



**Politechnika
Śląska**

PROJEKT INŻYNIERSKI

Aplikacja do weryfikacji dwuetapowej Open2FA

**Dawid Grygiel
303166**

Kierunek: informatyka

Specjalność: Bezpieczeństwo systemów i aplikacji sieciowych

**PROWADZĄCY PRACĘ
dr inż. Adrian Kapczyński
WYDZIAŁ MATEMATYKI STOSOWANEJ**

GLIWICE 2024

Tytuł pracy: Aplikacja do weryfikacji dwuetapowej Open2FA

Streszczenie:

Open2FA jest aplikacją służącą do weryfikacji dwuetapowej z dodatkowym systemem przesyłania wygenerowanych kodów za pomocą kodów QR.

Głównym celem aplikacji jest generowanie jednorazowych kodów służących do potwierdzania tożsamości użytkownika podczas logowania. Open2FA wyróżnia się na tle innych aplikacji dzięki szyfrowaniu sekretów (kluczy używanych do generowania kodów) za pomocą hasła oraz możliwości ich zapisu do pliku. Dzięki temu użytkownik może łatwo przenieść swoje kody na inne urządzenie, przesyłając wyłącznie plik. Aplikacja obsługuje także system przekaźnikowy, umożliwiający przesyłanie zaszyfrowanych kodów jednorazowych bezpośrednio do innego urządzenia poprzez skanowanie kodu QR.

Cały projekt składa się z 3 aplikacji:

1. **Aplikacja Mobilna** - Główna aplikacja służąca do generowania i zarządzania kodami jednorazowymi
2. **Aplikacja Webowa** - Przykład wykorzystania systemu przekaźnikowego, umożliwiający odbiór kodów jednorazowych z telefonu.
3. **Aplikacja serwera przekaźnikowego** - Element systemu odpowiedzialny za przesyłanie zaszyfrowanych kodów z telefonu do aplikacji webowej

Słowa kluczowe:

Aplikacja, Aplikacja webowa, Aplikacja mobilna, Weryfikacja dwuetapowa, TOTP, React Native, React.js, Rust, Node.js,

Thesis title:

Two-factor authentication app Open2FA

Abstract:

Open2FA is an application designed for two-factor authentication, featuring an additional system for transmitting generated codes via QR codes.

The primary purpose of the application is to generate one-time codes used for verifying user's identity during login. Open2FA stands out from other applications by encrypting secrets (keys used to generate the codes) with a password and allowing them to be saved to a file. This enables users to easily transfer their codes to

another device by simply sending the file. The application also supports a relay system that facilitates the direct transmission of encrypted one-time codes to another device via QR code scanning.

The entire project consists of three applications:

1. **Mobile Application** - The main app used for generating and managing one-time codes.
2. **Web Application** - A demonstration of the relay system, enabling the reception of one-time codes from a smartphone.
3. **Relay Server Application** - A system component responsible for transmitting encrypted codes from the phone to the web application.

Keywords:

Application, Web application, Mobile application, Two-factor authentication, TOTP, React Native, React.js, Rust, Node.js

Spis treści

1. Wstęp	9
2. Zagadnienia wprowadzające	11
2.1. Wstęp	11
2.2. Algorytmy i definicje	11
2.2.1. TOTP	11
2.2.2. HOTP	11
2.2.3. HMAC (Hash-Based Message Authentication Code)	11
2.2.4. PBKDF2	12
2.2.5. Ataki typu brute-force	12
2.2.6. SHA-256	12
2.2.7. Advanced Encryption Standard (AES)	13
2.2.8. RSA	13
2.2.9. PEM (Privacy-Enhanced Mail)	13
2.2.10. RSA-OAEP (RSA Optimal Asymmetric Encryption Padding)	14
2.2.11. Base64	14
2.2.12. ASCII	15
2.2.13. TOML (Tom's Obvious, Minimal Language)	15
2.2.14. Kody QR	16
2.2.15. URL (Uniform Resource Locator)	16
2.2.16. Pamięć RAM	16
2.2.17. HTTP	17
2.2.18. Aplikacja Webowa	17
2.2.19. Aplikacja mobilna	17
2.3. Przebieg weryfikacji dwuetapowej TOTP	17
2.3.1. Rejestracja w systemie	17
2.3.2. Potwierdzanie tożsamości przy logowaniu do konta	18
3. Projekt aplikacji	19
3.1. Wstęp	19
3.2. Wykorzystane technologie - Aplikacja mobilna	19
3.2.1. JavaScript	19
3.2.2. Typescript	19

3.2.3. React Native	19
3.2.4. Expo	20
3.2.5. Redux Toolkit	20
3.2.6. Git	20
3.2.7. CryptoES	20
3.2.8. i18next	21
3.2.9. Tailwind	21
3.2.10. node-forge	21
3.2.11. react-native-toast-message	21
3.2.12. otpauth	21
3.2.13. TOML	22
3.2.14. Zod	22
3.2.15. JSON	22
3.3. Wykorzystane technologie - Aplikacja webowa	22
3.3.1. React	22
3.3.2. Vite	23
3.3.3. React-Router	23
3.3.4. Typescript	23
3.3.5. tailwind	24
3.3.6. React-toastify	24
3.3.7. Socket.IO	24
3.3.8. Lodash	24
3.3.9. REST API (Representational State Transfer)	25
3.3.10. Express	25
3.3.11. JSON Web Tokens (JWT)	25
3.3.12. otpauth	25
3.3.13. node-qrcode	26
3.3.14. SQLite	26
3.3.15. node-sqlite3	26
3.3.16. sequelize	26
3.3.17. bcrypt	26
3.3.18. Postman	27
3.4. Wykorzystane technologie - Serwer przekaźnikowy	27
3.4.1. Rust	27
3.4.2. Axum	27
3.4.3. Tokio	27
3.4.4. Serde	28

3.4.5. Postman	28
3.5. UI Aplikacji	28
3.5.1. Paleta kolorów	28
3.5.2. Dostarczanie informacji	29
3.5.3. Nawigacja w aplikacji mobilnej	31
3.5.4. Nawigacja w aplikacji webowej	32
4. Implementacja	33
4.1. Wstęp	33
4.2. Przekazywanie kodów systemem serwerów przekaźnikowych	33
4.2.1. Proces weryfikacji dwuetapowej z pomocą systemu serwerów przekaźnikowych	34
4.3. System zapisywania danych do szyfrowanego pliku	34
4.4. Odczytywanie danych z kodu QR przy rejestracji nowego sekretu konta	36
4.5. Serwer przekaźnikowy	37
4.6. Testowanie aplikacji	38
4.6.1. Aplikacja webowa	38
4.6.2. Aplikacja mobilna	38
4.6.3. System przekaźnikowy	39
4.7. Instrukcja użytkowania aplikacji mobilnej	39
4.7.1. Inicjalizacja aplikacji - Tworzenie oraz załadowywanie pliku . .	39
4.7.2. Rejestrowanie nowego sekretu konta	44
4.7.3. Usuwanie sekretu konta	49
4.7.4. Przekazanie kodu jednorazowego do serwera przekaźnika	51
4.7.5. Zmiana języka oraz zamknięcie pliku	53
4.8. Instrukcja użytkowania aplikacji Webowej	54
4.8.1. Tworzenie użytkownika i logowanie	54
4.8.2. Skonfigurowanie weryfikacji dwuetapowej	56
4.8.3. Korzystanie z serwera przekaźnika	57
5. Podsumowanie	59
References	61

1. Wstęp

W coraz bardziej cyfrowym świecie ochrona kont online stała się ważniejsza niż kiedykolwiek. Banki, dostawcy poczty e-mail, a nawet platformy do gier zachęcają do skonfigurowania uwierzytelniania dwuetapowego. W miarę jak technologia coraz bardziej się rozwija, coraz częściej dochodzi do prób przejmowania kont użytkowników. Bezpieczeństwo danych osobowych staje się powoli priorytetem zarówno dla firm, jak i dla osób prywatnych. Jednym z najskuteczniejszych sposobów ochrony kont przed nieautoryzowanym dostępem jest weryfikacja dwuetapowa (Two-Factor Authentication, 2FA). System ten, wymagający podania dwóch niezależnych czynników uwierzytelniających, na przykład hasła oraz kodu jednorazowego, znacząco podnosi poziom bezpieczeństwa.

Mimo skuteczności weryfikacji dwuetapowej, wiele użytkowników unika jego stosowania, tłumacząc to trudnościami związanymi z obsługą takich systemów, w szczególności podczas zmiany urządzenia. W odpowiedzi na te wyzwania powstała aplikacja Open2FA. Celem aplikacji jest nie tylko zapewnienie wygodnego narzędzia do generowania jednorazowych kodów, ale także uproszczenie procesu przenoszenia danych pomiędzy urządzeniami.

Kluczowym ułatwieniem w Open2FA jest możliwość szyfrowania sekretów za pomocą hasła oraz ich zapisywania do pliku. Dzięki temu użytkownik może w prosty sposób przenieść swoje dane na nowe urządzenie, przesyłając jedynie plik. Dodatkowo aplikacja oferuje nowatorskie rozwiązanie w postaci systemu przesyłania jednorazowych kodów za pomocą kodów QR. Umożliwia to użytkownikom szybkie i bezpieczne przesłanie kodu z jednego urządzenia na drugie, bez konieczności ręcznego wpisywania danych.

Celem pracy jest analiza i omówienie działania aplikacji Open2FA, w tym jej architektury oraz unikalnych funkcjonalności. Przedstawione zostaną również rozwiązania techniczne zastosowane w aplikacji, które mają na celu zwiększenie bezpieczeństwa i wygody użytkowników. Praca ma na celu nie tylko zaprezentowanie rozwiązań technicznych, ale również wykazanie ich przydatności w kontekście codziennego użytkowania.

W pierwszym rozdziale przedstawione zostaną kluczowe pojęcia i mechanizmy związane z weryfikacją dwuetapową, w tym funkcje matematyczne i proces uwierzytelniania użytkowników. W drugim rozdziale omówione zostaną technologie oraz proces projektowania aplikacji, ze szczególnym uwzględnieniem interfejsu użytkownika.

Trzeci rozdział skupi się na szczegółowym przedstawieniu implementacji systemu, w tym architektury aplikacji, działania systemu przekaźników, obsługi plików zaszyfrowanych oraz kodów QR. Opisano w nim również proces testowania i instrukcję obsługi aplikacji. Pracę zamyka podsumowanie, które syntetyzuje omówione zagadnienia i ocenia efektywność proponowanych rozwiązań.

2. Zagadnienia wprowadzające

2.1. Wstęp

W tym rozdziale zostanie omówione działanie systemów weryfikacji dwuetapowej oraz kluczowe pojęcia związane z ich funkcjonowaniem. Na początku przedstawione zostaną najważniejsze skróty oraz funkcje matematyczne wykorzystywane we wszystkich aplikacjach. Następnie zostanie zaprezentowany proces uwierzytelniania użytkownika, ilustrujący sposób potwierdzania tożsamości w aplikacjach korzystających z weryfikacji dwuetapowej.

2.2. Algorytmy i definicje

2.2.1. TOTP

To skrót od Time-based One-Time Passwords i jest najbardziej powszechną formą uwierzytelniania dwuskładnikowego. Unikalne hasła numeryczne są generowane przy użyciu znormalizowanego algorytmu, który wykorzystuje bieżący czas jako dane wejściowe. Hasła czasowe są dostępne w trybie offline i zapewniają przyjazne dla użytkownika, zwiększone bezpieczeństwo konta, gdy są używane jako drugi czynnik.

2.2.2. HOTP

To skrót od HMAC-based One-Time Password i jest oryginalnym standardem, na którym oparto TOTP. Obie metody wykorzystują tajny klucz jako jedno z danych wejściowych, ale podczas gdy TOTP wykorzystuje czas systemowy jako drugie dane wejściowe, HOTP wykorzystuje licznik, który zwiększa się przy każdym nowym zatwierdzeniu. W HOTP obie strony zwiększają licznik i używają go do obliczenia hasła jednorazowego.

Podczas gdy HOTP jest nadal używany, aplikacje uwierzytelniające konsumentów, takie jak Authy, Google Authenticator oraz Open2FA, implementują standard TOTP.

2.2.3. HMAC (Hash-Based Message Authentication Code)

Technika kryptograficzna, która zapewnia integralność i autentyczność danych przy użyciu funkcji skrótu i tajnego klucza. W przeciwieństwie do podejść opartych na podpisach i kryptografii asymetrycznej. Sprawdzanie integralności danych jest

niezbędne dla stron zaangażowanych w komunikację. HTTPS i inne protokoły transferu wykorzystują HMAC. Kryptograficzną funkcją skrótu może być MD-5, SHA-1 lub SHA-256. Podpisy cyfrowe są prawie podobne do HMAC, tj. oba wykorzystują funkcję skrótu i klucz współdzielony. Różnica polega na kluczach, tj. HMAC wykorzystuje klucz symetryczny (ta sama kopia), podczas gdy podpisy wykorzystują klucz asymetryczny (dwa różne klucze).

2.2.4. PBKDF2

Funkcja generowania klucza używana do bezpiecznego przetwarzania haseł. Jest to jeden z algorytmów opartych na kryptograficznej funkcji haszującej, który jest zaprojektowany, aby zwiększyć czas potrzebny do złamania hasła przez atak brute-force. PBKDF2 dodaje losową sól i wykonuje funkcję haszującą wiele razy, co skutecznie zwiększa koszt takiego ataku. Dzięki temu, nawet jeśli algorytm haszujący lub moc obliczeniowa rosną, hasło pozostaje trudne do odszyfrowania. Jest szeroko stosowany w systemach zabezpieczeń, takich jak hasła do aplikacji internetowych, gdzie bezpieczeństwo haseł jest kluczowe.

2.2.5. Ataki typu brute-force

Metoda łamania haseł polegająca na wypróbowaniu wszystkich możliwych kombinacji znaków w celu uzyskania dostępu do zaszyfrowanego lub zabezpieczonego konta. Jest to najbardziej prymitywna i najprostsza metoda, ponieważ polega na systematycznym testowaniu każdej możliwej opcji. W kontekście łamania haseł, atak brute-force może obejmować próbę wpisania każdej możliwej kombinacji liter, cyfr i specjalnych znaków, zaczynając od haseł najprostszymi i najbardziej popularnymi, aż po najbardziej złożone.

W praktyce, ataki brute-force mogą być nieskuteczne wobec haseł o wysokim poziomie złożoności, które są długie, przypadkowe i zawierają kombinacje cyfr, liter oraz znaków specjalnych. Jednak przy słabszych hasłach, takich jak "123456" czy "hasło", ataki brute-force mogą być skuteczne, nawet przy użyciu niewielkiej mocy obliczeniowej. Ochrona przed atakami brute-force może być poprawiona przez zastosowanie technik, takich jak ograniczenia liczby prób logowania, czasowe blokowanie po nieudanych próbach, i używanie zaawansowanych algorytmów haszujących, takich jak PBKDF2.

2.2.6. SHA-256

Algorytm skrótu, który przekształca dane o dowolnej długości w 256-bitowy skrót. Proces hashowania polega na przetwarzaniu danych w blokach o rozmiarze

512 bitów, przy użyciu zaawansowanych funkcji logicznych i matematycznych operacji. Algorytm wykonuje skomplikowane operacje, takie jak rotacje bitów, sumowanie i mieszanie, aby stworzyć złożony skrót. Wynikowy skrót jest unikalną matematyczną reprezentacją danych wejściowych, co zapewnia odporność na kolizje i zwiększa bezpieczeństwo danych. Nawet niewielkie zmiany w danych wejściowych prowadzą do drastycznie różnego skrótu.

2.2.7. Advanced Encryption Standard (AES)

Nowoczesny algorytm szyfrowania symetrycznego, powszechnie stosowany do ochrony danych w różnych aplikacjach i systemach. W 2001 roku został zatwierdzony przez Narodowy Instytut Standaryzacji i Technologii (NIST) jako standard szyfrowania. AES używa tego samego klucza do szyfrowania i deszyfrowania danych. Klucz musi być znany zarówno nadawcy, jak i odbiorcy.

2.2.8. RSA

Algorytm szyfrowania asymetrycznego, który wykorzystuje dwa klucze: publiczny do szyfrowania i prywatny do deszyfrowania. Klucz publiczny można swobodnie udostępniać, podczas gdy klucz prywatny musi być przechowywany w tajemnicy. Bezpieczeństwo RSA opiera się na matematycznej trudności rozkładu dużych liczb na czynniki pierwsze, co sprawia, że proces odszyfrowania danych bez znajomości klucza prywatnego jest praktycznie niemożliwy.

Podczas generowania kluczy wybierane są dwie duże liczby pierwsze, które są używane do obliczenia wartości kluczy. Szyfrowanie polega na przekształceniu danych przy użyciu klucza publicznego i specjalnej operacji matematycznej, podczas gdy deszyfrowanie wykorzystuje klucz prywatny do odtworzenia oryginalnych danych. Rozmiar kluczy, zazwyczaj od 2048 do 4096 bitów, wpływa na poziom bezpieczeństwa oraz czas potrzebny na przetwarzanie.

RSA znajduje zastosowanie w ochronie danych i uwierzytelnianiu. Dzięki niemu można szyfrować poufne informacje, podpisywać cyfrowo dokumenty w celu potwierdzenia ich autentyczności oraz bezpiecznie wymieniać klucze szyfrowania w innych protokołach.

2.2.9. PEM (Privacy-Enhanced Mail)

Format używany do przechowywania i wymiany kluczy kryptograficznych, certyfikatów oraz innych danych związanych z kryptografią, takich jak żądania podpisania certyfikatu czy certyfikaty CA. Format PEM jest szeroko stosowany w systemach opartych na kryptografii, takich jak TLS/SSL.

Pliki w formacie PEM są zapisane jako tekst czytelny dla człowieka, co odróżnia je od binarnych formatów takich jak DER. Dane są najpierw kodowane za pomocą Base64, a następnie otaczane specjalnymi nagłówkami i stopkami, które określają typ zawartych danych.

2.2.10. RSA-OAEP (RSA Optimal Asymmetric Encryption Padding)

Schemat kryptograficzny, który rozszerza standardowe szyfrowanie RSA, dodając mechanizm bezpiecznego wypełnienia danych (padding). Został zaprojektowany, aby zwiększyć bezpieczeństwo RSA, eliminując słabości związane z szyfrowaniem surowego tekstu przy użyciu klucza publicznego.

2.2.11. Base64

Metoda kodowania danych binarnych w formacie tekstowym, która jest często stosowana w systemach komputerowych do przesyłania i przechowywania danych w sposób bezpieczny i zgodny z różnymi standardami. Base64 działa, zamieniając dane binarne na zestaw czytelnych znaków ASCII.

2.2.12. ASCII

Standard kodowania znaków, który przypisuje liczby do liter, cyfr, symboli i znaków sterujących, umożliwiając ich reprezentację w systemach komputerowych.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	,	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	I	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	-	127	7F	[DEL]

Rysunek 1: Tabela znaków ASCII - Źródło: [44]

2.2.13. TOML (Tom's Obvious, Minimal Language)

Format pliku konfiguracyjnego, który jest prosty, czytelny i przeznaczony do jednoznacznego mapowania danych w pliku tekstowym. TOML jest używany do przechowywania danych konfiguracyjnych w aplikacjach internetowych i nie tylko, a jego celem jest być łatwym do odczytania przez człowieka i maszynę.

Format TOML składa się z sekcji, kluczy i wartości, które są zapisane w formacie klucz-wartość. Klucze są oddzielane od wartości dwukropkiem, a wartości mogą być różnego typu, takich jak liczby, ciągi tekstowe, daty, listy czy inne struktury danych.

Listing 1: ”Przykład danych pliku przed szyfrowaniem”

```
[owner]
name = "Tom Preston-Werner"
dob = 1979-05-27T07:32:00Z
title = "Founder"

[database]
server = "10.0.0.12"
ports = [8001, 8001, 8002]
connection_max = 5000
enabled = true
```

2.2.14. Kody QR

Dwuwymiarowe kody kreskowe, które mogą przechowywać różnego rodzaju informacje w kompaktowej formie. W przeciwieństwie do zwykłych kodów kreskowych, kody QR mogą być odczytywane w dwóch wymiarach (poziomo i pionowo), co pozwala im na przechowywanie znacznie większej ilości danych. Kod QR składa się z czarno-białych modułów (kwadracików) ułożonych na kwadratowej siatce. Te moduły przechowują zakodowane informacje, które mogą obejmować tekst, adresy URL, dane kontaktowe, lokalizacje GPS, hasła Wi-Fi i wiele innych.

2.2.15. URL (Uniform Resource Locator)

Standardowy sposób określania lokalizacji zasobów w internecie, takich jak strony internetowe, pliki czy usługi. Składa się z kilku elementów, takich jak protokół określający sposób komunikacji (np. https), nazwa hosta wskazująca serwer (np. www.example.com), ścieżka prowadząca do konkretnego zasobu oraz opcjonalnie parametry zapytania, które modyfikują żądanego fragment, który wskazuje określoną część zasobu. URL jest kluczowym elementem działania sieci, umożliwiającym dostęp do różnorodnych treści oraz usług online.

2.2.16. Pamięć RAM

Rodzaj szybkiej, ulotnej pamięci komputerowej, która służy do przechowywania danych i instrukcji potrzebnych do bieżącego działania systemu i uruchamianych aplikacji. Jest to pamięć o dostępie swobodnym, co oznacza, że dane mogą być odczytywane i zapisywane w dowolnym miejscu z jednakową prędkością. RAM jest kluczowy dla wydajności komputera, ponieważ umożliwia szybki dostęp do infor-

macji, eliminując konieczność korzystania z wolniejszych nośników danych, takich jak dyski twardy. Po wyłączeniu zasilania zawartość pamięci RAM zostaje utracona.

2.2.17. HTTP

Protokół komunikacyjny wykorzystywany do przesyłania danych w sieci, głównie w Internecie. Umożliwia wymianę informacji między klientem, takim jak przeglądarka internetowa, a serwerem, na którym znajdują się zasoby, takie jak strony internetowe, obrazy czy pliki. Protokół działa w modelu klient-serwer i jest bezstanowy, co oznacza, że każde żądanie jest niezależne i nie zawiera informacji o wcześniejszych interakcjach. W komunikacji wykorzystuje różne metody, takie jak GET do pobierania danych, POST do przesyłania informacji czy DELETE do usuwania zasobów. Odpowiedzi serwera są opatrzone kodami statusu, które wskazują wynik operacji, np. 200 OK oznacza sukces, a 404 Not Found informuje o braku zasobu. Dla zwiększenia bezpieczeństwa stosuje się szyfrowaną wersję protokołu, HTTPS, która chroni dane przesyłane między klientem a serwerem.

2.2.18. Aplikacja Webowa

Program komputerowy działający w przeglądarce internetowej, który umożliwia interakcję użytkownika z różnymi funkcjami i danymi za pośrednictwem sieci. Działa na zasadzie komunikacji między klientem (przeglądarką) a serwerem, gdzie aplikacja jest przechowywana i przetwarzana. Użytkownik nie musi instalować aplikacji na swoim urządzeniu, wystarczy dostęp do Internetu i przeglądarka.

2.2.19. Aplikacja mobilna

Program zaprojektowany do działania na urządzeniach przenośnych, takich jak smartfony i tablety. Może być instalowana za pośrednictwem sklepów z aplikacjami, takich jak Google Play czy App Store, i jest dostosowana do specyfiki interfejsu dotykowego. Aplikacje mobilne mogą działać w trybie online, łącząc się z Internetem, lub offline, bez konieczności dostępu do sieci.

2.3. Przebieg weryfikacji dwuetapowej TOTP

2.3.1. Rejestracja w systemie

Żeby zabezpieczyć konto systemem weryfikacji dwuetapowej, najpierw należy zarejestrować taką chęć na serwerze. Podczas rejestracji serwer generuje unikalny klucz tajny (tzw. shared secret), który jest przypisany do konta użytkownika. Ten sekret, wraz z innymi danymi, jest wysyłany do aplikacji użytkownika, np. dane o

wystawcy sekretu, odstęp czasu między wygenerowanymi kluczami, długość kodu jednorazowego, algorytm haszujący, typ algorytmu OTP (np. TOTP czy HOTP) oraz etykieta, która zwykle informuje, do jakiego konta przypisany jest ten sekret.

Następnie sekret jest przedstawiany użytkownikowi w formie kodu QR. Użytkownik skanuje ten kod za pomocą aplikacji na swoim telefonie, która odczytuje dane z kodu QR i zapisuje je bezpiecznie w pamięci telefonu. Po skonfigurowaniu danych, aplikacja używa zapisany sekret do generowania kodów jednorazowych. Pobiera aktualny czas i dzieli go na określone przedziały (zwykle co 30 sekund), po czym oblicza skrót HMAC z użyciem tego sekretu. Wynik jest skracany do określonej długości, tworząc kod, który może być użyty do weryfikacji przynależności do konta.

By potwierdzić rejestrację, użytkownik wpisuje wygenerowany kod w odpowiednie pole w aplikacji, a następnie przesyła go na serwer. Serwer sprawdza kod, używając tego samego klucza, i porównuje kody dla kilku sąsiednich przedziałów czasowych, aby uwzględnić drobne przesunięcia zegara. Jeśli kod zgadza się z obliczonym przez serwer kodem, sekret jest na stałe przypisywany do konta użytkownika, umożliwiając przyszłe potwierdzanie przynależności do konta za pomocą kodu wygenerowanego z aplikacji.

2.3.2. Potwierdzanie tożsamości przy logowaniu do konta

Podczas logowania do konta z systemem weryfikacji dwuetapowej, proces potwierdzenia tożsamości wygląda nieco inaczej.

Najpierw, gdy użytkownik chce zalogować się do swojego konta, musi on podać hasło. Następnie, system prosi o wprowadzenie jednorazowego kodu wygenerowanego przez aplikację na telefonie, która wcześniej została skonfigurowana podczas rejestracji w systemie weryfikacji dwuetapowej.

Aplikacja na telefonie, która odczytała dane z kodu QR podczas rejestracji, używa zapisany sekret do obliczenia kodu jednorazowego na podstawie aktualnego czasu. Kod ten jest generowany co określony czas i zmienia się w każdym przedziale czasowym. Użytkownik wprowadza ten kod w odpowiednim polu na stronie logowania.

Serwer logowania, korzystając z tego samego sekretu, również generuje kod na podstawie aktualnego czasu i porównuje go z kodem wprowadzonym przez użytkownika. Jeżeli te kody są zgodne, oznacza to, że użytkownik jest rzeczywiście tą osobą, za którą się podaje, a dostęp do konta jest przyznany.

3. Projekt aplikacji

3.1. Wstęp

W tym rozdziale opisane zostaną poszczególne części projektowania poszczególnych aplikacji. Pierwszą częścią, na którą zostanie zwrócona uwaga to przegląd oraz opis technologii wykorzystytych w aplikacjach. Następnie zostanie przedstawiony proces projektowania interfejsu konkretnych aplikacji.

3.2. Wykorzystane technologie - Aplikacja mobilna

3.2.1. JavaScript

Język programowania wysokiego poziomu, używany głównie do tworzenia stron internetowych. Jest wykorzystywany też w innych środowiskach, takich jak Node.js, Apache CouchDB czy Adobe Acrobat. Jest językiem opartym na prototypach, wieloparadygmatowym, jednowątkowym i dynamicznym, obsługującym style programowania takie jak obiektowe, imperatywne i deklaratywne, w tym programowanie funkcyjne.

3.2.2. Typescript

Nakładka na język JavaScript, która dodaje dodatkową składnię, wspierającąścielszą integrację z edytarami i umożliwiającą wczesne wychwytywanie błędów. Kod TypeScript jest konwertowany na JavaScript, co pozwala na jego działanie wszędzie tam, gdzie działa JavaScript – w przeglądarkach, na platformach takich jak Node.js, Deno, Bun i innych aplikacjach. TypeScript rozumie JavaScript i wykorzystuje wnioskowanie o typach, aby zapewnić doskonałe narzędzia bez konieczności dodatkowego kodowania.

3.2.3. React Native

Framework typu open source do tworzenia aplikacji na Androïda i iOS przy użyciu Reacta i natywnych możliwości platformy aplikacji. W React Native używa się języka JavaScript do uzyskiwania dostępu do interfejsów API platformy, a także do opisywania wyglądu i zachowania interfejsu użytkownika za pomocą komponentów

React: pakietów kodu wielokrotnego użytku z możliwością zagnieżdżania. Umożliwia on użycie natywnych komponentów oraz oferuje wysoką wydajność poprzez renderowanie komponentów UI przy użyciu natywnych API.

3.2.4. Expo

Framework, który upraszcza tworzenie aplikacji na Androida i iOS przy użyciu frameworka React Native. Zmniejsza skomplikowanie procesów programowania, udostępniając zestaw narzędzi, bibliotek i usług, które ułatwiają typowe zadania, takie jak instalacja i konfiguracja. Dzięki temu eliminuje potrzebę zaawansowanej konfiguracji, umożliwiając szybszy rozwój aplikacji mobilnych.

3.2.5. Redux Toolkit

Zestaw narzędzi do zarządzania stanem w aplikacjach JavaScript, głównie w aplikacjach React. Uproszcza tworzenie, konfigurację i zarządzanie stanem aplikacji poprzez oferowanie gotowych funkcji i metod, takich jak „createSlice” do tworzenia reducerów i akcji. Redux Toolkit minimalizuje potrzebę ręcznego pisania kodu związanego z zarządzaniem stanem, co pozwala na łatwiejsze zarządzanie stanem globalnym i lokalnym aplikacji, poprawiając jednocześnie czytelność i utrzymanie kodu.

3.2.6. Git

System kontroli wersji, który umożliwia śledzenie zmian w kodzie źródłowym podczas tworzenia oprogramowania. Opracowany przez Linusa Torvaldsa, pozwala wielu programistom na jednoczesną współpracę nad projektem. Rejestruje zmiany w plikach i katalogach, śledzi historię modyfikacji oraz ułatwia współpracę, umożliwiając programistom pracę na różnych gałęziach bazy kodu. Git pozwala na łatwe rozgałęzianie, łączenie i zarządzanie kodem, co umożliwia niezależną pracę nad różnymi funkcjonalnościami projektu i późniejsze integrację tych zmian.

3.2.7. CryptoES

Biblioteka algorytmów kryptograficznych zgodna z ES6 i TypeScript. Jest inspirowana CryptoJS, oferując podobne API, ale została napisana zgodnie z najnowszym standardem ECMAScript. Obsługuje importy modułów ES6 oraz częściowy import, co sprawia, że jest idealna do użycia w środowisku TypeScript. CryptoES zapewnia typy dla TypeScriptu, co ułatwia integrację z projektami Type-Scriptowymi, oferując zaawansowane narzędzia kryptograficzne z użyciem nowoczesnych standardów.

3.2.8. i18next

Framework do internacjonalizacji (i18n) napisany w JavaScript. Umożliwia łatwe dodanie obsługi wielu języków do aplikacji JavaScript, ułatwiając zarządzanie tłumaczeniami i zmianą języka. i18next wspiera różne formaty plików tłumaczeń i integruje się zarówno z frontendem, jak i backendem, co czyni go wszechstronnym narzędziem do implementacji internacjonalizacji w aplikacjach JavaScript.

3.2.9. Tailwind

Tailwind CSS działa poprzez analizowanie wszystkich plików HTML, komponentów JavaScript i innych szablonów w celu znalezienia nazw klas, generując odpowiednie style, które są następnie zapisywane w statycznym pliku CSS. Jest szybki, elastyczny i niezawodny, oferując zerowy czas wykonywania, co oznacza, że nie wymaga dynamicznego przetwarzania w czasie wykonania, co przyspiesza wydajność aplikacji.

3.2.10. node-forge

W pełni natywna implementacja protokołu TLS w języku JavaScript, a także zestaw narzędzi do tworzenia aplikacji internetowych, które wykorzystują różne zasoby sieciowe. Oferuje wsparcie dla szeregu funkcji kryptograficznych, takich jak tworzenie i obsługa certyfikatów, szyfrowanie, podpisy cyfrowe oraz generowanie kluczy. Dzięki temu pozwala na zaawansowane operacje kryptograficzne bez potrzeby użycia dodatkowego oprogramowania czy bibliotek zewnętrznych.

3.2.11. react-native-toast-message

Biblioteka do tworzenia i zarządzania powiadomieniami typu toast w aplikacjach napisanych w React Native. Jest to narzędzie, które umożliwia wyświetlanie krótkich, konfigurowalnych powiadomień, które pojawiają się na ekranie przez określony czas, a następnie zanikają. Jest szeroko stosowana w aplikacjach mobilnych do przekazywania użytkownikom krótkich komunikatów o różnych zdarzeniach, takich jak udane operacje, błędy, czy potwierdzenia.

3.2.12. otpauth

Biblioteka haseł jednorazowych (OTP) przeznaczona dla Node.js, Deno, Bun oraz przeglądarek. Obsługuje generowanie i walidację haseł jednorazowych opartych na HMAC (HOTP) zgodnie z RFC 4226 oraz hasła jednorazowe oparte na czasie (TOTP) zgodnie z RFC 6238. Często stosowana w systemach uwierzytelniania dwuetapowego, umożliwiając bezpieczne i wygodne uwierzytelnienie użytkowników.

3.2.13. TOML

Minimalny format pliku obiektów językowych, zaprojektowany z myślą o czytelności i prostocie dzięki intuicyjnej i jasnej składni. Jego celem jest jednoznaczne mapowanie na tablicę mieszającą (hash map). TOML został stworzony tak, aby był łatwy do przetworzenia na struktury danych w wielu językach programowania, co czyni go uniwersalnym rozwiązaniem do przechowywania konfiguracji.

3.2.14. Zod

Biblioteka do deklaracji i walidacji schematów danych, stworzona z myślą o TypeScript. Pozwala opisywać dowolne typy danych — od prostych, jak ciągi znaków, po złożone, zagnieżdżone obiekty.

Zod został zaprojektowany tak, aby maksymalnie ułatwić pracę programistom. Kluczowym celem jest eliminacja powielania deklaracji typów — wystarczy raz zdefiniować validator, a Zod automatycznie wywnioskuje odpowiedni typ dla TypeScript. Dzięki temu można łatwo łączyć proste typy w bardziej złożone struktury danych.

3.2.15. JSON

Uniwersalny i lekki format wymiany danych, który pełni rolę mostu między różnymi aplikacjami i systemami. Dzięki swojej prostej, zrozumiałej składni opierającej się na parach klucz-wartość, JSON pozwala na łatwe przekazywanie danych między klientem a serwerem, a także integrację między aplikacjami napisanymi w różnych językach programowania. Jego struktura, obejmująca tablice i obiekty, umożliwia modelowanie złożonych danych, jednocześnie zachowując czytelność i kompaktość. JSON jest kluczowym elementem w komunikacji API i odgrywa ważną rolę w połączeniach między systemami, takich jak aplikacje webowe, mobilne, czy bazy danych.

3.3. Wykorzystane technologie - Aplikacja webowa

3.3.1. React

Biblioteka JavaScript stworzona przez Facebooka, zaprojektowana do budowy dynamicznych i interaktywnych interfejsów użytkownika w aplikacjach internetowych. React pełni rolę mostu między logiką aplikacji a jej wizualnym wyglądem, umożliwiając efektywne zarządzanie stanem i renderowaniem komponentów w odpowiedzi na zmiany danych.

React opiera się na koncepcji komponentów — niewielkich, wielokrotnego użytku bloków kodu, które opisują część interfejsu użytkownika. Dzięki wirtualnemu DOM (Virtual DOM), biblioteka optymalizuje aktualizacje interfejsu, minimalizując bezpośrednie manipulacje na rzeczywistym DOM i zwiększać wydajność aplikacji.

React znajduje zastosowanie w aplikacjach jednostronnicowych (SPA) oraz wszędzie tam, gdzie istotne są szybkie i responsywne interfejsy. Jego modularność i wsparcie przez dużą społeczność sprawiają, że jest jednym z najpopularniejszych narzędzi do tworzenia nowoczesnych aplikacji webowych.

3.3.2. Vite

Nowoczesne narzędzie do budowania aplikacji internetowych, które łączy etap programowania z produkcją, oferując szybki i wygodny proces tworzenia. Składa się z serwera deweloperskiego, który zapewnia błyskawiczny start projektu oraz funkcje, takie jak szybkie Hot Module Replacement (HMR), przyspieszające testowanie i rozwój aplikacji. Druga kluczowa funkcja to mechanizm komplikacji, który wykorzystuje Rollup do generowania zoptymalizowanych statycznych zasobów gotowych do wdrożenia. Vite dostarcza domyślne ustawienia gotowe do użycia, ale dzięki systemowi wtyczek i elastycznej konfiguracji można go łatwo dostosować do specyficznych potrzeb projektu. Dzięki tym cechom Vite idealnie nadaje się do nowoczesnych aplikacji, gdzie szybkość i wydajność są kluczowe.

3.3.3. React-Router

Biblioteka dla React, która pozwala na tworzenie tras w aplikacjach internetowych. Umożliwia definiowanie i zarządzanie różnymi widokami oraz nawigacją między nimi, co jest kluczowe w tworzeniu aplikacji jednostronnicowych (SPA). React Router działa jak most między interfejsem użytkownika a logiką nawigacji, pozwalając na dynamiczne zmianianie zawartości strony bez konieczności jej przeładowywania.

Biblioteka oferuje elastyczne i deklaratywne API, które pozwala łatwo definiować trasy, parametry w adresach URL, a także korzystać z zaawansowanych funkcji, takich jak nawigacja z ochroną dostępu, przekierowania czy ładowanie asynchroniczne komponentów. Dzięki React Router programiści mogą tworzyć spójne i intuicyjne doświadczenia użytkownika w swoich aplikacjach React.

3.3.4. Typescript

[3.2.2]

3.3.5. tailwind

[3.2.10]

3.3.6. React-toastify

Biblioteka dla React, która ułatwia dodawanie i zarządzanie powiadomieniami w aplikacjach internetowych. Dzięki prostemu i elastycznemu API pozwala na szybkie tworzenie powiadomień o różnym wyglądzie i zachowaniu. Obsługuje zarówno komunikaty informacyjne, sukcesy, ostrzeżenia, jak i błędy, zapewniając ich automatyczne zamknięcie lub możliwość ręcznego zamknięcia.

Wyróżnia się możliwością dostosowania stylów, animacji oraz pozycji wyświetlanego powiadomienia, a także obsługą systemów tematycznych (np. jasnego i ciemnego trybu). Dzięki tej bibliotece dodanie powiadomień do aplikacji React staje się intuicyjne i szybkie.

3.3.7. Socket.IO

Biblioteka, która umożliwia dwukierunkową, opartą na zdarzeniach komunikację w czasie rzeczywistym między klientem a serwerem. Jest szeroko stosowana w aplikacjach wymagających dynamicznej wymiany danych, takich jak czaty, powiadomienia w czasie rzeczywistym, gry wieloosobowe czy śledzenie aktywności użytkowników.

Zapewnia niezawodne połączenie, wykorzystując WebSocket jako podstawowy protokół, ale automatycznie przełącza się na inne mechanizmy transportu (np. długie odpytywanie HTTP), jeśli WebSocket nie jest obsługiwany. Oferuje intuicyjne API do obsługi zdarzeń po obu stronach połączenia, a także wbudowane funkcje, takie jak obsługa przestrzeni nazw, pokoi i nadawania wiadomości do wielu klientów jednocześnie. Dzięki temu tworzenie aplikacji w czasie rzeczywistym staje się prostsze i bardziej efektywne.

3.3.8. Lodash

Biblioteka narzędziowa dla JavaScript, oferująca funkcje do manipulacji danymi, pracę z kolekcjami, funkcjami i strukturami danych. Jest zaprojektowana z myślą o wydajności i elastyczności, zapewniając szeroki wachlarz funkcji, takich jak manipulacja tablicami, łańcuchami tekstowymi, obiektami i danymi liczbowymi. Lodash jest modułowa, co oznacza, że użytkownicy mogą wybierać tylko te funkcje, których potrzebują, co pozwala na zmniejszenie rozmiaru aplikacji przy zachowaniu wysokiej wydajności.

3.3.9. REST API (Representational State Transfer)

Sposób komunikacji pomiędzy aplikacjami, który opiera się na zasadach architektury REST. Umożliwia one wymianę danych pomiędzy serwerem a klientem za pomocą standardowych protokołów internetowych, takich jak HTTP. REST API definiuje zestaw metod, takich jak GET, POST, PUT i DELETE, które umożliwiają operacje na zasobach, takich jak pobieranie, tworzenie, modyfikowanie lub usuwanie danych. Dzięki temu zapewnia prostą, skalowaną i niezależną od platformy metodę integracji aplikacji z różnymi systemami i usługami.

3.3.10. Express

Szybki, nieopinionowany, minimalistyczny framework webowy dla Node.js, który zapewnia solidny zestaw funkcji do tworzenia aplikacji webowych i mobilnych. Umożliwia łatwe tworzenie interfejsów API, middleware oraz aplikacji internetowych, oferując prostą, ale potężną strukturę do zarządzania żądaniami HTTP, routowaniem, sesjami i innymi zadaniami backendowymi.

Dzięki Express, programiści mogą skupić się na logice aplikacji, zamiast na złożoności samego frameworku. Framework jest szeroko używany ze względu na swoją elastyczność i łatwość integracji z innymi narzędziami oraz bibliotekami. Dokumentacja Express oraz tłumaczenia społeczności ułatwiają naukę i szybkie wdrażanie.

3.3.11. JSON Web Tokens (JWT)

Otwarty, standardowy format określony w dokumencie RFC 7519, używany do bezpiecznego przekazywania danych (claims) pomiędzy dwiema stronami. Tokeny JWT są łatwe do przetworzenia, zarówno przez serwery jak i klientów, co czyni je popularnym narzędziem w implementacji uwierzytelniania w aplikacjach internetowych.

Tokeny te składają się z trzech części: nagłówka, danych (payload) i podpisu. Nagłówek zawiera metadane dotyczące algorytmu kryptograficznego używanego do podpisania tokenu, payload zawiera dane (informacje o użytkowniku, identyfikator sesji, zakres dostępu itp.), a podpis zapewnia integralność i autentyczność tokenu. Dzięki swojemu uniwersalnemu formatowi, JWT umożliwiają łatwe uwierzytelnianie i autoryzację, zarówno w środowisku serwera, jak i po stronie klienta.

3.3.12. otpauth

[3.2.12]

3.3.13. node-qrcode

Biblioteka w JavaScript, która umożliwia łatwe generowanie kodów QR w aplikacjach Node.js. Dzięki niej można szybko tworzyć obrazki kodów QR, które mogą być używane do kodowania różnorodnych danych, takich jak tekst, URL, informacje kontaktowe, adresy e-mail czy dane płatnicze.

3.3.14. SQLite

To lekka, samodzielna baza danych, która jest składowana w pliku. Jest popularna w aplikacjach mobilnych, takich jak systemy operacyjne Android i iOS, oraz w aplikacjach webowych, ze względu na swoją prostotę, wydajność i niskie wymagania systemowe.

3.3.15. node-sqlite3

Asynchroniczne, nieblokujące wiązanie bazy danych SQLite3 dla Node.js. Umożliwia programistom łatwe tworzenie, odczytywanie, aktualizowanie i usuwanie danych w bazach SQLite3 bezpośrednio z poziomu aplikacji Node.js.

3.3.16. sequelize

To ORM (Object-Relational Mapping) dla TypeScript i Node.js, który wspiera różne systemy zarządzania bazami danych, takie jak Oracle, Postgres, MySQL, MariaDB, SQLite i SQL Server. Oferuje solidną obsługę transakcji, relacji, chętne i leniwe ładowanie, replikację odczytu oraz wiele innych funkcji, które ułatwiają interakcję z bazami danych.

Sequelize automatyzuje procesy związane z mapowaniem obiektów na tabele w bazie danych, co pozwala programistom na skupienie się na logice aplikacji, zamiast zarządzania zapytaniami SQL. Dzięki temu, że działa z różnych rodzajów baz danych, jest niezwykle elastyczny i szeroko stosowany w aplikacjach webowych i mobilnych.

3.3.17. bcrypt

Popularna biblioteka do hashowania haseł, która zapewnia bezpieczne i wydajne szyfrowanie przy użyciu algorytmu bcrypt. Jest szeroko stosowana w aplikacjach webowych do ochrony danych użytkowników, poprzez przechowywanie haseł w postaci zaszyfrowanych wartości, które są trudne do odgadnięcia.

bcrypt stosuje iteracyjną metodę hashowania, która zwiększa bezpieczeństwo hasła poprzez zwiększenie liczby operacji rozdzielających przy każdym hashowaniu. Dzięki temu jest odporny na ataki typu brute-force i inne techniki odgadnięcia haseł.

3.3.18. Postman

Popularne narzędzie ułatwiające pracę z API. Umożliwia wysyłanie żądań HTTP i analizowanie odpowiedzi, co pozwala programistom testować, debugować i dokumentować interfejsy API w prosty i efektywny sposób. Obsługuje różne metody HTTP, takie jak GET, POST, PUT, DELETE, a także umożliwia dodawanie nagłówków, parametrów, autoryzacji oraz przesyłanie danych w formatach JSON, XML i innych.

3.4. Wykorzystane technologie - Serwer przekaźnikowy

3.4.1. Rust

Język programowania, który wyróżnia się niezwykłą szybkością i efektywnością pamięciową. Dzięki temu może zasilać usługi o krytycznym znaczeniu dla wydajności, działać na urządzeniach wbudowanych i łatwo integrować się z innymi językami programowania.

Oferuje bogaty system typów i model własności, które gwarantują bezpieczeństwo pamięci i wątków. Dzięki temu możliwe jest eliminowanie wielu klas błędów na etapie kompilacji, co sprawia, że aplikacje są bardziej niezawodne i bezpieczne.

Charakteryzuje się doskonałą dokumentacją, przyjaznym kompilatorem z użytecznymi komunikatami o błędach oraz najwyższej klasy narzędziami. Posiada zintegrowany menedżer pakietów i narzędzie do kompilacji, inteligentne wsparcie dla wielu edytorów, takie jak automatyczne uzupełnianie i inspekcja typów, automatyczne formatowanie i inne funkcje, które wspomagają codzienną pracę programisty.

3.4.2. Axum

Framework aplikacji internetowych, który wyróżnia się na tle innych dzięki swojej ergonomii i modułowości. Główna różnica polega na tym, że Axum nie posiada własnego systemu middleware, lecz korzysta z usług (Service) biblioteki tower. Dzięki temu, framework oferuje wszystkie podstawowe funkcje, takie jak timeouts, śledzenie, kompresję, autoryzację, automatycznie, bez potrzeby ręcznego konfigurowania.

3.4.3. Tokio

Asynchroniczne środowisko uruchomieniowe dla języka programowania Rust. Zapewnia niezbędne bloki konstrukcyjne do tworzenia aplikacji sieciowych, oferując elastyczność w obsłudze różnorodnych systemów, od dużych serwerów z dziesiątkami rdzeni po małe urządzenia wbudowane. Dzięki swojej asynchroniczności, Tokio

umożliwia wydajne przetwarzanie równoległe, co sprawia, że jest idealnym wyborem do tworzenia aplikacji sieciowych, które muszą radzić sobie z wysokimi obciążeniami i wymagającymi operacjami I/O.

3.4.4. Serde

Framework do serializacji i deserializacji struktur danych w języku Rust, umożliwiający ich efektywne przekształcanie między formatami. Ekosystem Serde obejmuje zarówno struktury danych, które są zdolne do serializacji i deserializacji siebie, jak i różne formaty danych, które potrafią przekształcać różne obiekty. Serde pełni rolę warstwy pośredniczącej, która umożliwia płynne współdziałanie tych dwóch grup, pozwalając na łatwe serializowanie i deserializowanie dowolnej obsługiwanej struktury danych przy użyciu różnych formatów danych, takich jak JSON, CBOR, czy TOML.

3.4.5. Postman

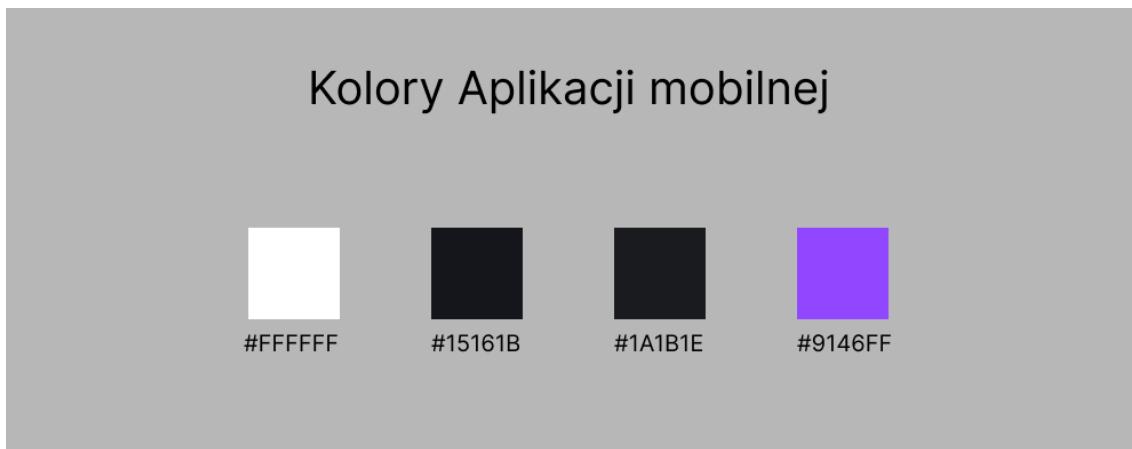
[3.3.18]

3.5. UI Aplikacji

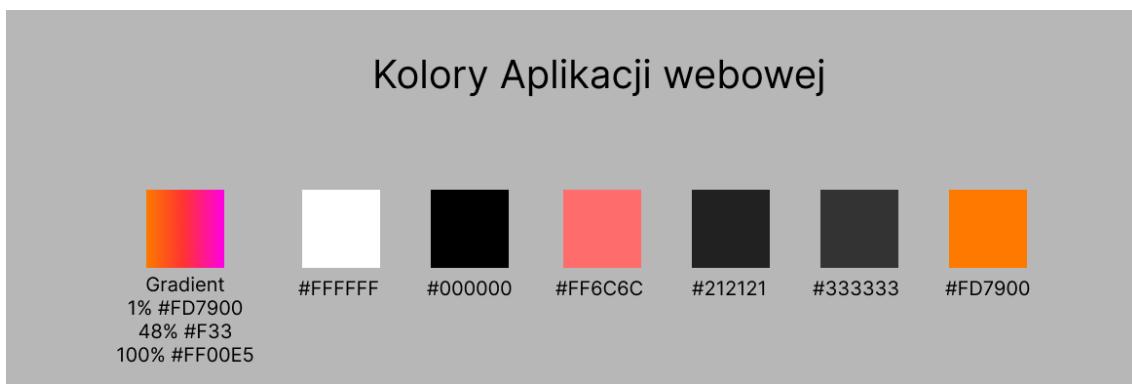
Projektowanie interfejsu użytkownika aplikacji opiera się na trendzie minimalistycznym, co odzwierciedla prostota zarówno aplikacji mobilnej, jak i webowej. Jedynym odstępstwem od tej zasady jest aplikacja mobilna, która wyróżnia się funkcjonalnościami specyficznymi dla procesu weryfikacji dwuetapowej. Zawiera ona szczegółowe instrukcje dotyczące zapisu i odczytu danych do pliku oraz unikalny interfejs skanowania kodów QR. W tej sekcji użytkownik ma do dyspozycji dwa różne przyciski: jeden do dodawania sekretów do weryfikacji, drugi do skanowania kodu QR i przesyłania wygenerowanego kodu do serwera przekaźnikowego.

3.5.1. Paleta kolorów

Z uwagi na preferencje użytkowników i komfort korzystania z aplikacji, zarówno wersja webowa, jak i mobilna korzystają z ciemnego motywów. Główną różnicą jest dominujący kolor.



Rysunek 2: Paleta barw aplikacji mobilnej - Opracowanie własne



Rysunek 3: Paleta barw aplikacji webowej - Opracowanie własne

3.5.2. Dostarczanie informacji

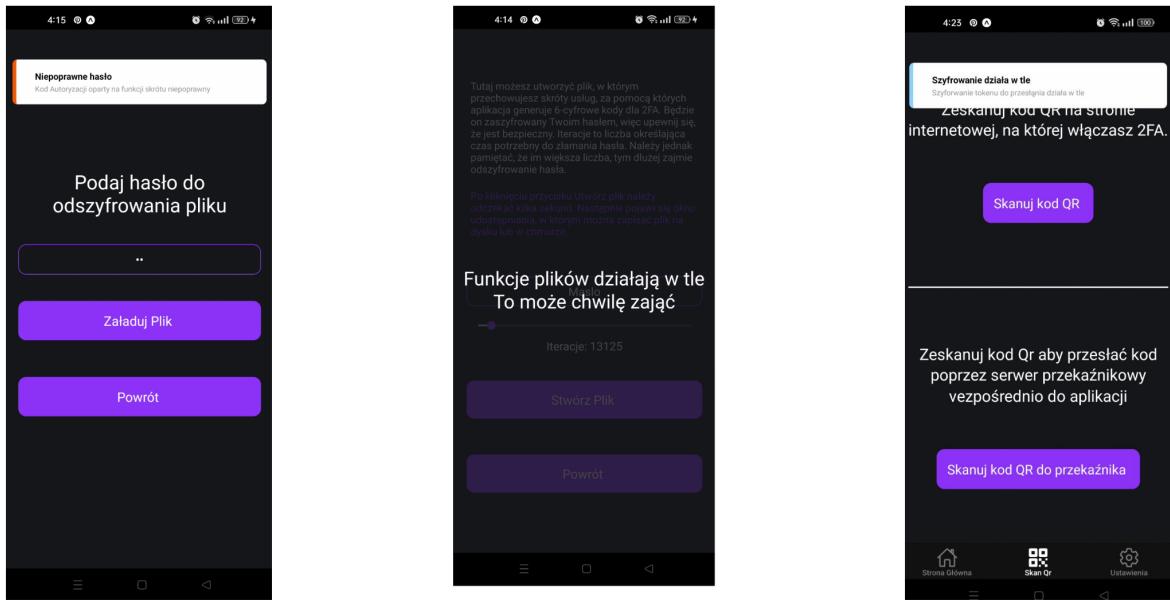
W obu wersjach aplikacji zadbane o wyraźne i intuicyjne dostarczanie informacji zwrotnej w przypadku wystąpienia błędów lub trwających procesów.

- Aplikacja webowa: Używa biblioteki react-toastify do wyświetlania komunikatów o błędach, takich jak niepoprawne dane logowania czy kod weryfikacyjny.
- Aplikacja mobilna: Korzysta z biblioteki react-native-toast-message, która działa zarówno na systemie Android, jak i iOS.

Dodatkowo, podczas procesów takich jak deszyfrowanie pliku, w aplikacji mobilnej wyświetlana jest pełnoekranowa nakładka, która informuje użytkownika o trwających operacjach. Ma to na celu zapobieganie przypadkowemu klikaniu przycisków, co mogłoby zakłócić działanie aplikacji.

W obu wersjach komunikaty wyróżniają się jasnymi kolorami, kontrastującymi z ciemnym motywem, co zwiększa ich widoczność i ułatwia odbiór.

Przykłady komunikaty aplikacji mobilnej



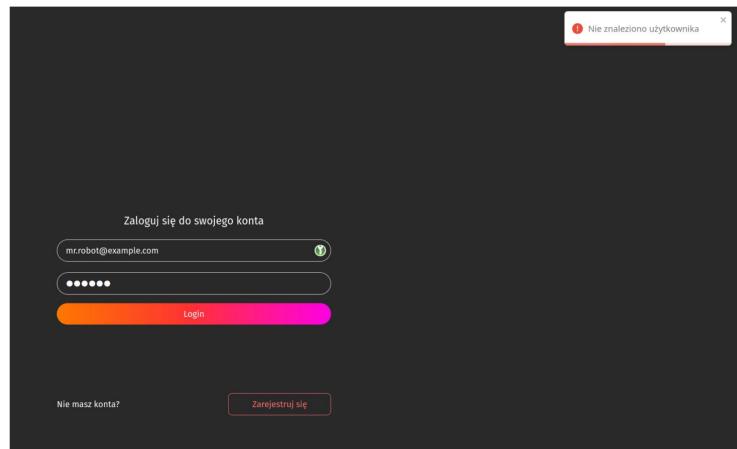
Komunikat błędu

Komunikat o
szyfrowaniu pliku
w tle

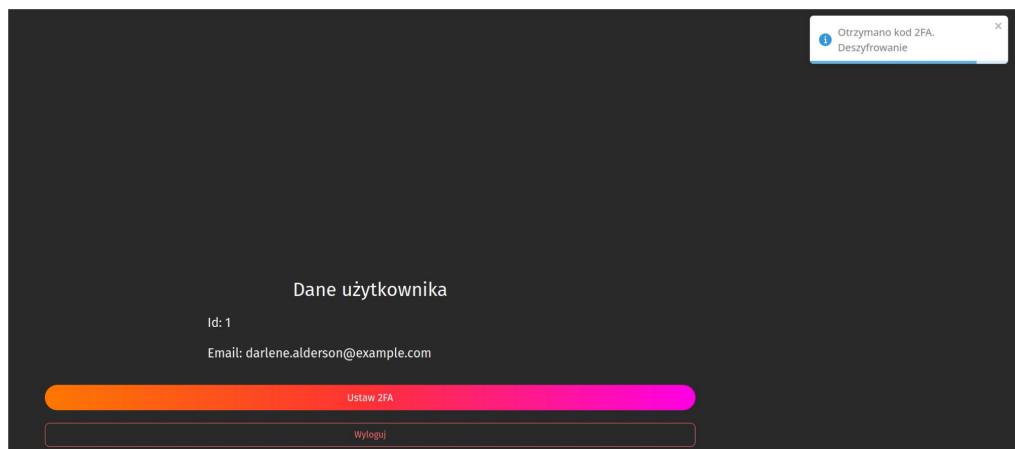
Komunikat o
szyfrowaniu kodu

Rysunek 4: Przykłady komunikatów aplikacji mobilnej - Opracowanie własne

Przykłady komunikatów aplikacji webowej



komunikat błędu



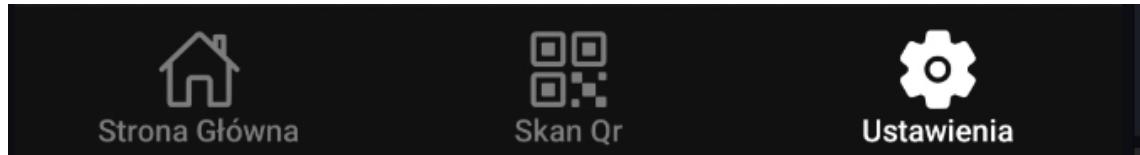
komunikat o deszyfrowaniu kodu z serwera przekaźnikowego

Rysunek 5: Przykłady komunikatów aplikacji mobilnej - Opracowanie własne

3.5.3. Nawigacja w aplikacji mobilnej

Nawigacja w aplikacji mobilnej składa się z dwóch głównych widoków: widoku głównego oraz widoku zakładkowego.

1. **Widok główny** gdzie dostępne są dwa przyciski nawigacyjne:
 - **Ładowanie pliku** - Przenosi użytkownika do zakładki odpowiedzialnej za ładowanie pliku.
 - **Tworzenie pliku** - Otwiera zakładkę do generowania nowego pliku.
2. **Widok zakładkowy** gdzie nawigacja odbywa się za pomocą dolnego paska nawigacyjnego, który zawiera trzy zakładki:
 - **Strona Główna** - Wyświetla generowane hasła jednorazowe które można usunąć
 - **Skan QR** - Tu występują dwa przyciski które służą kolejno pierwszy do dodawania sekretów, z których generowane są hasła, a drugi do skanowania kodu QR w celu przesyłania wygenerowanego kodu jednorazowego przez serwer przekaźnikowy.
 - **Ustawienia** - Umożliwia zmianę języka aplikacji oraz zamknięcie załączanego pliku.



Rysunek 6: Panel nawigacyjny aplikacji mobilnej - Opracowanie własne

3.5.4. Nawigacja w aplikacji webowej

Nawigacja w aplikacji webowej opiera się na podstronach, które umożliwiają dostęp do różnych funkcji:

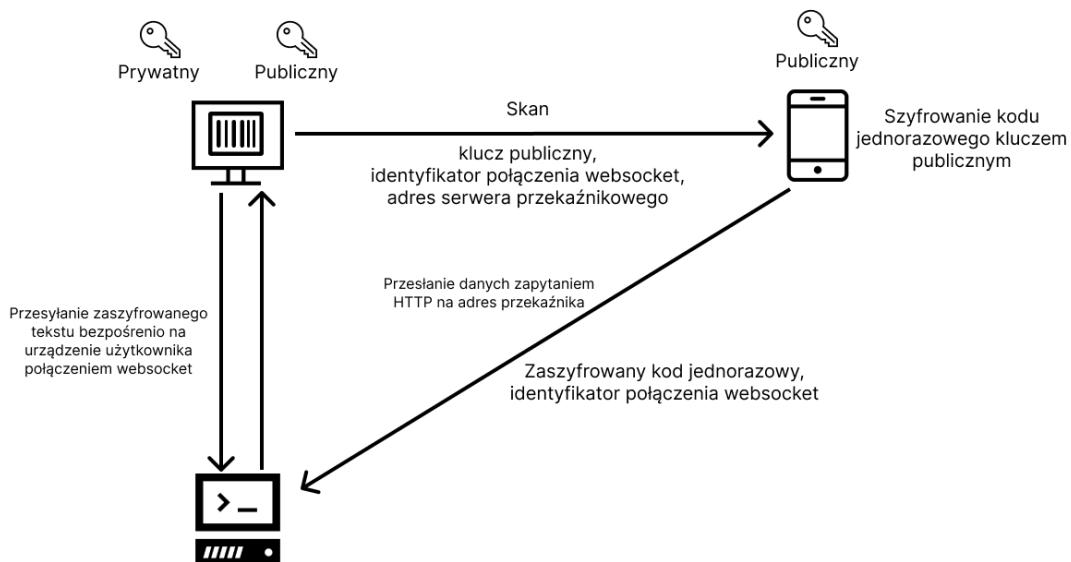
- **/register:** Użytkownik może tu stworzyć nowe konto.
- **/userdata:** Tu wyświetlane są podstawowe informacje zalogowanego użytkownika. Można też tu włączyć weryfikację dwuetapową.
- **/login:** Tu można się zalogować oraz potwierdzić tożsamość za pomocą hasła jednorazowego, jeśli aktywowano uwierzytelnianie dwuetapowe.

4. Implementacja

4.1. Wstęp

Rozdział ten przedstawia szczegółowy opis konstrukcji oraz działania aplikacji. W pierwszej części omówione zostaną kluczowe systemy, takie jak system przekaźników, mechanizmy odczytu i zapisu zaszyfrowanych plików oraz sposób odczytu danych z kodów QR przy rejestrowaniu nowego sekretu do konta. Następnie zaprezentowana zostanie ogólna architektura aplikacji, wraz z wyjaśnieniem sposobu ich działania, testów oraz zasad korzystania z poszczególnych funkcji.

4.2. Przekazywanie kodów systemem serwerów przekaźnikowych



Rysunek 7: Zobrazowanie systemu serwerów przekaźnikowych - Opracowanie Własne

System ten umożliwia przekazywanie kodów jednorazowych dwuetapowej weryfikacji bez konieczności ręcznego ich wpisywania. Użytkownik skanuje kod QR za pomocą aplikacji, który zawiera niezbędne dane do przekazania kodu z telefonu do aplikacji użytkownika.

4.2.1. Proces weryfikacji dwuetapowej z pomocą systemu serwerów przekaźnikowych

1. Generowanie kluczy i kodu QR

Przy próbie zalogowania serwer przekazuje informację, że wymagany jest dodatkowo kod jednorazowy do weryfikacji dwuetapowej. Podaje też odpowiednie dane niezbędne do wygenerowania kodu dla danego konta (wystawca oraz etykieta). Jeśli aplikacja webowa jest połączona z serwerem przekaźników, generuje ona parę kluczy RSA-OAEP i konwertuje je do formatu PEM (by klucz publiczny mógł być dodany do kodu QR). Do kodu QR dodawane są: adres URL serwera przekaźnikowego, identyfikator połączenia z serwerem przekaźnika (połączenie Socket.io do wymiany danych w czasie rzeczywistym), wystawcę z etykietą sekretu konta oraz klucz publiczny, umożliwiający szyfrowanie jednorazowego kodu.

2. Skanowanie kodu QR

Aplikacja mobilna po zeskanowaniu danych z kodu QR w pierwszej kolejności sprawdza integralność danych które zeskanowała. Jeśli wszystko się zgadza aplikacja przeszukuje dane z pamięci RAM wyciągając odpowiedni sekret użytkownika. Używając tego sekretu i bieżącego czasu, generowany jest jednorazowy kod, który następnie jest szyfrowany kluczem publicznym.

3. Serwer przekaźnika

Zaszyfrowany kod wraz z id połączenia jest przekazywany do serwera przekaźnikowego który następnie z pomocą tego id przesyła kod bezpośrednio do aplikacji użytkownika.

4. Deszyfrowanie kodu jednorazowego

Aplikacja użytkownika otrzymuje zaszyfrowany kod i natychmiast go odszyfrowuje kluczem prywatnym. Jeśli wszystko się zgadza, używa odszyfrowany kod do potwierdzenia tożsamości z serwerem. Serwer zwraca dane potrzebne do logowania, umożliwiając użytkownikowi zalogowanie się.

4.3. System zapisywania danych do szyfrowanego pliku

System zapisu danych został zaprojektowany z naciskiem na bezpieczeństwo, korzystając z szyfrowania za pomocą algorytmu AES. Proces rozpoczyna się od przekształcenia hasła wejściowego za pomocą algorytmu PBKDF2, który wzmacnia je,

stosując funkcję pseudolosową, taką jak HMAC. Proces ten powtarza się wielokrotnie, aby wygenerować klucz pochodny. Dzięki temu zmniejszona zostaje podatność na ataki typu brute-force, zapewniając wyższy poziom ochrony danych.

Dane przechowywane w pliku przed ich zaszyfrowaniem zapisane są w formacie TOML, który zapewnia czytelność i łatwość zarządzania strukturą danych. W głównym bloku znajduje się tablica `platformServices`, zawierająca informacje o zarejestrowanych sekretach. Każda usługa przechowuje dane takie jak wystawca (issuer), klucz tajny (secret), parametry OTP (czas życia kodu, długość, typ algorytmu), a także etykietę identyfikacyjną.

Listing 2: ”Przykład danych pliku przed szyfrowaniem”

```
[[ platformServices ]]
issuer = "PrzykładowyWystawca"
secret = "M5LHQUSVA4JCEARN"
period = 30
digits = 6
algorithm = "SHA1"
otpType = "totp"
label = "PrzykładowyWystawca:darlene@example.com"

[[ platformServices ]]
issuer = "PrzykładowyWystawca"
secret = "G5IE4IZEPQBUYPYY"
period = 30
digits = 6
algorithm = "SHA1"
otpType = "totp"
label = "PrzykładowyWystawca:elliot@example.com"
```

Listing 3: ”Przykład danych pliku po zaszyfrowaniu”

```
13125$ZgswZrP/AH/DfXf+LsZ5mA==$PlUrnK4pDamwxv7Cu4xWBw==$8wt4Tn/
YTLDh6L2pdY/j5DkbpUzDsbT5PQT3JyXAWKg==$nG0qr9GX12fq3xVGTe3Vw6Fs8
zB7gNayuVPEoy+Gv06PBStdjR0Y0hgzAvOKjeMDysa22JzLlBzRn+bRESuv4Psj
UM9wWub+3XVAqnhB5nQTMnJrZz2sNzyk58cjvkFKQwOt/fo4jXCjqmY78lixlk8
xJdfFt1a6mUuj3sHng6VbG9iQjWVmkcFSiu8TO8J0goGiZzGXqSbZ0lK0zBE41K
MbWK8AqhmC2lesDRgZXAuoMTnewaO7QmBmVRaXzh9n4rfVHfDg8EihYEZ5YfB+q
WfNtz8ZIf9bE6j2ylygWlo0tzU/iVb5LwDAQevb9k+juT9uEGDClx3C087h6kmO
```

GGWtkErutqlnDed+skCwtjRmd9jGt1BgU5dyNfbhtB7KR13g5n48rQcPUQ2MPBdT
Q2vpZxk2mq2uDONIQLb6o+6SMIXOgDIAeP2boVg80V1XCew4NlOQhpDIBszXCcoN
lnpmeshid6ysFbdMkYOWUdpJvcYtuyEaWIHH4NPx6/a1

Struktura pliku zaszyfrowanego zawiera na początku dane niezbędne do poprawnego odszyfrowania danych oddzielone znakiem \$:

- iterations - liczba iteracji algorytmu PBKDF2,
- salt - sól używana w procesie generowania klucza,
- iv - wektor inicjalizujący (Initialization Vector) dla algorytmu AES,
- HMAC - podpis uwierzytelniający integralność danych.
- szyfr - zaszyfrowany text z sekretami pojedyńczych usług.

4.4. Odczytywanie danych z kodu QR przy rejestracji nowego konta

W celu włączenia weryfikacji dwuetapowej serwer danej aplikacji generuje sekret, który służy do tworzenia jednorazowych haseł. Sekret, wraz z dodatkowymi parametrami konfiguracyjnymi, jest umieszczany w kodzie QR w formacie zgodnym z protokołem otpauth://.

Listing 4: "Struktura adresu URL protokołu otpauth"

```
otpauth:///[typ OTP]/[etykieta]?secret=[sekret]&  
period=[odst p czasowy]&digits=[cyfry]&algorithm=[algorytm]&  
issuer=[wystawca]
```

Gdzie poszczególne pola oznaczają:

- **typ OTP** – wskazuje, czy generowane hasło jednorazowe jest oparte na czasie (TOTP) czy na liczniku (HOTP).
- **etykieta** – opis identyfikujący konto (często w formacie [wystawca] : [email użytkownika]).
- **sekret** – klucz tajny, który jest podstawą generowania haseł jednorazowych.
- **odstęp czasowy (period)** – długość interwału czasowego w sekundach, dla którego hasło jednorazowe jest ważne (np. 30 sekund dla (TOTP)).

- **cyfry (digits)** – liczba cyfr w generowanym haśle (najczęściej 6).
- **algorytm (algorithm)** – algorytm kryptograficzny używany do generowania haseł jednorazowych (domyślnie **SHA1**).
- **wystawca (issuer)** – nazwa aplikacji lub organizacji generującej kod.

Listing 5: "Przykładowy tekst sekretu zakodowany w kodzie QR"

```
otpauth://totp/PrzykladowyWystawca:elliot%40example.com?
secret=CF2AGE3CLAJSOXTZ&period=30&digits=6&algorithm=SHA1&
issuer=PrzykladowyWystawca
```

Aplikacja po zeskanowaniu takiego kodu QR wyciąga wymienione dane, które są następnie zapisywane w strukturze danych aplikacji oraz przechowywane w pamięci (RAM) na czas bieżącego użycia. Zebrane informacje pozwalają na poprawne generowanie haseł jednorazowych za pomocą algorytmu TOTP (Time-Based One-Time Password) lub HOTP (HMAC-Based One-Time Password), w zależności od podanego typu OTP.

4.5. Serwer przekaźnikowy

To aplikacja obsługująca dwie główne usługi: Socket.IO oraz HTTP.

Usługa Socket.IO umożliwia wymianę danych w czasie rzeczywistym z aplikacjami. Dzięki niej aplikacja webowa może otrzymać zaszyfrowane hasło jednorazowe od aplikacji mobilnej. Każda aplikacja, która inicjuje połączenie w usłudze Socket.IO, dołącza do prywatnego pokoju. Tylko ta aplikacja otrzymuje wiadomości wysyłane przez serwer do tego pokoju, co zapewnia pełną izolację i bezpieczeństwo danych. Aplikacja webowa, po nawiązaniu połączenia, otrzymuje identyfikator połączenia (connection ID), który serwer wykorzystuje do przesyłania wiadomości bezpośrednio do odpowiedniego pokoju.

Usługa HTTP obsługuje jedną ścieżkę **/sendCode**. Otrzymuje ona dwa kluczowe parametry:

- zaszyfrowane hasło jednorazowe,
- identyfikator połączenia Socket.IO (connection ID).

Gdy serwer odbierze zapytanie HTTP na ścieżce **/sendCode**, przesyła zaszyfrowane hasło jednorazowe do pokoju Socket.IO zidentyfikowanego przez otrzymany

identyfikator połączenia. W ten sposób zapewniona jest szybka i bezpieczna komunikacja pomiędzy aplikacjami.

4.6. Testowanie aplikacji

Proces testowania aplikacji został przeprowadzony równolegle z jej implementacją, co umożliwiło szybkie wykrywanie i eliminowanie potencjalnych błędów. Ze względu na prostotę struktury aplikacji webowej oraz mobilnej, testy były wykonywane manualnie, w trakcie tworzenia poszczególnych funkcji.

4.6.1. Aplikacja webowa

Tworzenie aplikacji webowej rozpoczęło się od implementacji podstawowych funkcjonalności, takich jak system logowania i rejestracji. Ze względu na prostotę obsługi aplikacji webowej, nie zostały wdrożone zautomatyzowane testy, a proces testowania polegał na ręcznym sprawdzaniu poprawności działania każdej nowo dodanej funkcjonalności programem Postman.

Po zakończeniu prac nad systemem logowania i rejestracji, wdrożono system weryfikacji dwuetapowej. Testy tego systemu zostały przeprowadzone z wykorzystaniem zewnętrznych aplikacji mobilnych obsługujących weryfikację dwuetapową. Sprawdzano, czy generowane kody są poprawne i czy proces uwierzytelniania przebiega zgodnie z założeniami. W przypadku tego etapu testów wyniki były jednoznaczne: system działał poprawnie lub nie działał wcale.

4.6.2. Aplikacja mobilna

Testowanie aplikacji mobilnej odbywało się równolegle z implementacją jej funkcjonalności. Prace rozpoczęto od systemu szyfrowania plików, który został przetestowany manualnie na urządzeniach mobilnych. Po zakończeniu tej części wdrożono funkcję generowania kodów dwuetapowych.

Funkcjonalność aplikacji mobilnej była testowana zarówno na gotowej aplikacji webowej, jak i na innych platformach obsługujących weryfikację dwuetapową. Testy te miały na celu potwierdzenie kompatybilności generowanych kodów z różnymi systemami weryfikacji. Wykorzystanie wielu platform umożliwiło upewnienie się, że rozwiązanie działa poprawnie niezależnie od środowiska docelowego.

4.6.3. System przekaźnikowy

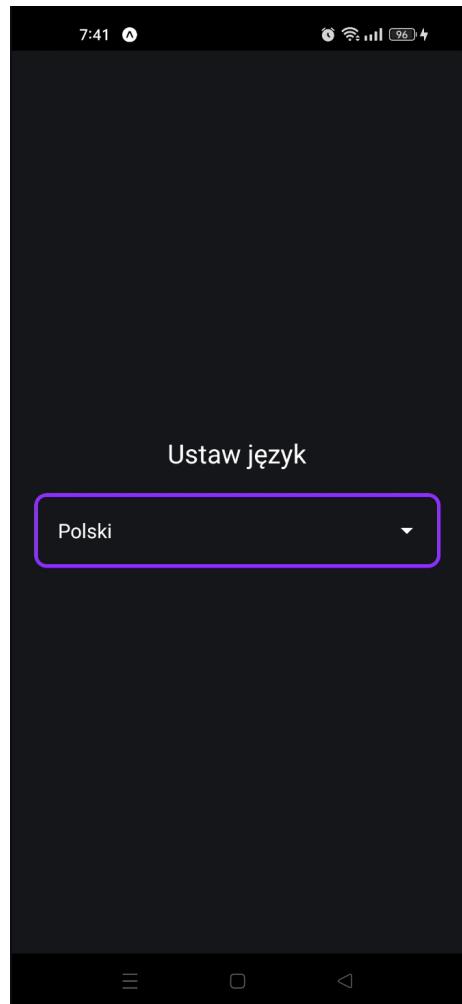
Serwer przekaźnikowy to lekka aplikacja, dzięki czemu testy mogły być wykonywane ręcznie przy użyciu programu Postman.

System był implementowany równolegle z aplikacją webową i mobilną. Funkcjonowanie tego systemu opiera się na szyfrowaniu z użyciem przekazywanego klucza, co wymuszało natychmiastowe testowanie każdej nowo dodanej funkcji. Proces testowania polegał na sprawdzaniu poprawności komunikacji pomiędzy aplikacjami oraz skuteczności przesyłania danych za pomocą systemu przekaźnikowego. W tym przypadku wyniki testów również były zero-jedynkowe: system albo działał, albo nie.

4.7. Instrukcja użytkowania aplikacji mobilnej

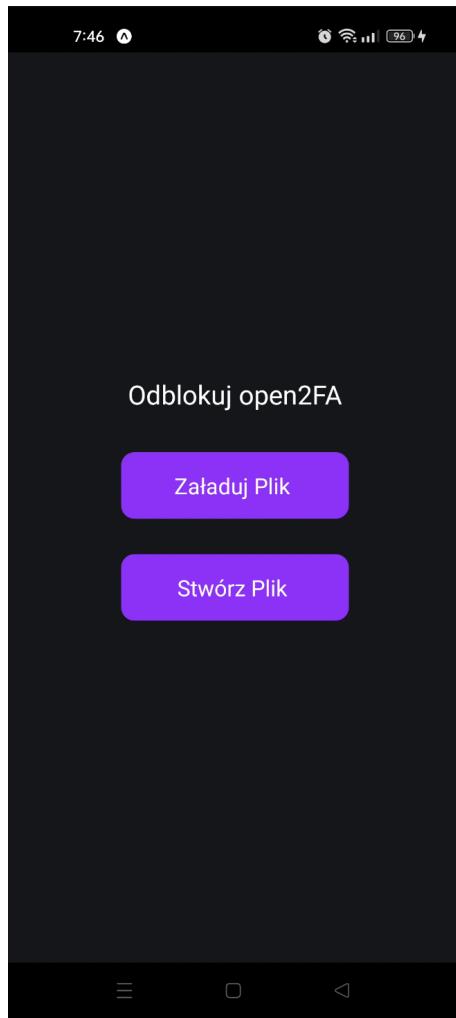
4.7.1. Inicjalizacja aplikacji - Tworzenie oraz załadowywianie pliku

Przy pierwszym uruchomieniu aplikacji użytkownik zobaczy ekran wyboru języka przedstawiony w formie listy. Aplikacja w obecnym stanie obsługuje język polski oraz angielski:



Rysunek 8: Wybór języka przy pierwszym otwarciu aplikacji - Opracowanie Własne

Wybrany język zostaje zapisany w pamięci telefonu, aby aplikacja mogła automatycznie go ustawić podczas kolejnych uruchomień. Po dokonaniu wyboru języka użytkownik zostaje przeniesiony do głównego menu aplikacji, które zawiera dwa przyciski: „Stwórz plik” oraz „Załaduj plik”.

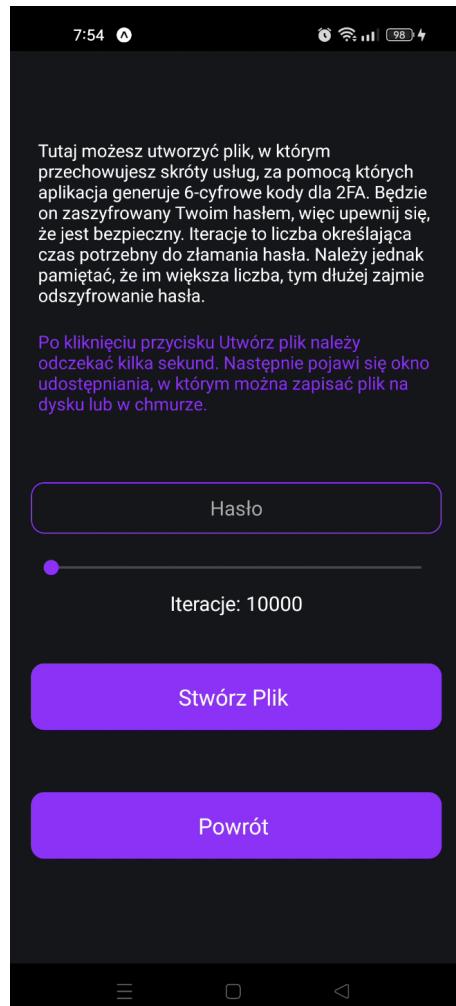


Rysunek 9: Główne menu aplikacji mobilnej - Opracowanie Własne

Na początku nie istnieje żaden plik, który mógłby zostać załadowany, dlatego konieczne jest jego utworzenie. Po kliknięciu przycisku „Stwórz plik” użytkownik zostaje przeniesiony do menu tworzenia pliku. Znajduje się tam krótka instrukcja wyjaśniająca proces tworzenia pliku oraz formularz umożliwiający jego konfigurację. Formularz wymaga:

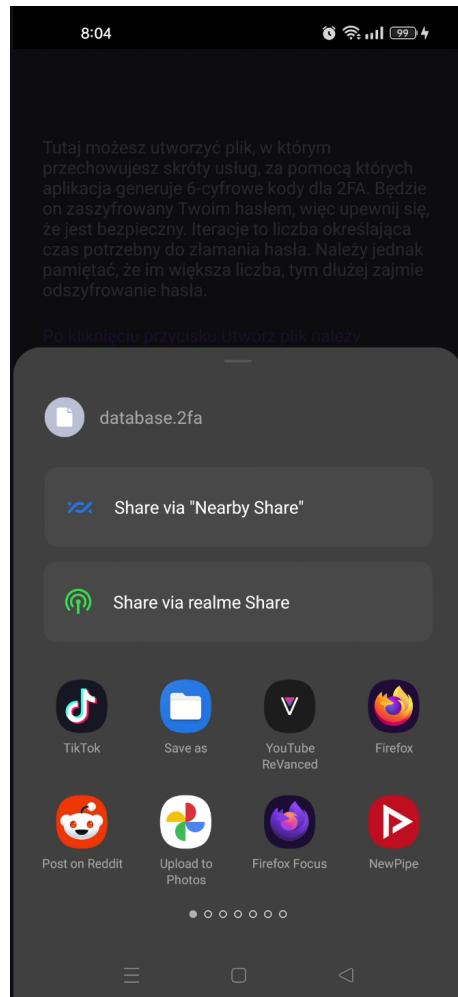
- Wpisania bezpiecznego hasła, które będzie używane do deszyfrowania pliku,
- Ustawienia liczby iteracji algorytmu PBKDF2 za pomocą suwaka, co pozwala dostosować czas generowania skrótu hasła.

Szczegóły dotyczące tworzenia plików zostały opisane w rozdziale [4.3].



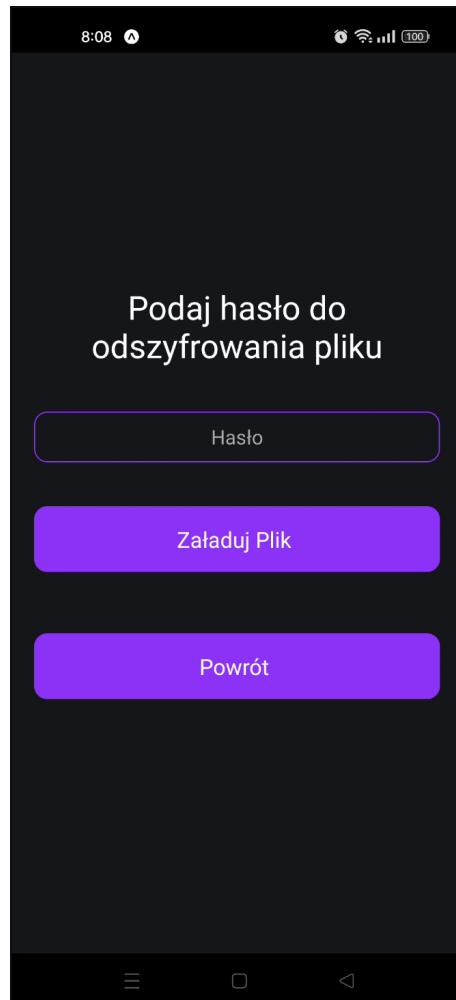
Rysunek 10: Menu tworzenia pliku aplikacji - Opracowanie Własne

Po kliknięciu przycisku „Stwórz plik” aplikacja uruchomi proces szyfrowania, wyświetlając nakładkę blokującą ekran, która informuje użytkownika o trwającym procesie. Nakładka została przedstawiona na Rysunku 4. Po zakończeniu szyfrowania użytkownik zobaczy okienko udostępniania, w którym opcja „Save as” pozwala zapisać plik w pamięci telefonu.



Rysunek 11: okienko udostępnienia pliku - Opracowanie Własne

Następnie zostaniemy przeniesieniu z powrotem do menu głównego gdzie opcja „Załaduj Plik” jest teraz możliwa do wykorzystania. Po jej wybraniu aplikacja otworzy okno wyboru pliku. Jeśli użytkownik wcześniej nie nadał aplikacji dostępu do pamięci urządzenia, zostanie poproszony o odpowiednie uprawnienia. Następnie użytkownik wybiera plik, który chce załadować, i zostaje przeniesiony do ekranu odszyfrowywania pliku.

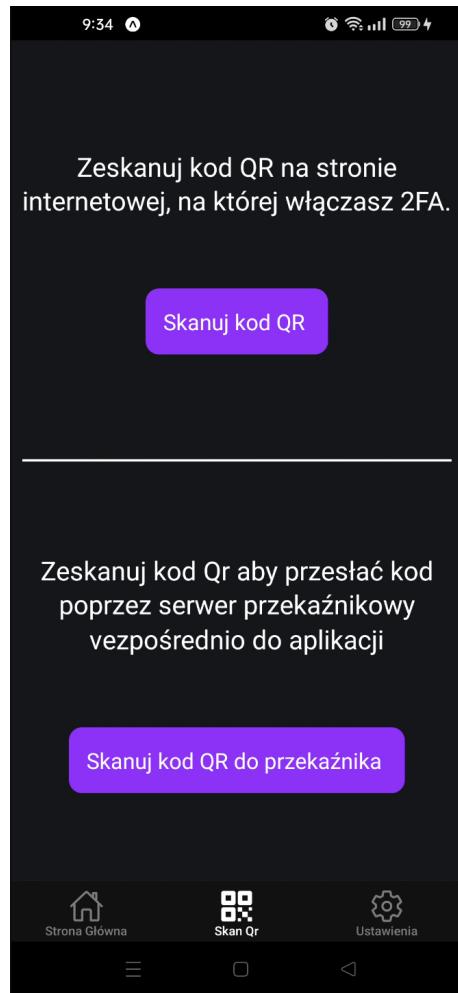


Rysunek 12: Menu deszyfrowania pliku - Opracowanie Własne

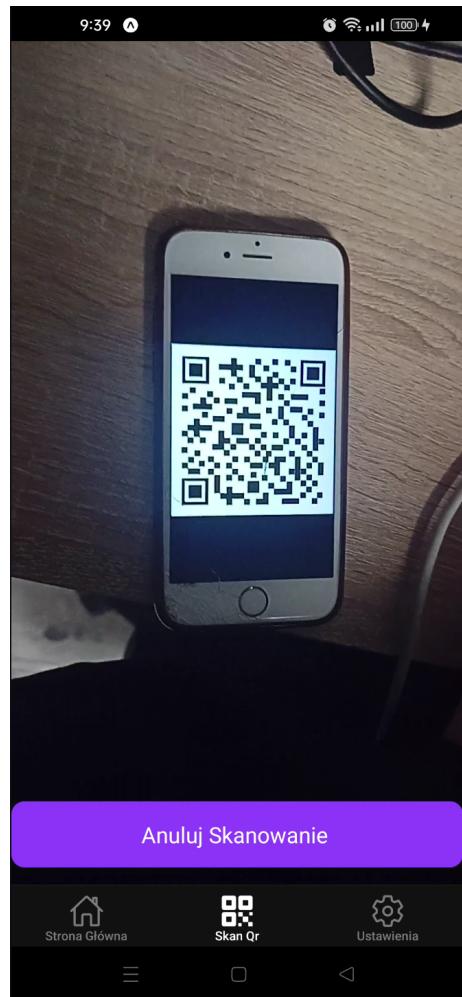
Na ekranie odszyfrowywania użytkownik wpisuje hasło, którym plik został za-szyfrowany. Podczas deszyfrowania aplikacja ponownie wyświetli nakładkę blokującą ekran z informacją o trwającym procesie (przykład na Rysunku 4). W przypadku poprawnego hasła aplikacja otworzy widok główny w zakładkach, opisany w rozdziale [3.6.3].

4.7.2. Rejestrowanie nowego sekretu konta

Aby generować hasła jednorazowe dla konta, należy je zarejestrować w aplikacji. W tym celu przejdź do zakładki „Skan QR”, upewniając się, że załadowano plik z danymi. Następnie dotknij przycisku „Skanuj kod QR”, który uruchomi przednią kamerę urządzenia, umożliwiając zeskanowanie kodu QR. Szczegółowe informacje o tym, jak aplikacja analizuje kod QR, znajdują się w rozdziale [4.4].

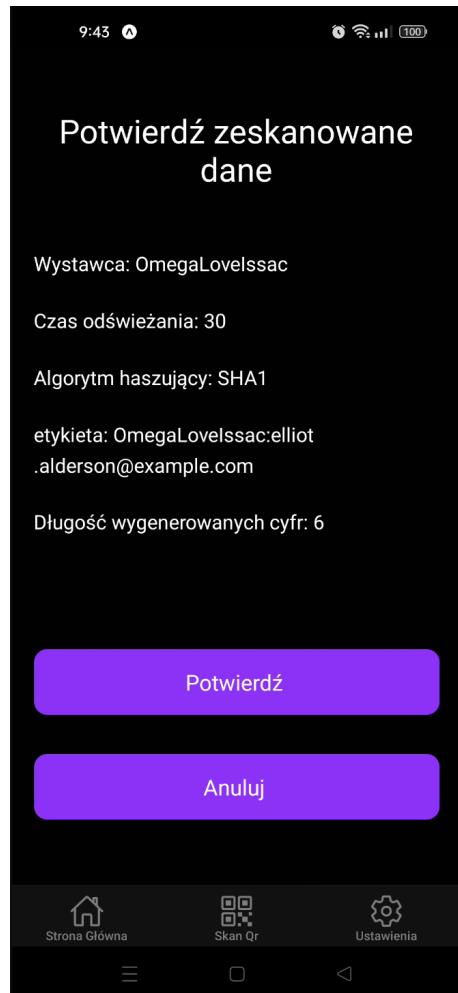


Rysunek 13: Menu skanowania kodów QR - Opracowanie Własne

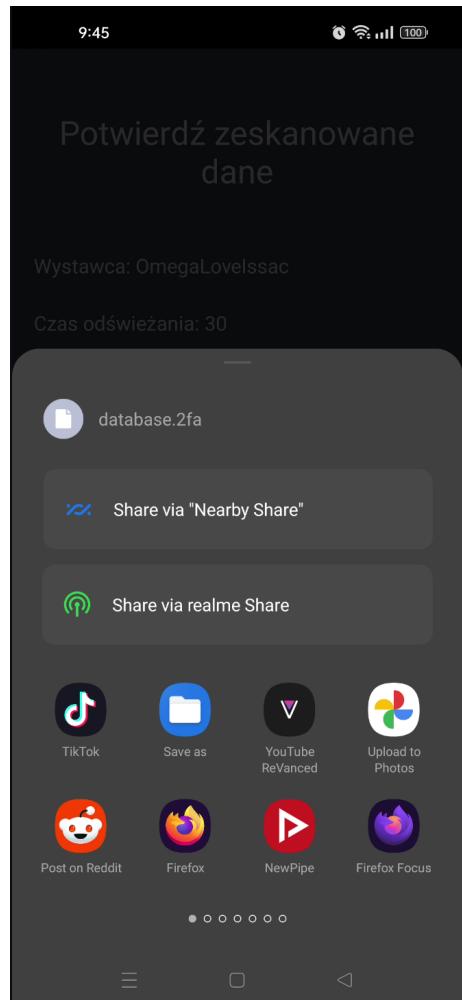


Rysunek 14: Skanowanie kodu QR - Opracowanie Własne

Jeśli kod QR zostanie zeskanowany poprawnie, aplikacja wyświetli okno potwierdzenia, w którym użytkownik zobaczy dane zawarte w kodzie QR. W tym miejscu dostępne są dwa przyciski: „Anuluj”, który przerywa proces dodawania danych, oraz „Potwierdź”, który zapisuje zeskanowane dane w pamięci aplikacji. Po zatwierdzeniu danych aplikacja utworzy nową wersję zaszyfrowanego pliku i wyświetli okno udostępniania z opcją „Save as”, pozwalającą nadpisać istniejący plik na urządzeniu.

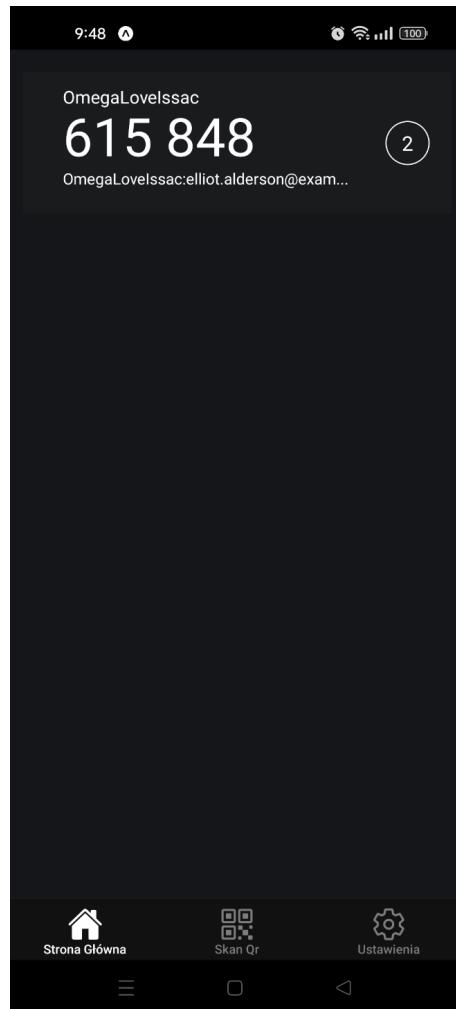


Rysunek 15: Okno potwierdzenia skanu kodu QR - Opracowanie Własne



Rysunek 16: Udostępnienie pliku do nadpisania na dysk - Opracowanie Własne

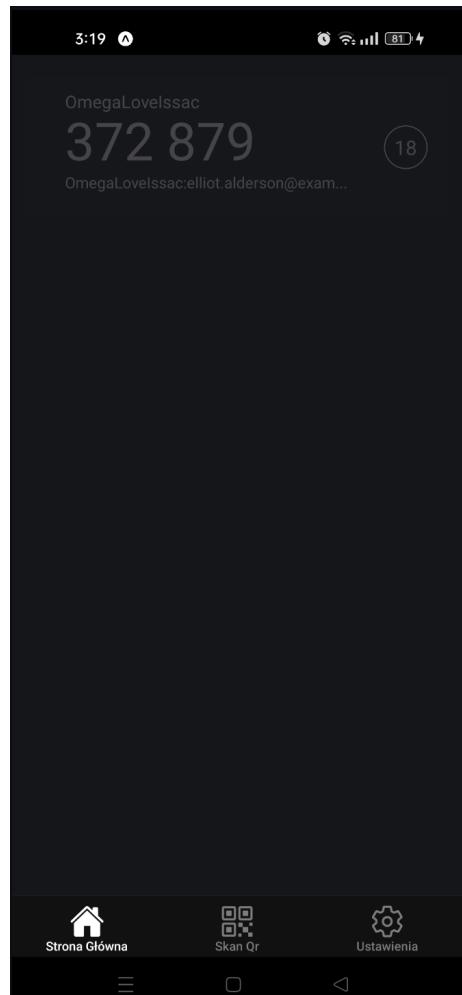
Po zapisaniu pliku użytkownik zostanie automatycznie przeniesiony do strony głównej aplikacji. W tym miejscu wyświetlane będzie generowane hasło jednorazowe.



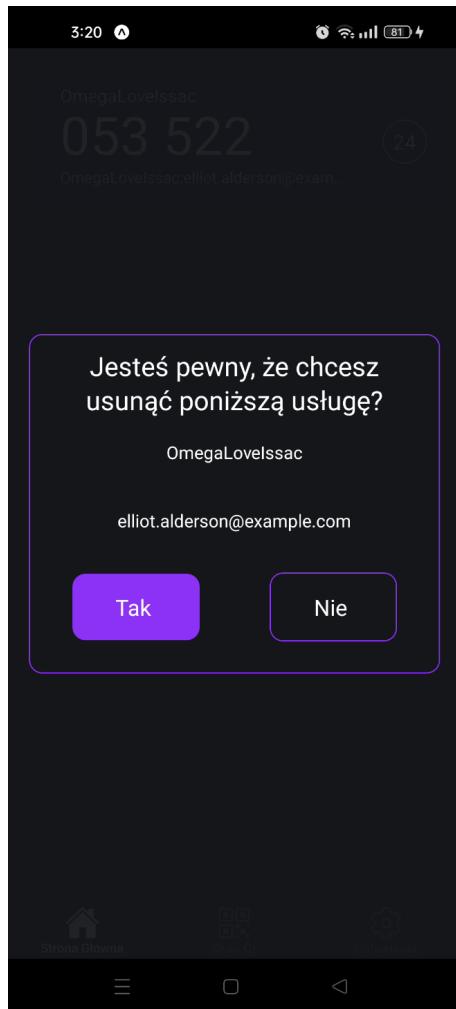
Rysunek 17: Strona główna z generowanym kodem - Opracowanie Własne

4.7.3. Usuwanie sekretu konta

Aby usunąć konkretny sekret, na podstawie którego generowane jest hasło jednorazowe, należy przejść do strony głównej aplikacji i przytrzymać palec na prostokącie wyświetlającym kod. Po chwili cały prostokąt zostanie przyciemniony, a użytkownikowi wyświetli się komunikat z pytaniem, czy na pewno chce usunąć wybrany sekret.



Rysunek 18: Przytrzymanie prostokąta z w celu jego usunięcia - Opracowanie Wła-sne



Rysunek 19: Komunikat z potwierdzeniem usunięcia sekretu - Opracowanie Własne

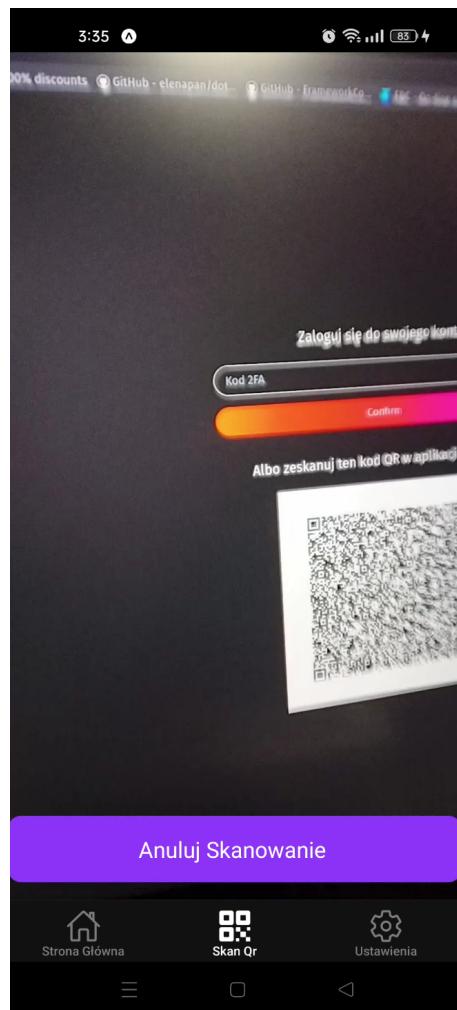
Po potwierdzeniu wyboru przyciskiem „Tak” aplikacja wyświetli nakładkę blokującą ekran i utworzy nową wersję zaszyfrowanego pliku, z którego usunięto wybrany sekret. Proces ten jest analogiczny do mechanizmu zapisu po dodaniu nowego sekretu. Nowy plik można nadpisać na urządzeniu za pomocą funkcji „Save as”.

4.7.4. Przekazanie kodu jednorazowego do serwera przekaźnika

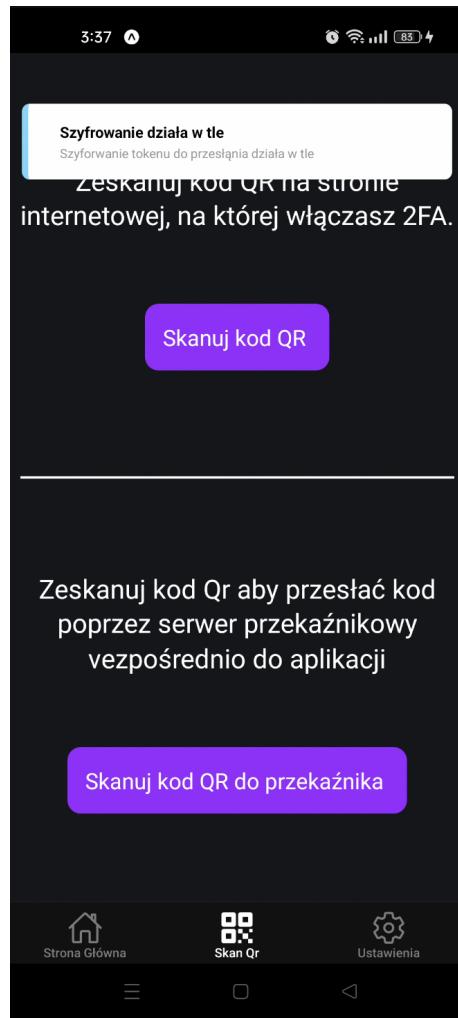
Aby przekazać kod jednorazowy do aplikacji użytkownika za pośrednictwem serwera przekaźnikowego, należy przejść do zakładki „Skan QR” i wybrać dolny przycisk „Skanuj kod QR do przekaźnika”. Po jego naciśnięciu uruchomi się kamera, umożliwiająca zeskanowanie kodu QR wyświetlonego w interfejsie użytkownika, który zawiera wymagane dane.

Jeśli dane zapisane w kodzie QR są poprawne, aplikacja mobilna wyświetli komunikat informujący o szyfrowaniu kodu jednorazowego przed jego przekazaniem. Szczegółowy opis mechanizmu działania aplikacji podczas tego procesu znajduje się

w rozdziale [4.5.2].



Rysunek 20: Skanowaniu kodu QR do serwera przekaźnika - Opracowanie Własne

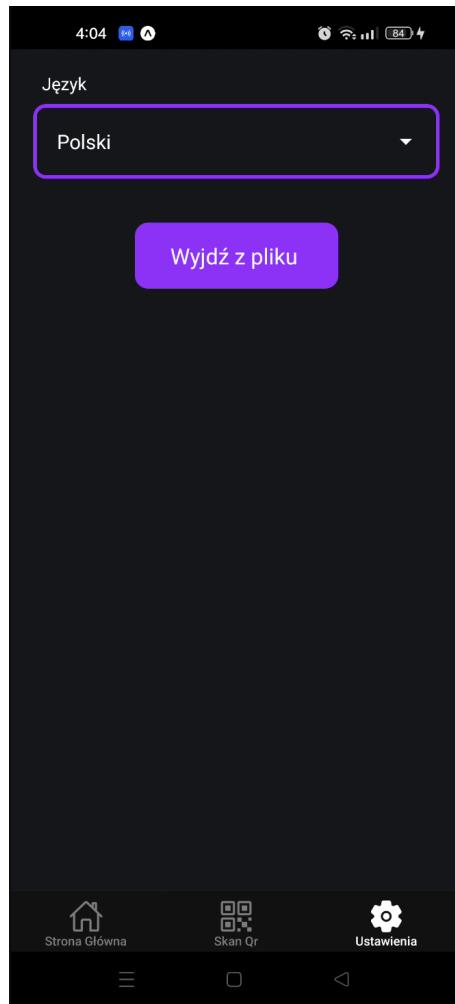


Rysunek 21: Informacja o szyfrowaniu hasła jednorazowego w celu przekazania go do serwera przekaźnikowego - Opracowanie Własne

4.7.5. Zmiana języka oraz zamknięcie pliku

Aby zmienić język aplikacji lub załadować inny plik, należy przejść do zakładki „Ustawienia”. W zakładce tej znajduje się selektor umożliwiający wybór preferowanego języka. Zmiany w ustawieniach językowych zapisywane są automatycznie, co eliminuje konieczność dodatkowego potwierdzania.

Pod selektorem znajduje się przycisk „Wyjdź z pliku”, który pozwala bezpiecznie wyczyścić wszystkie dane wcześniej odczytane z pliku. Po naciśnięciu przycisku użytkownik zostanie przeniesiony z powrotem do Widoku głównego.



Rysunek 22: Zakładka ustawień - Opracowanie Własne

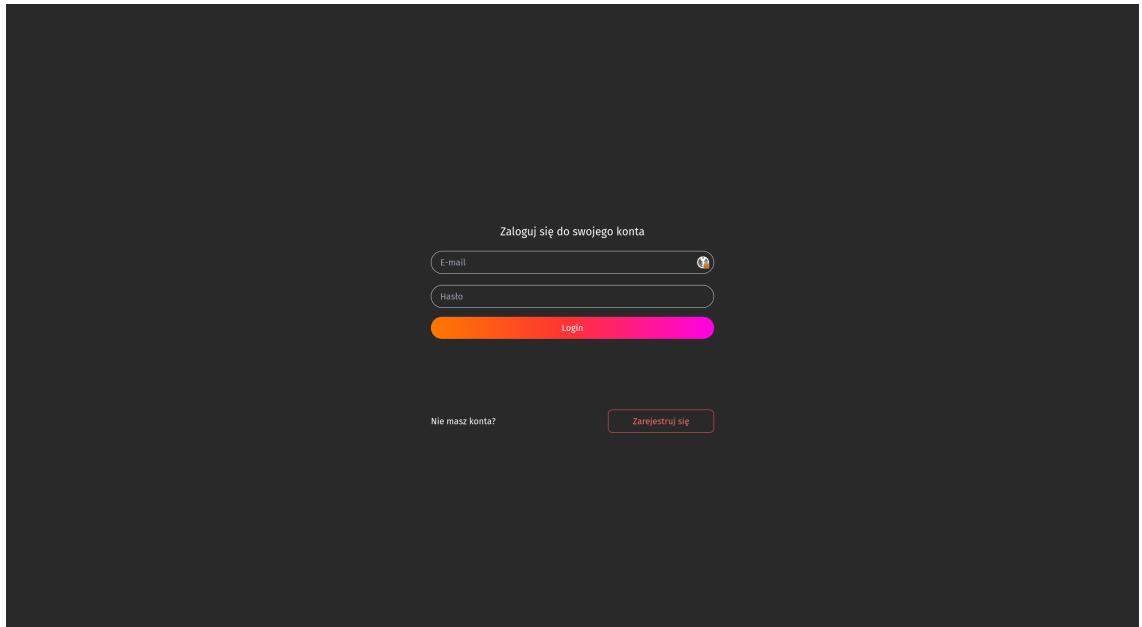
4.8. Instrukcja użytkowania aplikacji Webowej

4.8.1. Tworzenie użytkownika i logowanie

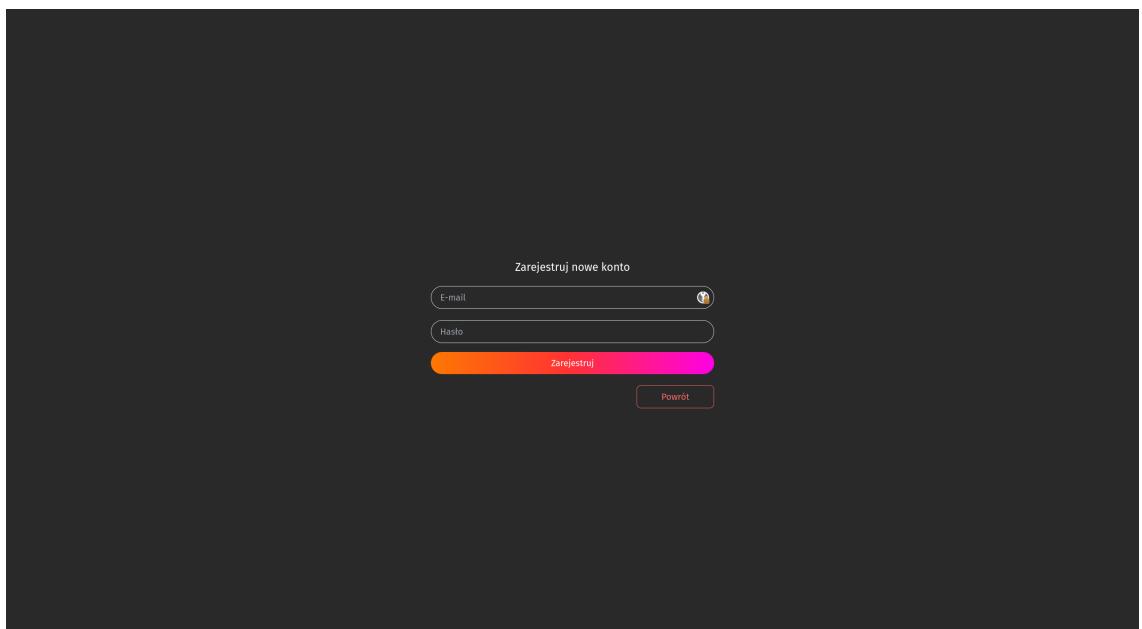
Pierwszą stroną, która powita użytkownika, jest strona logowania. W przypadku braku zarejestrowanego konta konieczne jest jego utworzenie. Aby to zrobić, należy nacisnąć przycisk „Zarejestruj się”, który przekieruje użytkownika na stronę rejestracji konta.

Na stronie rejestracji użytkownik powinien wprowadzić swój adres e-mail oraz hasło, a następnie nacisnąć przycisk „Zarejestruj”. Po pomyślnym sprawdzeniu poprawności podanych danych użytkownik zostanie automatycznie przekierowany z powrotem na stronę logowania. Tam, po wpisaniu swoich danych logowania, zostanie przeniesiony na stronę z danymi użytkownika.

Strona z danymi użytkownika umożliwia wylogowanie się z konta oraz skonfigurowanie weryfikacji dwuetapowej.



Rysunek 23: Strona logowania aplikacji webowej - Opracowanie Własne



Rysunek 24: Strona rejestracji użytkownika aplikacji webowej - Opracowanie Własne

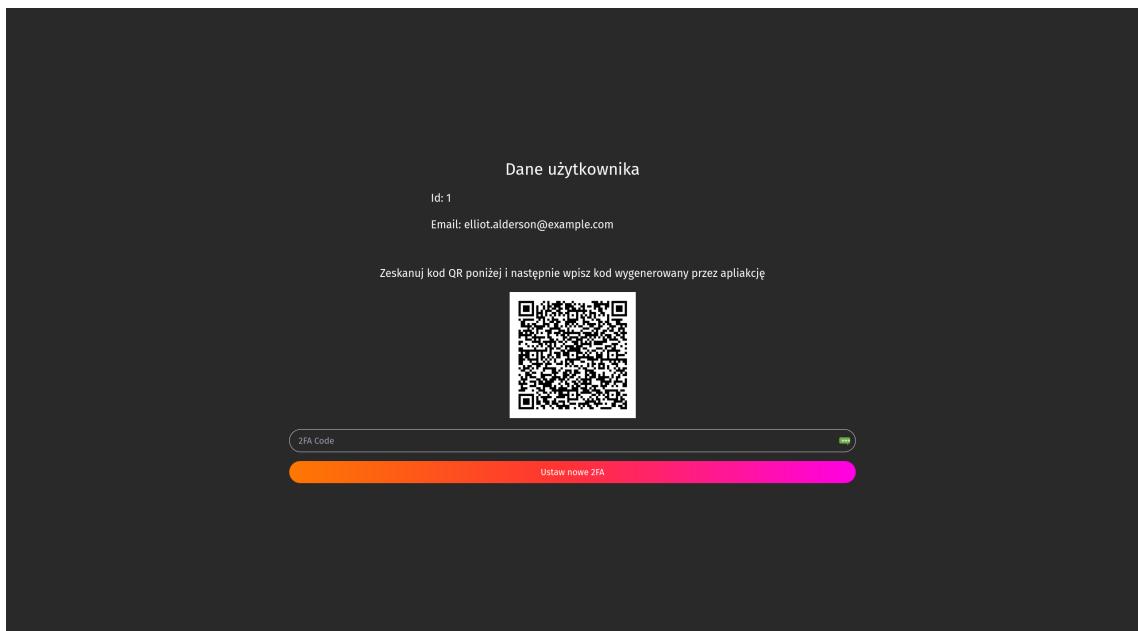


Rysunek 25: Strona z danymi użytkownika w aplikacji webowej - Opracowanie Właścne

4.8.2. Skonfigurowanie weryfikacji dwuetapowej

W celu ustawienia weryfikacji dwuetapowej na koncie użytkownika należy najpierw zalogować się do aplikacji webowej. Na stronie z danymi użytkownika znajduje się przycisk „Ustaw 2FA”. Po jego naciśnięciu, aplikacja nawiązuje kontakt z serwerem i wyświetla unikalny kod QR oraz formularz wprowadzenia kodu.

Kod QR należy zeskanować za pomocą kompatybilnej aplikacji mobilnej, która obsługuje generowanie haseł jednorazowych. Następnie użytkownik powinien wprowadzić wygenerowany kod w formularzu pod kodem QR i nacisnąć przycisk „Ustaw nowe 2FA”. Szczegółowy opis procesu skanowania kodu QR w aplikacji mobilnej znajduje się w rozdziale [4.5.2].



Rysunek 26: Ustawianie weryfikacji dwuetapowej w aplikacji webowej - Opracowanie Własne

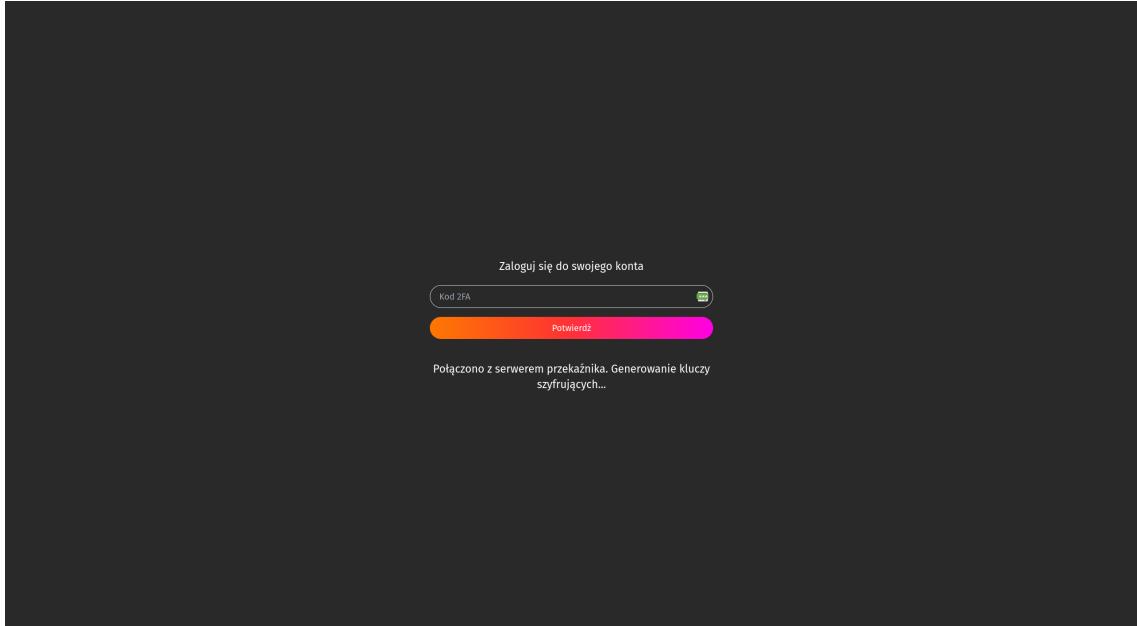
Jeśli wprowadzony kod jednorazowy jest poprawny, proces konfiguracji kończy się pomyślnie. Formularz zamknie się, a użytkownik zostaje przeniesiony z powrotem do strony z danymi konta, tak jak przedstawiono na rysunku 25.

Ponowne naciśnięcie przycisku „Ustaw nowe 2FA” umożliwia nadpisanie istniejącego sekretu użytkownika na serwerze, co unieważnia wcześniej zeskanowany kod QR. Dzięki temu użytkownik może łatwo zaktualizować ustawienia weryfikacji dwuetapowej w razie potrzeby.

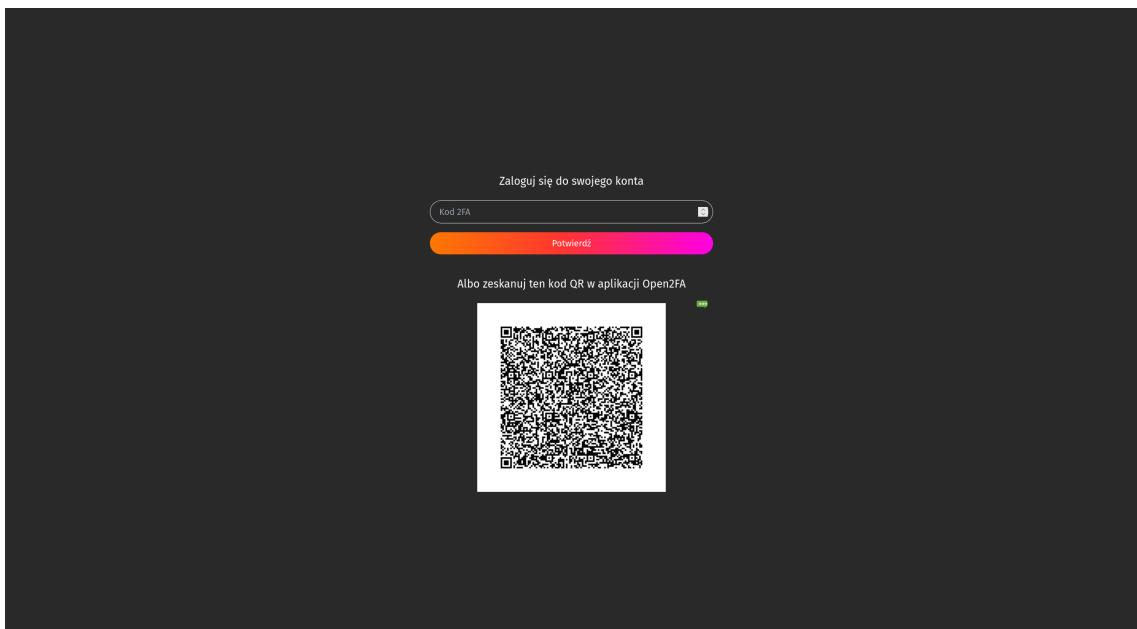
4.8.3. Korzystanie z serwera przekaźnika

Jeśli konto użytkownika ma włączoną weryfikację dwuetapową, podczas próby logowania wyświetlany jest formularz, który umożliwia wpisanie jednorazowego kodu. Użytkownik może ręcznie wprowadzić kod z aplikacji mobilnej lub skorzystać z serwera przekaźnikowego. Jeśli aplikacja webowa jest połączona z serwerem przekaźnikowym, po krótkim czasie wygenerują się klucze szyfrujące i pod formularzem pojawi się kod QR. Można go zeskanować za pomocą aplikacji mobilnej, aby uzyskać dostęp do serwerów przekaźników. Proces ten został szczegółowo opisany w rozdziale [4.5.4].

W obu przypadkach, jeśli podany kod jednorazowy jest poprawny, proces logowania zostanie pomyślnie zakończony, a użytkownik zostanie przekierowany do strony z danymi konta.



Rysunek 27: Formularz wpisania kodu weryfikacyjnego przy próbie logowania przed wygenerowaniem kluczy szyfrujących - Opracowanie Własne



Rysunek 28: Formularz wpisania kodu weryfikacyjnego przy próbie logowania po wygenerowaniu kluczy szyfrujących - Opracowanie Własne

5. Podsumowanie

Projekt „Aplikacja do weryfikacji dwuetapowej Open2FA” był koncentrowany na stworzeniu aplikacji mobilnej która w połączeniu z systemem przekaźników ułatwiała proces weryfikacji dwuetapowej. Aplikacja umożliwia generowanie kodów jednorazowych, ich bezpieczne przechowywanie oraz przesyłanie za pomocą kodów QR, co znacząco usprawnia proces uwierzytelniania w systemach wymagających podwyższonego poziomu bezpieczeństwa.

Podczas realizacji projektu szczególny nacisk położono na innowacyjne rozwiązania zwięksające funkcjonalność i ochronę danych użytkownika. Sekrety niezbędne do generowania kodów jednorazowych są przechowywane w zaszyfrowanym pliku, co minimalizuje ryzyko nieautoryzowanego dostępu. Wprowadzono także system przekaźników, który umożliwia synchronizację i szybki dostęp do haseł na różnych urządzeniach bez konieczności ręcznego wprowadzania danych.

Aplikacja została napisana w React Native, co zapewnia jej działanie na systemach Android i iOS. Stylizację zrealizowano przy użyciu Tailwind CSS, a zarządzanie stanem aplikacji obsłużono za pomocą Redux Toolkit. Do obsługi generowania i weryfikacji kodów jednorazowych wykorzystano biblioteki takie jak otpauth i node-forge. Bezpieczeństwo kluczy użytkownika zapewniono dzięki zastosowaniu algorytmu PBKDF2, a dane przesyłane między urządzeniami są zabezpieczone szyfrowaniem.

Projekt ma potencjał do rozwoju. Jednym z priorytetowych kierunków rozwoju jest umożliwienie zapisu zaszyfrowanego pliku bezpośrednio na dysk urządzenia, co pozwoli użytkownikowi na automatyczne przechowywanie danych bez konieczności każdorazowego udostępniania pliku. Dalsze prace nad serwerem przekaźnikowym mogą skupić się na dodaniu obsługi skalowalności, aby zapewnić lepszą wydajność przy rosnącej liczbie użytkowników oraz większą niezawodność w przesyłaniu danych. Rozważane jest również wprowadzenie funkcji eksportu sekretów przechowywanych w Open2FA do innych aplikacji obsługujących weryfikację dwuetapową, co zwiększy elastyczność użytkowania oraz możliwość integracji z różnymi systemami.

Open2FA stanowi odpowiedź na rosnące potrzeby użytkowników w zakresie wygodnej i bezpiecznej weryfikacji dwuetapowej, jednocześnie wprowadzając innowacyjne rozwiązania, które wykraczają poza standardowe podejście do tego typu aplikacji. Dodatkowo ma ona potencjał na rozwój, który może stanowić podstawę do przyszłych rozszerzeń lub udoskonaleń.

References

- [1] Twilio Inc *What is a Time-based One-time Password (TOTP)?*
Adres Url: <https://www.twilio.com/docs/glossary/totp>
Dostęp: 10.12.2024
- [2] OneLogin *What's the Difference Between OTP, TOTP and HOTP?*
Adres Url: <https://www.onelogin.com/learn/otp-totp-hotp>
Dostęp: 10.12.2024
- [3] Meltem Sönmez Turan, Elaine Barker, William Burr, Lily Chen *Recommendation for Password-Based Key Derivation*
Adres Url: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf>
Dostęp: 10.12.2024
- [4] fortinet *What Is A Brute Force Attack?*
Adres Url: <https://www.fortinet.com/resources/cyberglossary/brute-force-attack>
Dostęp: 10.12.2024
- [5] D. Eastlake 3rd, Huawei, T. Hansen, AT&T Labs *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)*
Adres Url: <https://www.rfc-editor.org/rfc/rfc6234>
Dostęp: 10.12.2024
- [6] Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg *Advanced Encryption Standard (AES)*
Adres Url: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf>
Dostęp: 10.12.2024
- [7] K. Moriarty, Ed. , EMC Corporation, B. Kaliski, Verisign, J. Jonsson, Subset AB , A. Rusch *PKCS #1: RSA Cryptography Specifications Version 2.2*
Adres Url: <https://www.rfc-editor.org/rfc/rfc8017>
Dostęp: 10.12.2024

- [8] S. Josefsson, SJD *The Base16, Base32, and Base64 Data Encodings*
Adres Url: <https://www.rfc-editor.org/rfc/rfc4648>
Dostęp: 10.12.2024
- [9] Vint Cerf, UCLA *ASCII format for Network Interchange*
Adres Url: <https://www.rfc-editor.org/rfc/rfc20>
Dostęp: 10.12.2024
- [10] Tom Preston-Werner *Toml - strona główna*
Adres Url: <https://toml.io/en/>
Dostęp: 10.12.2024
- [11] DENSO WAVE INCORPORATED *QR Code standardization*
Adres Url: <https://www.qrcode.com/en/about/standards.html>
Dostęp: 10.12.2024
- [12] MDN contributors. *HTTP resources and specifications*
Adres Url: https://developer.mozilla.org/en-US/docs/Web/HTTP/Resources_and_specifications
Dostęp: 10.12.2024
- [13] Kinza Yasar *What is web application (web apps) and its benefits?*
Adres Url: <https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app>
Dostęp: 10.12.2024
- [14] Marshall Gunnell, Natalie Medleva *Mobile Application*
Adres Url: <https://www.techopedia.com/definition/2953/mobile-application-mobile-app>
Dostęp: 10.12.2024
- [15] D. M'Raihi, Verisign, Inc., S. Machani, Diversinet Corp., M. Pei, Symantec, J. Rydell, Portwise, Inc. *TOTP: Time-Based One-Time Password Algorithm*
Adres Url: <https://www.rfc-editor.org/rfc/rfc6238>
Dostęp: 10.12.2024
- [16] MDN contributors. *JavaScript*
Adres Url: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
Dostęp: 10.12.2024

- [17] Microsoft *TypeScript Documentation*
Adres Url: <https://www.typescriptlang.org/docs/>
Dostęp: 10.12.2024
- [18] Meta Platforms, Inc. *React Native Documentation*
Adres Url: <https://reactnative.dev/docs/accessibilityinfo>
Dostęp: 10.12.2024
- [19] Expo Documentation Team. *Expo Docs.*
Adres Url: <https://docs.expo.dev/>
Dostęp: 10.12.2024
- [20] Dan Abramov and the Redux documentation authors. *Redux Toolkit*
Adres Url: <https://redux-toolkit.js.org>
Dostęp: 10.12.2024
- [21] Atlassian *What is Git?*
Adres Url: <https://www.atlassian.com/git/tutorials/what-is-git>
Dostęp: 10.12.2024
- [22] Preetam Gahlot *Securing Data Locally with Crypto ES: A Guide for React Native Developers.*
Adres Url: <https://medium.com/@PreetamGahlot/securing-data-locally-with-crypto-es-a-guide-for-react-native-developers-a84c3dfe5ce7>
Dostęp: 10.12.2024
- [23] Tailwind Labs Inc. *Tailwind Documentation*
Adres Url: <https://tailwindcss.com/docs/installation>
Dostęp: 10.12.2024
- [24] Digital Bazaar, Inc. *forge github project*
Adres Url: <https://github.com/digitalbazaar/forge>
Dostęp: 10.12.2024
- [25] calintamas *react-native-toast-message github project*
Adres Url:
<https://github.com/calintamas/react-native-toast-message>
Dostęp: 10.12.2024
- [26] Héctor Molinero Fernández. *otpauth Documentation*
Adres Url: <https://hector.m.github.io/otpauth/>
Dostęp: 10.12.2024

[27] Colin McDonnell *Zod Docs*

Adres Url: <https://zod.dev/>

Dostęp: 10.12.2024

[28] MDN contributors. *JSON*

Adres Url: <https://developer.mozilla.org/en-US/docs/Glossary/JSON>

Dostęp: 10.12.2024

[29] Meta Platforms, Inc. *React Documentation*

Adres Url: <https://react.dev/learn>

Dostęp: 10.12.2024

[30] VoidZero Inc. & Vite Contributors. *Vite Documentation*

Adres Url: <https://vite.dev/guide/>

Dostęp: 10.12.2024

[31] Shopify, Inc. *React Router Documentation*

Adres Url: <https://reactrouter.com/home>

Dostęp: 10.12.2024

[32] Fadi Khadra *React-toastify Documentation*

Adres Url: <https://fkhadra.github.io/react-toastify/introduction/>

Dostęp: 10.12.2024

[33] Socket.IO *Socket.IO Documentation*

Adres Url: <https://socket.io/docs/v4/>

Dostęp: 10.12.2024

[34] veksenn & zthall. *Lodash Documentation*

Adres Url: <https://lodash.com/docs/4.17.15>

Dostęp: 10.12.2024

[35] Auth0 *Introduction to JSON Web Tokens*

Adres Url: <https://jwt.io/introduction>

Dostęp: 10.12.2024

[36] Hipp, Wyrick & Company, Inc. *SQLite Documentation*

Adres Url: <https://www.sqlite.org/docs.html>

Dostęp: 10.12.2024

- [37] Sequelize Contributors. *Sequelize Documentation*
Adres Url: <https://sequelize.org/docs/v6/getting-started/>
Dostęp: 10.12.2024
- [38] Dan Arias *Hashing in Action: Understanding bcrypt*
Adres Url: <https://auth0.com/blog/hashing-in-action-understanding-bcrypt/>
Dostęp: 10.12.2024
- [39] Rust Team. *Rust Programming Language*
Adres Url: <https://www.rust-lang.org/>
Dostęp: 10.12.2024
- [40] David Pedersen *Rust Programming Language*
Adres Url:
<https://tokio.rs/blog/2021-07-announcing-axum#learn-more>
Dostęp: 10.12.2024
- [41] Rust Community *Rust Tokio Documentation*
Adres Url: <https://tokio.rs/tokio/tutorial>
Dostęp: 10.12.2024
- [42] Rust Community *Serde*
Adres Url: <https://serde.rs/>
Dostęp: 10.12.2024
- [43] Big Human *What is Minimalistic UI? A Guide to the Art of Minimal UI Design*
Adres Url:
<https://www.bighuman.com/blog/guide-to-minimalistic-user-interface-design>
Dostęp: 10.12.2024
- [44] Wikipedia *Tablica znaków ASCII*
Adres Url: <https://simple.m.wikipedia.org/wiki/File:ASCII-Table-wide.svg>
Dostęp: 10.12.2024