

# Dokumentacja projektu realizowanego w ramach przedmiotu Analiza i Przetwarzanie Obrazów

APLIKACJA DO ZARZĄDZANIA DOKUMENTAMI

*Michał Rutkowski*

*Weronika Wronka*

*Eryk Zarębski*

*Zuzanna Sulima*

*Bartłomiej Wypart*

*Damian Łyszczarz*

*Adam Jędrychowski*

*Tomasz Koziół*

*Michał Rogowski*

## 1. Cele i założenia projektu

Celem projektu było stworzenie aplikacji pozwalającej użytkownikowi na dodawanie, edytowanie oraz wyszukiwanie odpowiednich dokumentów tożsamości na podstawie zdjęcia danego użytkownika. Początkowo baza danych miała zostać zasilona wygenerowanymi zdjęciami oraz odpowiednimi danymi. Również będzie istniała możliwość dodawania nowych rekordów na podstawie zdjęcia dokumentu tożsamości (jak np. karty rowerowej). Przy takiej konfiguracji, aplikacja podczas swojego działania powinna zapewnić możliwość wystania zdjęcia, które to następnie zostanie poddane analizie i jeżeli zostanie znaleziona w bazie danych taka osoba, to zwróci dane tej osoby, co jest głównym efektem oczekiwanym działania programu.

## 2. Opis zrealizowanego rozwiązania

W ramach przedmiotu "Analiza i Przetwarzanie Obrazów" został zrealizowany projekt, którego celem było stworzenie aplikacji umożliwiającej generowanie zdjęć, danych osobowych oraz dokumentów tożsamości, a także ich zapisywanie w bazie danych. Aplikacja miała pozwalać na późniejsze wyszukiwanie osób na podstawie dostarczonego zdjęcia oraz wyświetlanie powiązanych z nimi danych. Kluczowym założeniem projektu było stworzenie narzędzia typu CLI (Command Line Interface), co zapewniało łatwość obsługi i integracji z innymi systemami.

W projekcie wykorzystano bazę danych PostgreSQL wzbogaconą o rozszerzenie umożliwiające efektywne zapisywanie wektorów cech. Dzięki temu możliwe było szybkie i precyzyjne wyszukiwanie osób w bazie na podstawie analizowanych obrazów. Do generacji zdjęć użyto modelu StyleGAN2-ADA. Dodatkowo, dla ochrony generowanych zdjęć przed nieautoryzowanym użyciem, dodano znaki wodne.

Niniejsza dokumentacja opisuje cele i założenia projektu, a także szczegółowo przedstawia poszczególne etapy jego realizacji, od przygotowania modelu generacyjnego, przez proces generowania i zapisywania danych, aż po mechanizmy wyszukiwania i wyświetlania informacji o osobach.

### 3. Generowanie danych

W ramach projektu został przygotowany notatnik Jupyter Notebook, przy pomocy którego istnieje możliwość wygenerowania danych. Model do generacji twarzy został przygotowany z użyciem *StyleGAN2-ADA* oraz dataset *ffhq*. Dane, które są generowane przez działanie programu to:

- Wygenerowane zdjęcie osoby,
- Pięć dodatkowych zdjęć tej samej osoby, ale w delikatnie innych okolicznościach (np. zmienione tło, inne ułożenie włosów itp),
- Zdjęcie karty biblioteczej wraz z uzupełnionymi odpowiednimi polami na podstawie szablonu,
- Zdjęcie karty rowerowej wraz z uzupełnionymi odpowiednimi polami na podstawie szablonu.

Został dodany mechanizm, który umożliwia deterministyczne generowanie danych (seed). Przy takich samych wartościach, wygenerowane dane będą zawsze identyczne przy każdym uruchomieniu programu.



Rysunek 1. Przykład wygenerowanego dokumentu – karta rowerowa



Rysunek 2. Przykład wygenerowanego dokumentu – karta biblioteczna

## 4. Opis algorytmu

W projekcie wykorzystano algorytm Tesseract, czyli algorytm OCR (Optical Character Recognition), który zajmuje się ekstrakcją tekstu z obrazów. Zastosowany algorytm składa się z następujących kroków:

- wstępne przetwarzanie obrazu – konwersja do odcieni szarości, binaryzacja w celu lepszego rozróżnienia tekstu od tła oraz redukcja szumów
- detekcja i segmentacja obrazu – identyfikacja obszarów na obrazie z tekstem, wykrycie tekstu i słów
- rozpoznanie tekstu – analiza znalezionych znaków, klasyfikacja znaków korzystając z algorytmów uczenia maszynowego i późniejsza korekcja błędów
- zwrócenie znalezionej treści użytkownikowi, który jest potem poddawany ekstrakcji kluczowych informacji z wczytanego dokumentu

## 5. Struktura aplikacji

Projekt został napisany w języku Python. Katalog główny aplikacji składa się z:

- **document\_schemas/** – katalog, w którym zgromadzone są schematy poszczególnych dokumentów (w tym przypadku karta rowerowa oraz karta biblioteczna)
- **images/** – katalog, który zawiera wygenerowane zdjęcia poszczególnych osób wraz z kartą biblioteczną oraz rowerową. Każda osoba ma przypisany katalog person\_x (x – numer osoby), w której znajduje się siedem zdjęć tej osoby (jedno zdjęcie z wygenerowaną twarzą, pięć zdjęć wygenerowanych na podstawie twarzy z innymi ujęciami oraz zdjęcie ze znakiem wodnym) wraz z dokumentami.
- **Arial.ttf** – definicja czcionki, potrzebna do generowania tekstu przy tworzeniu dokumentów w pliku *Facial\_image\_generation.ipynb*.
- **config.ini** – zawiera informacje o adresie oraz danych uwierzytelniających do bazy danych.
- **docker-compose.yml** – zawiera informacje potrzebne do definiowania i konfigurowania aplikacji przy użyciu Docker-a. Określa usługi potrzebne do uruchomienia środowiska kontenerowego.
- **Requirements.txt** – zawiera informację o zależnościach, które są wymagane do prawidłowego działania aplikacji. Zawiera listę bibliotek i ich wersji.
- **init.sql** – zawiera skrypt sql potrzebny do stworzenia bazy danych.
- **ArgParser.py** – moduł odpowiedzialny za obsługę interakcji z użytkownikiem. Sprawdza konfigurację poszczególnych parametrów, podanych przez użytkownika..
- **ConfigParser.py** – Wczytuje informacje z pliku konfiguracyjnego.
- **EmbeddingExtractor.py** – Zawiera definicję modelu reset34 użytej do ekstrakcji cech ze zdjęcia. Wektoryzuje podane cechy celem późniejszego zapisu w bazie danych.
- **Logger.py** – zawiera klasę odpowiedzialną za logowanie danych w projekcie. Pozwala na określenie priorytetu danego komunikatu. Komunikaty wypisywane są na ekran w kolorze zależnych od wybranego formatu.
- **main.py** – główny plik projektu, który łączy funkcjonalności poszczególnych modułów. Łączy się z bazą danych oraz dostarcza całej logiki biznesowej aplikacji.
- **Facial\_image\_generation.ipynb** – notatnik Jupyter Notebook, zawierający kod pozwalający na deterministyczne wygenerowanie danych w postaci zdjęcia osób oraz dokumentów na podstawie podanych zdjęć i danych.

## 6. Uruchomienie aplikacji

Zaleca się używanie wirtualnego środowiska (venv) dla z Pythonem w wersji 3.10.12.

### 1. Utwórz wirtualne środowisko:

```
python3 -m venv myenv
```

### 2. Aktywuj wirtualne środowisko:

- *Dla Linux/Mac:*  
`source myenv/bin/activate`
- *Dla Windows:*  
`myenv\Scripts\activate`

### 3. Zainstaluj wymagane zależności:

```
pip install -r requirements.txt
```

### 4. Generowanie zdjęć twarzy

Aby wygenerować zdjęcia twarzy z znakami wodnymi wskazującymi, że obrazy zostały wygenerowane przez AI, uruchom notebook `Facial_image_generation.ipynb` i wykonaj odpowiedni kod. Należy pamiętać, że obrazy są zapisywane w bieżącym folderze, a aby uruchomić ostatnią komórkę odpowiedzialną za dodawanie znaków wodnych, konieczne jest dodanie pliku `Arial.ttf` zawartego w repozytorium do bieżącego folderu.

Ostateczne zdjęcia zarówno ze znakami wodnymi, jak i bez nich oraz wszystkie dokumenty zostały umieszczone w folderze `images`.

### 5. Uruchomienie programu wymaga docker-compose. Jeśli nie masz dockera, zainstaluj go:

```
sudo snap install docker
```

### 6. Gdy masz dockera, skonfiguruj bazę danych w drugim terminalu:

```
sudo docker-compose up --build
```

lub

```
sudo docker compose up --build
```

### 7. Uruchom program z wybranymi opcjami, np.:

```
python3 main.py --initDatabase --documentPhotoPath [ścieżka do zdj dokumentu]  
--clearDatabase
```

## 7. Prezentacja aplikacji

Aplikacja zawiera następujące funkcjonalności:

- Dodanie nowego użytkownika do bazy danych za pomocą wczytywanego dokumentu

Flagi      komendy      wywołania:      --initDatabase      --documentPhotoPath  
images/generated/person\_0/bicycle\_card.png --clearDatabase

```
Current database state:
-----
id      | name      | address      | phone_number | bicycle_card_id | student_card_id | student_class | embedding      | created_at
-----
Person data:
-----
id      | name      | address      | phone_number | bicycle_card_id | student_card_id | student_class | embedding      | created_at
-----
64      | Emily Smith | Smith Street, Ne| 555 444 333 | 89302745162     | None           | None         | [0.87139565,0.96] | 2024-06-23 21:27

Current database state:
-----
id      | name      | address      | phone_number | bicycle_card_id | student_card_id | student_class | embedding      | created_at
-----
64      | Emily Smith | Smith Street, Ne| 555 444 333 | 89302745162     | None           | None         | [0.87139565,0.96] | 2024-06-23 21:27
```

- Dodanie użytkownikowi nowej karty do bazy danych za pomocą nowego wczytywanego dokumentu

Flagi      komendy      wywołania:      --initDatabase      --documentPhotoPath  
images/generated/person\_0/university\_card.png --clearDatabase

```
Current database state:
-----
id      | name      | address      | phone_number | bicycle_card_id | student_card_id | student_class | embedding      | created_at
-----
65      | Emily Smith | Smith Street, Ne| 555 444 333 | 89302745162     | None           | None         | [0.87139565,0.96] | 2024-06-23 21:36

Person data:
-----
id      | name      | address      | phone_number | bicycle_card_id | student_card_id | student_class | embedding      | created_at
-----
65      | Emily Smith | Smith Street, Ne| 555 444 333 | 89302745162     | 89302745162    | 1A           | [0.87139565,0.96] | 2024-06-23 21:36

Current database state:
-----
id      | name      | address      | phone_number | bicycle_card_id | student_card_id | student_class | embedding      | created_at
-----
65      | Emily Smith | Smith Street, Ne| 555 444 333 | 89302745162     | 89302745162    | 1A           | [0.87139565,0.96] | 2024-06-23 21:36
```

- Wyświetlenie danych osoby na podstawie zdjęcia jej dokumentu

Flagi      komendy      wywołania:      --initDatabase      --personPhotoPath  
images/generated/person\_1/bicycle\_card.png

Karta rowerowa Emily Johnson

```
Current database state:
-----
id      | name      | address      | phone_number | bicycle_card_id | student_card_id | student_class | embedding      | created_at
-----
123     | Emily Smith | Smith Street, Ne| 555 444 333 | 89302745162     | 89302745162    | 1A           | [0.87139565,0.96] | 2024-06-23 21:51
125     | Emily Johnson | Johnson Street, | 123456789    | 98701234567     | 98701234567    | 1B           | [1.0365342,1.028] | 2024-06-23 21:51
127     | Michael Thompson | Thompson Street, | (555) 234-5678 | 67439105827     | 67439105827    | 2B           | [0.9098549,0.980] | 2024-06-23 21:51
129     | Minh Nguyen | Nguyen Street, C | (12) 456-7890 | 89123456789     | 89123456789    | 2D           | [0.8891604,1.029] | 2024-06-23 21:51
131     | Jenny Li | Li Street, New Y | (888) 555-1234 | 12345678901     | 12345678901    | 2A           | [1.0065488,0.993] | 2024-06-23 21:51
133     | Jason Chen | Chen Street, Was | (777) 222-3333 | 9865432109      | 9865432109     | 2D           | [0.92845684,0.92] | 2024-06-23 21:51
135     | Theo Demetriou | Demetriou Street | (333) 666-5555 | 11223344556     | 11223344556    | 1A           | [0.89740986,1.08] | 2024-06-23 21:51
137     | Amira Khan | Khan Street, New | (444) 555-6666 | 54321098765     | 54321098765    | 2c           | [0.93325865,0.97] | 2024-06-23 21:51

INFO | main.py:85 | Printing closest embeddings for photo: images/generated/person_1/bicycle_card.png

Closest embeddings:
-----
id      | name      | address      | phone_number | bicycle_card_id | student_card_id | student_class | embedding      | created_at | distance
-----
125     | Emily Johnson | Johnson Street, | 123456789    | 98701234567     | 98701234567    | 1B           | [1.0365342,1.028] | 2024-06-23 21:51 | 0.0
129     | Minh Nguyen | Nguyen Street, C | (12) 456-7890 | 89123456789     | 89123456789    | 2D           | [0.8891604,1.029] | 2024-06-23 21:51 | 1.63617211001287
137     | Amira Khan | Khan Street, New | (444) 555-6666 | 54321098765     | 54321098765    | 2c           | [0.93325865,0.97] | 2024-06-23 21:51 | 1.64724645026781
```

- Wyświetlenie danych osoby na podstawie zdjęcia jej twarzy

Flagi komendy wywołania: --initDatabase --personPhotoPath  
images/generated/person\_1/generated\_face\_10\_variation\_3.png

Jedno ze zdjęć Emily Johnson

```
Current database state:
```

id	name	address	phone_number	bicycle_card_id	student_card_id	student_class	embedding	created_at
123	Emily Smith	Smith Street, Ne	555 444 333	89302745162	89302745162	1A	[0.87139565,0.96]	2024-06-23 21:51
125	Emily Johnson	Johnson Street,	123456789	98701234567	98701234567	1B	[1.0365342,1.028]	2024-06-23 21:51
127	Michael Thompson	Thompson Street,	(555) 234-5678	67439105827	67439105827	2B	[0.9098549,0.980]	2024-06-23 21:51
129	Minh Nguyen	Nguyen Street, C	(12) 456-7890	89123456789	89123456789	2D	[0.8891604,1.029]	2024-06-23 21:51
131	Jenny Li	Li Street, New Y	(888) 555-1234	12345678901	12345678901	2A	[1.0065408,0.993]	2024-06-23 21:51
133	Jason Chen	Chen Street, Was	(777) 222-3333	9865432109	9865432109	2D	[0.92845684,0.92]	2024-06-23 21:51
135	Theo Demetriou	Demetriou Street	(333) 666-5555	11223344556	11223344556	1A	[0.89740986,1.08]	2024-06-23 21:51
137	Amira Khan	Khan Street, New	(444) 555-6666	54321098765	54321098765	2c	[0.93325865,0.97]	2024-06-23 21:51

```
INFO | main.py:85 | Printing closest embeddings for photo: images/generated/person_1/generated_face_10_variation_3.png
```

Closest embeddings:

id	name	address	phone_number	bicycle_card_id	student_card_id	student_class	embedding	created_at	distance
123	Emily Smith	Smith Street, Ne	555 444 333	89302745162	89302745162	1A	[0.87139565,0.96]	2024-06-23 21:51	2.17364320199803
133	Jason Chen	Chen Street, Was	(777) 222-3333	9865432109	9865432109	2D	[0.92845684,0.92]	2024-06-23 21:51	2.20238379597326
125	Emily Johnson	Johnson Street,	123456789	98701234567	98701234567	1B	[1.0365342,1.028]	2024-06-23 21:51	2.22707695622567

- Usunięcie osoby z bazy danych na podstawie jej dokumentu

Flagi komendy wywołania: --deleteRecord --documentPath  
images/generated/person\_1/bicycle\_card.png

Jedno ze zdjęć Emily Johnson

```
Current database state:
```

id	name	address	phone_number	bicycle_card_id	student_card_id	student_class	embedding	created_at
123	Emily Smith	Smith Street, Ne	555 444 333	89302745162	89302745162	1A	[0.87139565,0.96]	2024-06-23 21:51
125	Emily Johnson	Johnson Street,	123456789	98701234567	98701234567	1B	[1.0365342,1.028]	2024-06-23 21:51
127	Michael Thompson	Thompson Street,	(555) 234-5678	67439105827	67439105827	2B	[0.9098549,0.980]	2024-06-23 21:51
129	Minh Nguyen	Nguyen Street, C	(12) 456-7890	89123456789	89123456789	2D	[0.8891604,1.029]	2024-06-23 21:51
131	Jenny Li	Li Street, New Y	(888) 555-1234	12345678901	12345678901	2A	[1.0065408,0.993]	2024-06-23 21:51
133	Jason Chen	Chen Street, Was	(777) 222-3333	9865432109	9865432109	2D	[0.92845684,0.92]	2024-06-23 21:51
135	Theo Demetriou	Demetriou Street	(333) 666-5555	11223344556	11223344556	1A	[0.89740986,1.08]	2024-06-23 21:51
137	Amira Khan	Khan Street, New	(444) 555-6666	54321098765	54321098765	2c	[0.93325865,0.97]	2024-06-23 21:51

```
Person data:
```

id	name	address	phone_number	bicycle_card_id	student_card_id	student_class	embedding	created_at
125	Emily Johnson	Johnson Street,	123456789	98701234567	98701234567	1B	[1.0365342,1.028]	2024-06-23 21:51

```
Current database state:
```

id	name	address	phone_number	bicycle_card_id	student_card_id	student_class	embedding	created_at
123	Emily Smith	Smith Street, Ne	555 444 333	89302745162	89302745162	1A	[0.87139565,0.96]	2024-06-23 21:51
125	Emily Johnson	Johnson Street,	123456789	98701234567	98701234567	1B	[1.0365342,1.028]	2024-06-23 21:51
127	Michael Thompson	Thompson Street,	(555) 234-5678	67439105827	67439105827	2B	[0.9098549,0.980]	2024-06-23 21:51
129	Minh Nguyen	Nguyen Street, C	(12) 456-7890	89123456789	89123456789	2D	[0.8891604,1.029]	2024-06-23 21:51
131	Jenny Li	Li Street, New Y	(888) 555-1234	12345678901	12345678901	2A	[1.0065408,0.993]	2024-06-23 21:51
133	Jason Chen	Chen Street, Was	(777) 222-3333	9865432109	9865432109	2D	[0.92845684,0.92]	2024-06-23 21:51
135	Theo Demetriou	Demetriou Street	(333) 666-5555	11223344556	11223344556	1A	[0.89740986,1.08]	2024-06-23 21:51
137	Amira Khan	Khan Street, New	(444) 555-6666	54321098765	54321098765	2c	[0.93325865,0.97]	2024-06-23 21:51

```
INFO | DatabaseManager.py:71 | Person Emily Johnson deleted from database.
```

```
Current database state:
```

id	name	address	phone_number	bicycle_card_id	student_card_id	student_class	embedding	created_at
123	Emily Smith	Smith Street, Ne	555 444 333	89302745162	89302745162	1A	[0.87139565,0.96]	2024-06-23 21:51
127	Michael Thompson	Thompson Street,	(555) 234-5678	67439105827	67439105827	2B	[0.9098549,0.980]	2024-06-23 21:51
129	Minh Nguyen	Nguyen Street, C	(12) 456-7890	89123456789	89123456789	2D	[0.8891604,1.029]	2024-06-23 21:51
131	Jenny Li	Li Street, New Y	(888) 555-1234	12345678901	12345678901	2A	[1.0065408,0.993]	2024-06-23 21:51
133	Jason Chen	Chen Street, Was	(777) 222-3333	9865432109	9865432109	2D	[0.92845684,0.92]	2024-06-23 21:51
135	Theo Demetriou	Demetriou Street	(333) 666-5555	11223344556	11223344556	1A	[0.89740986,1.08]	2024-06-23 21:51
137	Amira Khan	Khan Street, New	(444) 555-6666	54321098765	54321098765	2c	[0.93325865,0.97]	2024-06-23 21:51



## 8. Wykorzystane biblioteki

W projekcie zostały użyte następujące biblioteki:

- **numpy** – podstawowa biblioteka służąca do obliczeń numerycznych zawierająca szeroki zestaw funkcji matematycznych
- **pillow** – biblioteka do przetwarzania obrazów
- **psycopg** – biblioteka obsługująca wsparcie bazy danych PostgreSQL dla Pythona, umożliwiające wykonywanie zapytań SQL
- **torch** – główna biblioteka do obliczeń tensorowych i uczenia maszynowego przydatna przy głębokim uczeniu
- **dnnlib** – biblioteka umożliwiająca zarządzanie oraz optymalizację architektur głębokich sieci neuronowych
- **legacy** – moduł, który umożliwia zapewnienie kompatybilności modeli oraz danych z różnych wersji frameworków AI
- **logging** – moduł Pythona służący do logowania komunikatów
- **argparse** – biblioteka do parsowania argumentów, umożliwiająca obsługę interakcji z użytkownikiem w aplikacjach konsolowych
- **configparser** – biblioteka do pracy z plikami konfiguracyjnymi w formacie .ini

## 9. Podział ról w grupie

- Michał Rutkowski - stworzenie i zarządzanie repozytorium projektu, serwerem do komunikacji, stworzenie szkieletu programu i jego interfejsu CLI, pomoc w refactoringu i łączeniu modułów, koordynowanie pracy zespołu
- Weronika Wronka - wygenerowanie zdjęć twarzy za pomocą StyleGAN2-ADA, dodanie znaków wodnych informujących o tym, że zdjęcie zostało utworzone za pomocą sztucznej inteligencji, przygotowanie szablonów dokumentów
- Eryk Zarębski - łączenie poszczególnych elementów rozwiązania, wektoryzacja zdjęć i wektorowa baza danych na dockerze
- Zuzanna Sulima - połączenie skanowania informacji z dokumentu z resztą aplikacji, usuwanie rekordu po dowolnym dokumencie, debugowanie
- Bartłomiej Wypart - generowanie zdjęć na podstawie wygenerowanych twarzy, generowanie dokumentów na podstawie szablonów, dokumentacja
- Damian Łyszczarz - implementacja ekstrakcji zdjęcia twarzy ze snaku dokumentu
- Adam Jędrychowski - implementacja skanera dokumentu, integracja z cli
- Tomasz Kozieł - implementacja skanowania tekstu z dokumentu
- Michał Rogowski - sporządzenie dokumentacji projektu