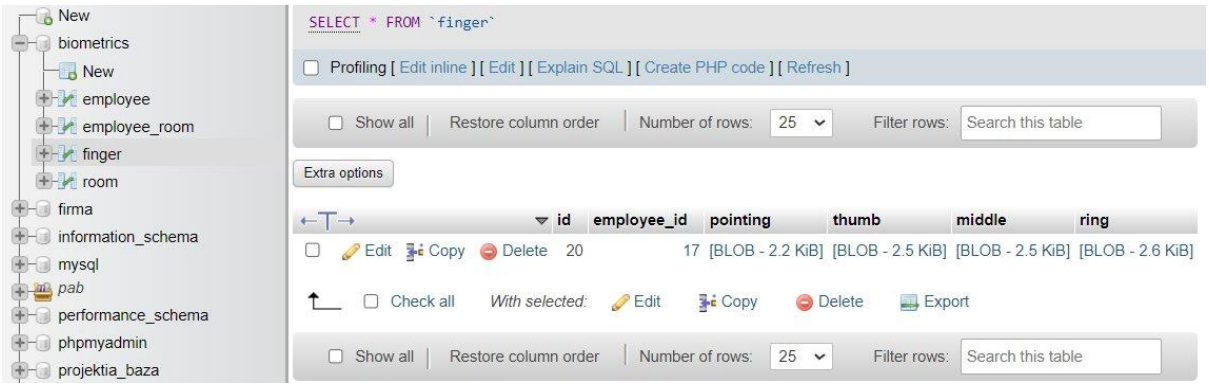
Zaprojektowanie i stworzenie Bazy Danych

W projekcie korzystamy z bazy danych MySQL, postawionej lokalnie za pomocą XAMPP. W skład bazy wchodzą 4 tabele:

- **Employee** – przechowuje dane pracownika (id, imię, nazwisko, pesel)
- **Finger** – przechowuje dane biometryczne pracowników (template'y z czterech pierwszych palców, licząc od kciuka)
- **Room** – przechowuje dane na temat pokoi (id, nazwa)
- **Employee_Room** – tabela pomocnicza, która łączy tabele Employee oraz Room w relacji wiele-do-wiele



Rysunek 9. Diagram ERD



Rysunek 10. Tabela finger z przykładowymi danymi

```sql
-- Usuwanie bazy danych jesli istnieje
DROP DATABASE IF EXISTS biometrics;

-- Tworzenie bazy danych, jeśli nie istnieje
CREATE DATABASE IF NOT EXISTS biometrics;
USE biometrics;

-- Tworzenie tabeli Room, jeśli nie istnieje
CREATE TABLE IF NOT EXISTS Room (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255)
);

-- Tworzenie tabeli Employee, jeśli nie istnieje
CREATE TABLE IF NOT EXISTS Employee (
    id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    pesel VARCHAR(11)
);

-- Tworzenie tabeli Employee_Room (table łącząca m-t-m)
CREATE TABLE IF NOT EXISTS Employee_Room (
    employee_id INT,
    room_id INT,
    PRIMARY KEY (employee_id, room_id),
    FOREIGN KEY (employee_id) REFERENCES Employee(id),
    FOREIGN KEY (room_id) REFERENCES Room(id)
);

-- Tworzenie tabeli Finger, jeśli nie istnieje
CREATE TABLE IF NOT EXISTS Finger (
    id INT PRIMARY KEY AUTO_INCREMENT,
    employee_id INT,
    thumb BLOB NULL,
    pointing BLOB NULL,
    middle BLOB NULL,
    ring BLOB NULL,
    FOREIGN KEY (employee_id) REFERENCES Employee(id)
);
```

Rysunek 11. Skrypt tworzący bazę danych

Stworzenie aplikacji i połączenie jej z bazą danych

System, który staramy się stworzyć jest na tyle uniwersalny, że może znaleźć zastosowanie w każdym miejscu, gdzie istnieje potrzeba zarządzania dostępem do pomieszczeń przez odpowiednio upoważnione osoby. Aplikacja składa się z dwóch okien: „Add Employee" i„Enter to Room".

Okno „Add Employee" jest przeznaczone dla osoby zarządzającej dostępem do pomieszczeń w budynku, tzn. w tym miejscu pracodawca może przydzielić dostęp dla innych pracowników do wybranych pomieszczeń. Wystarczy podać dane pracownika, któremu chcemy przydzielić dostęp i wybrać do jakich pomieszczeń będzie mógł wchodzić. Po wpisaniu danych i wybraniu pomieszczeń, klikamy przycisk „scan" aby pracownik mógł zeskanować swoje palce. Skanuje on 4 palce (kciuk, palec wskazujący, palec środkowy oraz palec serdeczny), po każdym skanie(jeśli był odpowiednio dobry) i kliknięciu przycisku „Add to DB", do bazy danych zostanie wysłany template wygenerowany w aplikacji. Z każdym kolejnym palcem, osoba pobierająca skany palca powinna zmienić na liście rozwijanej skanowany palec i kliknąć „scan".

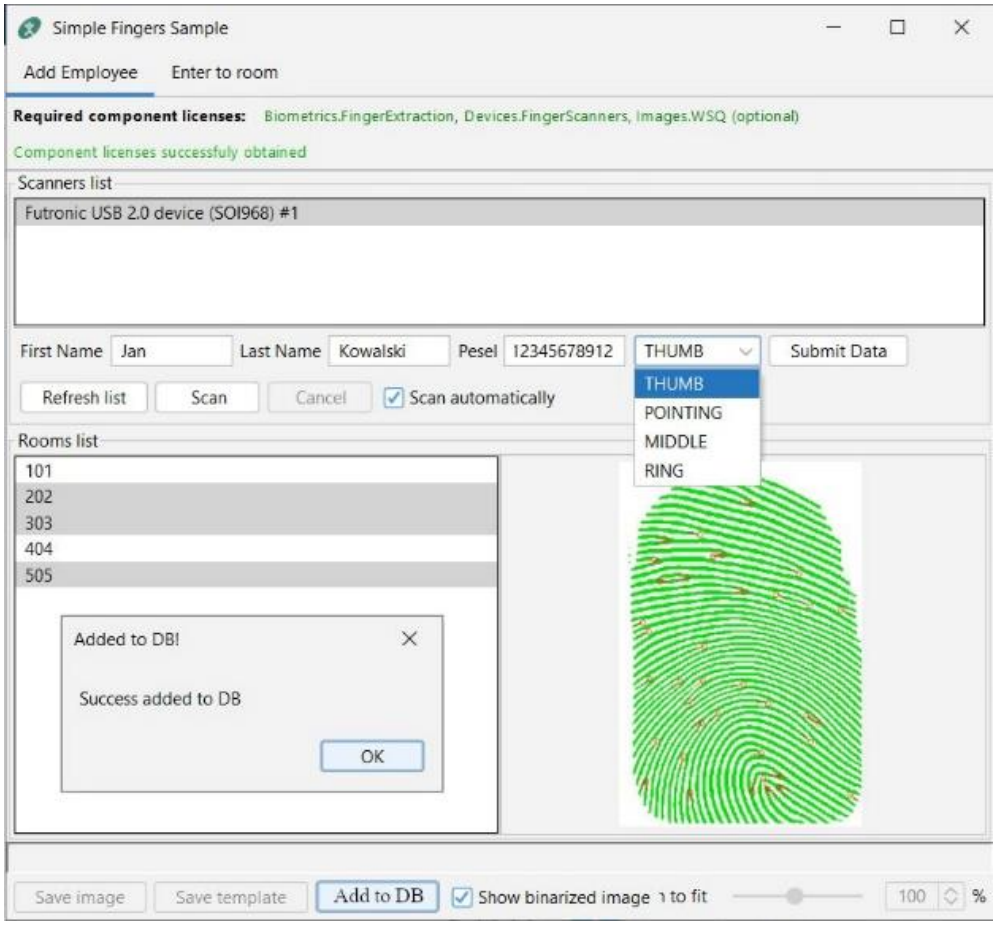
```
private void processDB() {
    // Sprawdzmy czy pracownik o podanym peseslu istnieje w bazie
    int employeeId = getEmployeeByPesel();

    if (employeeId == -1) { // Jeśli nie istnieje
        // Dodanie nowego pracownik do bazy
        addEmployeeToDB();

        // Pobranie ID pracownika
        employeeId = getEmployeeByPesel();
    }
    // Sprawdzamy czy isnieją dane biometryczne pracownika o podanym ID w bazie danych
    int fingerId = getFingerByEmployeeId(employeeId);

    if (fingerId == -1) {// Jeśli nie istnieją
        // Dodajemy dane biometryczne
        addEmployeeFingers(employeeId);

        // Przypisujemy pracownikowi wybrane pokoje
        addEmployeeToRoom(employeeId);
    } else { // Jeśli istnieją
        // Przypisujemy pracownikowi wybrane pokoje
        addEmployeeToRoom(employeeId);

        // Dodajemy/aktualizujemy dane biometryczne
        updateEmployeeFinger(employeeId);
    }
    // Po wszyskim wyświetlamy komunikat
    SwingUtilities.invokeLater(() ->
            JOptionPane.showMessageDialog( parentComponent: this, message: "Success added to DB", title: "Added to DB!",
                    JOptionPane.PLAIN_MESSAGE));
}
```

Rysunek 12. Dodanie odcisku do bazy danych

```java
private void addEmployeeFingers(int id) {
    // Sprawdzamy, który palec jest teraz skanowany
    String finger = ((String) Objects.requireNonNull(comboFingers.getSelectedItem())).toLowerCase();

    // Tworzymy QUERY dla języka MYSQL
    final String query = String.format("INSERT INTO finger(%s, employee_id) VALUES (?, ?)", finger);

    // Łączymy się z bazą danych
    try (Connection connection = DriverManager.getConnection(DB_URL, user: "root", password: "")) {
        try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {
            preparedStatement.setBytes( i: 1, subject.getTemplateBuffer().toByteArray());
            preparedStatement.setInt( i: 2, id);

            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
}
```

Rysunek 13. Zapisywanie danych biometrycznych pracownika do bazy

Powyższy kod przedstawia operacje dodania tokenu do bazy danych, token jest to wektor cech konkretnego odcisku palca. Jest on trzymany w bazie danych jako tablica bajtów.



Rysunek 14. Dodawanie nowego pracownika

W czasie tworzenia aplikacji gdy nie mieliśmy skanerów to korzystaliśmy z obrazów .jpg wcześniej pobranych odcisków podczas zajęć, poniżej znajduje się kod (Rysunek 15), który tworzył template dla odcisku palca z obrazka.

```
private byte[] getFingerTemplate() {
    // Pobieramy aktualnie zaznaczony palec z listy rozwijanej
    String fn = ((String) Objects.requireNonNull(comboFingers.getSelectedItem())).toLowerCase();
    String imgPath = null;

    // Wybieramy zdjęcie odcisku w zależności od wybranego palca (fn)
    switch (fn) {
        case "thumb": imgPath = "C:\\Users\\Filip\\Desktop\\odciski\\PL_THUMB.jpg";
            break;
        case "pointing": imgPath = "C:\\Users\\Filip\\Desktop\\odciski\\PL_POINTING.jpg";
            break;
        case "middle": imgPath = "C:\\Users\\Filip\\Desktop\\odciski\\PL_MIDDLE.jpg";
            break;
        case "ring": imgPath = "C:\\Users\\Filip\\Desktop\\odciski\\PL_RING.jpg";
            break;
    }

    // Tworzymy obiekty NFinger i NSubject
    NFinger finger = new NFinger();
    NSubject subject = new NSubject();

    // Ustawiamy dla obiektu finger ścieżke do zdjęcia odisku palca
    finger.setFileName(Objects.requireNonNull(imgPath));

    // Dodajemy finger do subject
    subject.getFingers().add(finger);

    // Ustawiamy template size na large
    FingersTools.getInstance().getClient().setFingersTemplateSize(NTemplateSize.LARGE);

    // Tworzymy template dla danego subjecta
    NBiometricStatus status = FingersTools.getInstance().getClient().createTemplate(subject);

    // Jeśli wszystko poszło dobrze to zwracamy template jako tablice bajtów
    if (status == NBiometricStatus.OK) {
        return subject.getTemplateBuffer().toByteArray();
    }
    return new byte[]{};
}
```

Rysunek 15. Kod z testów

Okno „Enter to room” jest przeznaczone dla osoby próbującej wejść do danego pomieszczenia, to tutaj jest symulowane wejście do pokoju. Aby sprawdzić czy pracownik ma dostęp do pokoju musi on wybrać z listy numer pomieszczenia, do którego chce wejść(w rzeczywistości każde pomieszczenie powinno mieć przypisany osobny skaner), następnie kliknąć przycisk „scan”. Po zeskanowaniu palca system stwierdzi czy pracownik ma dostęp do pomieszczenia

czy nie. Aby system mógł stwierdzić czy dany odcisk palca znajduje się w bazie, najpierw musi pobrać wszystkie tokeny dla danego palca (thumb, pointing itd.) i zapisać je w liście, wygląda to następująco:

```
private void loadTemplates() {
    // Czyścimy listę subjectów
    subjects.clear();
    // Wyciągamy z bazy danych ID użytkownika razem z template'm palca
    Map<Integer, byte[]> fingers = getSpecificFingers();
    // Czyścimy wszystkie poprzednie dane aby móc załadować nowe do klienta
    FingersTools.getInstance().getClient().clear();
    // Tworzymy nowe zadanie do zapisania (enroll) danych
    NBiometricTask enrollmentTask = new NBiometricTask(EnumSet.of(NBiometricOperation.ENROLL));
    // Dla każdego odcisku palca z bazy
    for (Map.Entry<Integer, byte[]> entry : fingers.entrySet()) {
        // bierzemy id pracownika razem z palcem
        Integer employeeId = entry.getKey();
        byte[] fingersData = entry.getValue();

        // Tworzymy buffer i "wypełniamy" go template'm
        NBuffer buffer = new NBuffer(fingersData);
        // Tworzymy subject z dancyh buffera
        NSubject subject = NSubject.fromMemory(buffer);
        // Ustawiamy id subjecta na id employee
        subject.setId(String.valueOf(employeeId));
        // dodajemy do listy subjectów subject
        subjects.add(subject);
        // dodajemy do zadania subject, aby mógł go wykorzystać przy późniejszej identyfikacji
        enrollmentTask.getSubjects().add(subject);
    }
    // Zlecamy wykonanie powyższego zadania biometrycznemu silnikowi,
    // wyniki zadania będą przchwytywane przez enrollHandler
    FingersTools.getInstance().getClient().performTask(enrollmentTask,  attachment: null, enrollHandler);
}
```

Rysunek 16. Wczytywanie tokenów z bazy danych

Po zeskanowaniu, oraz wczytaniu tokenów z bazy, następuje proces identyfikacji aktualnie zeskanowanego odcisku palca, z odciskami wczytanymi z bazy. Gdy identyfikacja zakończy się pozytywnie, IdentyficationHandler wywoła metodę completed:

```java
@Override
public void completed(final NBiometricStatus status, final Object attachment) {
    SwingUtilities.invokeLater(() -> {
        if ((status == NBiometricStatus.OK) || (status == NBiometricStatus.MATCH_NOT_FOUND)) {
            // Ustawiamy maxScore na najniższą możliwą liczbę
            int maxScore = Integer.MIN_VALUE;
            System.out.println("STATUS = " + status);
            // Match subjects.
            for (NSubject s : getSubjects()) {
                boolean match = false;
                for (NMatchingResult result : getSubject().getMatchingResults()) { // dla każdego wyniku identyfikacji
                    if (s.getId().equals(result.getId())) {
                        match = true;
                        // Sprawdzamy czy maxScore jest mniejsze niż score dla danego wyniku identyfikacji
                        if (maxScore < result.getScore()) {
                            // przypisujemy do maxScore wynik rezultatu
                            maxScore = result.getScore();
                            // przypisujemy do employeeId, id subjectu (czyli id pracownika)
                            employeeId = Integer.parseInt(s.getId());
                        }
                        System.out.println("maxScore = " + maxScore + " emplyee = " + employeeId);
                        break;
                    }
                }
                if (!match) {
                    System.out.println("NOT MACHES");
                }
            }
            // po wszystkim sprawdzamy czy użytkownik, który skanował palec może wejść do wybranego pokoju
            checkIfUserCanEnter(maxScore, employeeId);
        } else {
            SwingUtilities.invokeLater(() -> {
                JOptionPane.showMessageDialog( parentComponent: EnterToRoom.this,
                        message: "Identification failed: " + status,
                        title: "Error", JOptionPane.WARNING_MESSAGE);
            });
        }
    });
}
```

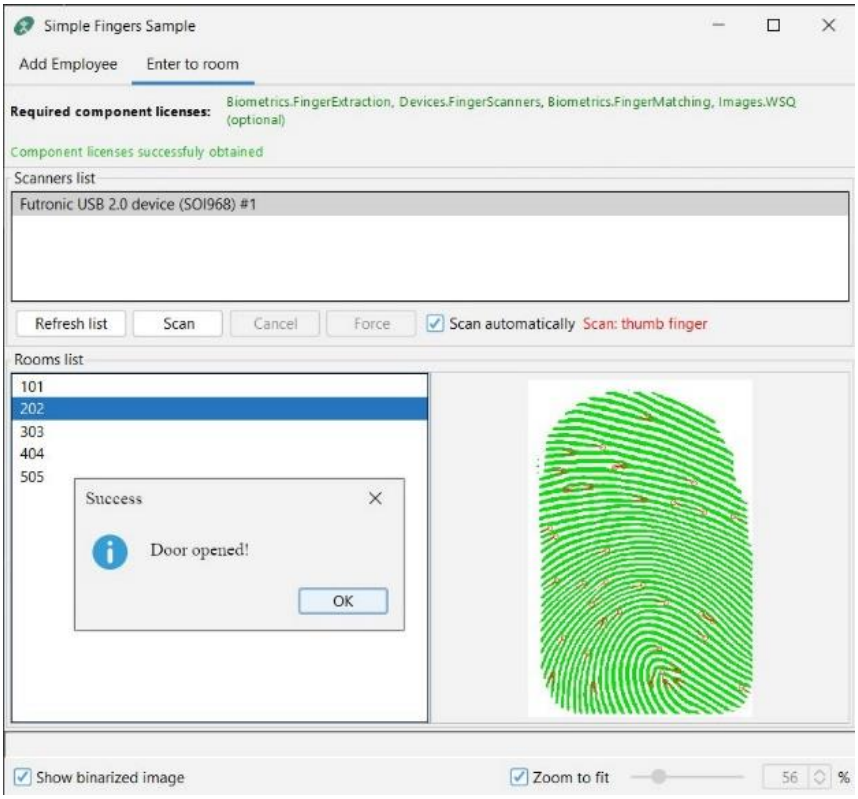
Rysunek 17. Identyfikacja zakończyła się pomyślnie

Po sprawdzeniu wszystkich wyników w powyższym kodzie, wywoływana jest metoda **checkIfUserCanEnter()**, która sprawdza czy pracownik ma dostęp do danego pokoju.

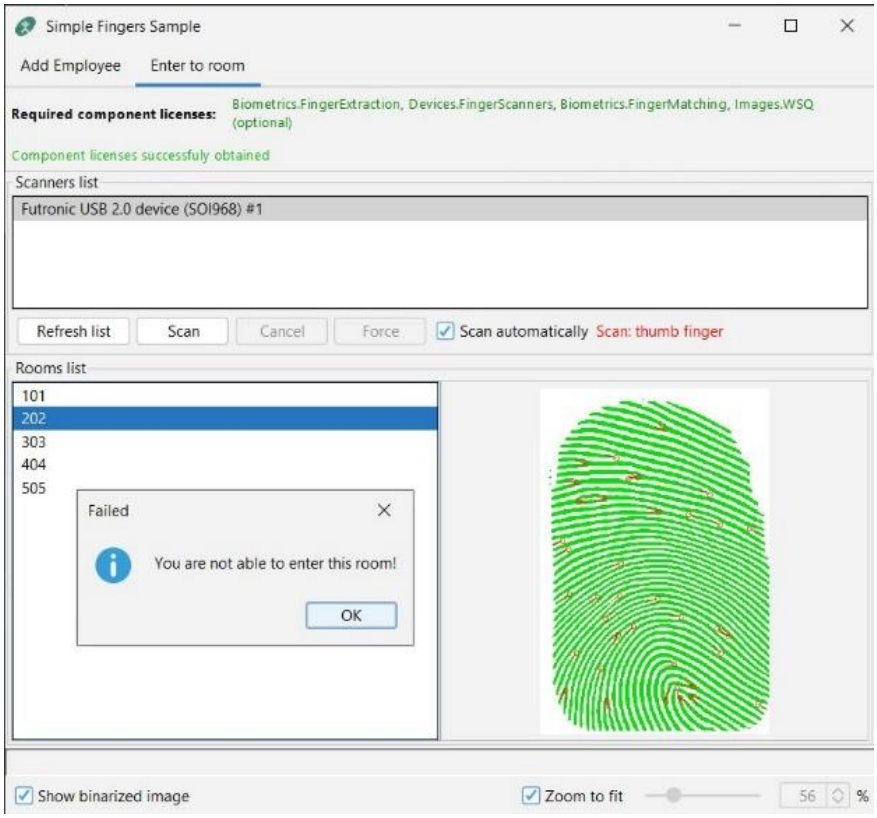
```
private void checkIfUserCanEnter(int maxScore, int employeeId) {
    // sprawdzamy czy użytkownik znajduje się w bazie
    if (employeeId != -1) {
        // Wczytujemy z bazy liste pokoii do ktorej moze wejsc
        List<Integer> userRooms = getUserRooms();
        // Bierzemy id aktualnie zaznaczonego pokoju
        int roomId = getSelectedRoomId();
        // Jeśli na liście pokoii pracownika znajduje się zaznaczony pokój
        if (userRooms.contains(roomId) && maxScore > 0) {
            // Drzwi są otwierane
            SwingUtilities.invokeLater(() -> JOptionPane.showMessageDialog( parentComponent: EnterToRoom.this,
                    message: "Door opened!",
                    title: "Success",
                    JOptionPane.INFORMATION_MESSAGE));
            setFingerToScan();
            fingerInfoLabel.setText(String.format("Scan %s finger", fingerToScan));
        } else { // Jeśli pracownik nie może wejść do pokoju
            SwingUtilities.invokeLater(() -> JOptionPane.showMessageDialog( parentComponent: EnterToRoom.this,
                    message: "You are not able to enter this room!",
                    title: "Failed",
                    JOptionPane.INFORMATION_MESSAGE));
        }
    } else { // jesli uzytkownik nie znajduje się w bazie
        SwingUtilities.invokeLater(() -> JOptionPane.showMessageDialog( parentComponent: EnterToRoom.this,
                message: "You are not able to enter this room!",
                title: "Failed",
                JOptionPane.INFORMATION_MESSAGE));
    }
}
```

Rysunek 18. Metoda sprawdzająca dostęp pracownika do pomieszczenia

Jeśli pracownik istnieje w bazie, oraz wybrany pokój znajduje się na jego liście dostępu, to drzwi są otwierane, w przeciwnym razie drzwi nie otworzą się.



Rysunek 19. Autoryzowane wejście pracownika(sukces)



Rysunek 20. Brak autoryzacji wejścia(porażka)

```
private void processDB() throws IOException {
    if (subject == null) {  // test code
        subject = new NSubject();
        String imgPath = null;

        // Wybieramy zdjęcie odcisku w zależności od wylosowanego odcisku palca
        switch (fingerToScan.toLowerCase()) {
            case "thumb":
                imgPath = "C:\\Users\\Filip\\Desktop\\odciski\\PL_THUMB.jpg";
                break;
            case "pointing":
                imgPath = "C:\\Users\\Filip\\Desktop\\odciski\\PL_POINTING.jpg";
                break;
            case "middle":
                imgPath = "C:\\Users\\Filip\\Desktop\\odciski\\PL_MIDDLE.jpg";
                break;
            case "ring":
                imgPath = "C:\\Users\\Filip\\Desktop\\odciski\\PL_RING.jpg";
                break;
        }

        NFinger finger = new NFinger();

        finger.setFileName(imgPath);

        subject.getFingers().add(finger);
    }
    // ładujemy temapltes z bazy dancyh jak w przypadku kodu produkcyjnego
    loadTemplates();
}
```

Rysunek 21. Kod używany podczas testów

Użytkownik wchodząc w zakładkę EnterToRoom zobaczy komunikat jaki palec ma zeskanować, jest on losowany przy pomocy poniżej funkcji:

```java
private void setFingerToScan() {
    Random r = new Random();

    // losujemy numer z przedziału 1 do 4
    int finger = r.nextInt( bound: 4) + 1;

    // W zależności od wylosowanego numeru, ustawiamy odpowiedni palec
    switch (finger) {
        case 1:
            fingerToScan = "thumb";
            break;
        case 2:
            fingerToScan = "pointing";
            break;
        case 3:
            fingerToScan = "middle";
            break;
        case 4:
            fingerToScan = "ring";
            break;
    }
}
```

Rysunek 22. Losowanie palca do skanowania

Wnioski

Kompilacja

Powyższy kod kompilowany jest z pomocą *gradle*, które jest narzędziem do automatyzacji kompilacji. Gradle pozwala na napisanie tzw. zadań, które to zawierają polecenia, które zostaną wykonane w wierszu poleceń. W związku z błędami, które wystąpiły podczas prób kompilacji zgodnej z dokumentacją SDK „*gradle build clean”*, zmuszeni byliśmy do poszukiwania rozwiązań ww. problemu. Po dogłębnej analizie struktury plików gradle, udało nam się skompilować projekt z wykorzystaniem polecenia „*gradle run”,* które to uruchamia zadanie „run”, pomijając problematyczny kod znajdujący się w zadaniu „build”. Problematycznym kodem okazał się dodatek „The Distribution Plugin”, który pozwala na tworzenie różnych dystrybucji projektu. Problematyczna okazała się również kwestia licencji na oprogramowanie VeriFinger. Skrypt gradle odpowiedzialny za wykrycie plików licencyjnych sprawdza zawartość katalogu „Licenses” który powinien znajdować się dwa katalogi wyżej od katalogu projektu. Po przeniesieniu katalogu „Licenses” w odpowiednie miejsce, gradle był w stanie prawidłowo zweryfikować kwestie licencyjne.