Politechnika Poznańska
Wydział Elektryczny
Instytut Automatyki, Robotyki i Inżynierii Informatycznej

Praca dyplomowa inżynierska

ANALIZA DZIAŁANIA I BEZPIECZEŃSTWA APLIKACJI WWW OPARTYCH NA JĘZYKU JAVASCRIPT Z WYKORZYSTANIEM NOWOCZESNYCH TECHNOLOGII IMPLEMENTACJI WARSTWY PREZENTACJI.

SECURITY AND FUNCTIONING ANALISYS OF APPLICATIONS BASED ON JAVASCRIPT LANGUAGE WITH THE USE OF MODERN FRONTEND TECHNOLOGIES.

Krzysztof Figiel

Promotor
dr inż. Izabela Janicka-Lipska

Poznań, 2018 r.

Temat
pracy dyplomowej inżynierskiej

Uczelnia:	Politechnika Poznańska	Profil kształcenia:	ogólnoakademicki
Wydział:	Elektryczny	Forma studiów:	stacjonarne
Kierunek:	Informatyka	Poziom studiów:	I stopnia
Specjalność:	Bezpieczeństwo systemów informatycznych		

Zobowiązuję/zobowiązujemy się samodzielnie wykonać pracę w zakresie wyspecyfikowanym niżej. Wszystkie elementy (m.in. rysunki, tabele, cytaty, programy komputerowe, urządzenia itp.), które zostaną wykorzystane w pracy, a nie będą mojego/naszego autorstwa, będą w odpowiedni sposób zaznaczone i będzie podane źródło ich pochodzenia.

	Imię i nazwisko	Nr albumu	Data i podpis
Student:	**Krzysztof FIGIEL**	**123652**	14.09.2017 Figiel
Student:			

Tytuł pracy:	DIALOG – elektroniczny dzienniczek diabetyka
Wersja angielska tytułu:	*DIALOG – online diabetic register*
Dane wyjściowe:	Literatura przedmiotu.
Zakres pracy:	1. Projekt i implementacja internetowego dzienniczka diabetyka. 2. System zarządzania bazą danych (monitorowanie pracy aplikacji, gromadzenie i analiza wyników, podstawowe obliczenia związane z tematyką). 3. Zapewnienie bezpieczeństwa systemu oraz przyjaznego interfejsu użytkownika. 4. Testowanie i wnioski.
Termin oddania pracy:	31 stycznia 2018
Promotor:	dr inż. Izabela Janicka-Lipska
Jednostka organizacyjna promotora:	Instytut Automatyki i Inżynierii Informatycznej

Z-ca DYREKTORA INSTYTUTU
Automatyki, Robotyki
i Inżynierii Informatycznej
dr Jerzy Bartoszek

podpis dyrektora/kierownika jednostki organizacyjnej promotora

PRODZIEKAN
Wydziału Elektrycznego
Politechniki Poznańskiej
dr hab. inż. Andrzej Tomczewski

podpis Dziekana

Poznań, 14 września 2017
miejscowość, data

Streszczenie

Za cel pracy postawiono omówienie oraz analizę działania i bezpieczeństwa aplikacji WWW opartych na języku *JavaScript* z wykorzystaniem nowoczesnych technologii implementacji warstwy prezentacji takich jak *Angular*, hybrydowa technologia *Ionic*, framework *React* czy *Vue.js*. W celu lepszego poznania odpowiednich bibliotek i wzorców projektowych z nimi powiązanych posłużono się dokumentacjami technicznymi poszczególnych zestawów narzędzi. Przeanalizowano mechanizmy bezpieczeństwa oferowane przez frameworki. Wykorzystano autorskie implementacje kluczowych fragmentów funkcjonowania aplikacji internetowych i na ich podstawie zestawiono i omówiono otrzymane wyniki i wnioski.

Abstract

The main aim of this graduation work is to discuss and analyze the functionality and security of WWW applications based on *JavaScript* language using modern frontend technologies like *Angular*, hybrid framework *Ionic*, *React* and *Vue.js*. For better understanding each of the technologies technical documentation was used. The security mechanisms offered by each framework were analyzed in depth. Authors implementations of key fragments of the functioning of WWW applications were used. The obtained results anc conclusions were summarized and discussed.

Spis treści

Rozdział 1

Wstęp

1.1 Wprowadzenie

Wybór tematu pracy magisterskiej spowodowany był chęcią udoskonalenia i poszerzenia swojej wiedzy na temat tworzenia nowoczesnych aplikacji WWW opartych na języku *JavaScript* z wykorzystaniem technologii implementacji warstwy prezentacji oraz mechanizmów bezpieczeństwa z nimi powiązanych. Aktualny rynek pracy i ciagły rozwój technologii informatycznych (zwłaszcza tych internetowych) powodują stałe udoskonalanie aktualnych rozwiązań technologicznych, a co za tym idzie uodparnianie ich na problemy związane z bezpiecznym przechowywaniem danych. Kolejnym argumentem, który wskazuje na mój wybór jest fakt, że aktualnie pracuję na stanowisku web dewelopera, tak więc tworzenie bezpiecznych aplikacji internetowych jest zarówno źródłem mojego dochodu, jak i zainteresowań. W dzisiejszych czasach programista ma dostęp do wielu użytecznych bibliotek i rozwiązań, które w znacznym stopniu ułatwiają implementację najważniejszych mechanizmów komunikacji poszczególnych warstw aplikacji z serwerem, dlatego też postanowiłem omówić najważniejsze z nich wraz z krótkimi implementacjami podstawowych funkcjonalności stron WWW.

1.2 Cel projektu

Za cel projektu postawiono analizę działania i bezpieczeństwa aplikacji opartych na języku *JavaScript* w oparciu o proste implementacje podstawowych funkcjonalności serwisów WWW takich jak uwierzytelnianie, formularze użytkownika czy wymiana danych między warstwami prezentacji i aplikacji. Skupiono się na zrealizowaniu podstawowych mechanizmów prewencji przeciwko atakom z poziomu warstwy aplikacji. Zaproponowano również autorskie pomysły metod dodatkowego zabezpieczania stron WWW przeciwko nieporządanemu działaniu osób trzecich. Zestawiono sposoby realizacji powyższych rozwiązań w poszczególnych technologiach frontendowych i określono, która z nich jest potencjalnie najlepsza. Ostatecznie, zaproponowano ewentualne metody ulepszeń omawianych mechanizmów i podsumowano całokształt pracy.

1.3 Różnice pomiędzy istniejącymi aplikacjami

Główną zaletą projektu jest jego dostępność. Większość aplikacji dostępnych na rynku ograniczona jest do jednego systemu operacyjnego. Tutaj celem było stworzenie aplikacji dostępnej z poziomu przeglądarki internetowej, co znacznie poszerza jej zakres kompatybilności.

1.4 Struktura pracy (ToDo)

Praca składa się z X rozdziałów. Pierwszy zawiera wprowadzenie, ukazuje cel projektu oraz przedstawia różnice pomiędzy istniejącymi aplikacjami. W drugim opisane są technologie wykorzystane do stworzenia aplikacji, wymagania funkcjonalne z podziałem na aktorów i niefunkcjonalne wraz z bardziej szczegółową analizą podobnych aplikacji istniejących na rynku. Trzeci rozdział opisuje procesy zachodzące w systemie. Kolejny, czwarty, przedstawia dokładny opis bazy danych. W piątym rozdziale omówiono panel administratora aplikacji. W następnych dwóch przedstawiono kolejno panel użytkownika zarejestrowanego oraz panel użytkownika niezarejestrowanego i niezalogowanego. Rozdział ósmy dotyczy bezpieczeństwa aplikacji i opisuje wszystkie aspekty z

nim związane. W rozdziale dziewiątym przedstawiono testy funkcjonalne wykonane na aplikacji. Ostatni, dziesiąty rozdział jest podsumowaniem całej pracy. Omówiono tam także dalsze etapy rozwoju aplikacji. Na końcu znajduje się spis literatury. Do pracy dołączono załącznik w postaci płyty DVD.

Rozdział 2

Technologie

2.1 Wprowadzenie

W poniższym rozdziale opisano najważniejsze technologie opisane w pracy. W kolejnych sekcjach przedstawiono zestaw wymagań funkcjonalnych z podziałem na biorących udział aktorów oraz wymagania niefunkcjonalne.

2.2 Technologie

2.2.1 Framework Angular 6

Otwarty framework stworzony przez firmę *Google*, wykorzystywany do tworzenia aplikacji SPA (*Single Page Application*) zarówno na platformy internetowe, jak i na natywne aplikacje mobilne i desktopowe. *Angular* oparty jest na języku *JavaScript* przez co zyskał on wielką popularność wśród deweloperów, w dużej mierze ze względu na swoją prostotę. Jego struktura wymusza u programistów stosowanie dobrych praktyk pisania kodu i pomaga usestymatyzować na pozór skomplikowaną architekturę aplikacji webowych. Framework ten został napisany całkowicie w mocno typowanym języku *TypeScript*, który jest transpilowany do wynikowego kodu *JavaScript*. *Angular* swoją strukturą zachęca do budowania maksymalnie odseparowanych od siebie części kodu i komponentów. Kolejną zaletą tego frameworku jest wieloplatformowość. Pozwala on bowiem na tworzenie stron internetowych, aplikacji webowych, aplikacji PWA (*Progressive Web Application*), aplikacji mobilnych z użyciem bibliotek takich jak *Cordova* czy też aplikacji desktopowych mogących odnosić się do funkcji systemu i lokalnych urządzeń. Wersja szósta frameworka *Angular* wprowadza zmiany m.in. w rejestrowaniu serwisów, stosowaniu *RxJS*, walidacji formularzy czy w komendach CLI (*Command Line Interface*).

2.2.2 TypeScript 3

TypeScript jest językiem kompilowanym do języka *JavaScript*, który urchamiany jest w dowolnej przeglądarce, na serwerze *Node.js* czy w innym silniku, który wspiera *ECMAScript* w wersji trzeciej lub nowszej. Jest to język silnie i statycznie typowany, który pomaga deweloperom stosować dobre, programistyczne praktyki. Statyczne typowanie rozumiane jest przez to, że zmienne w tym języku mają nadane typy, które nie mogą ulec zmianie. Silne typowanie wprowadza nie tylko przejrzystość kodu i jego lepsze debugowanie, ale również takie możliwości jak *IntelliSense*, czyli podpowiedzi. Zastosowanie *TypeScript* pozwala pozbyć się przypadkowych błędów związanych z niepewnością odnośnie typów danych oraz eliminuje problemy związane ze skalowalnością. Umożliwia stosowanie interfejsów, które pozwalają podnieść poziom abstrakcji i uzyskać luźniejsze powiązania pomiędzy klasami w aplikacjach.

2.2.3 HTML5

HTML5 (*HyperText Markup Language*) jest najnowszą wersją popularnego standardu opisującego język HTML. W stosunku do poprzedników zawiera on nowe elementy, atrybuty i zachowania. Pozwala na bardziej różnorodne tworzenie stron i aplikacji internetowych. Umożliwia nowoczesną komunikację z serwerem, pozwala stronom internetowym na bardziej efektywne przechowywanie

danych lokalnie i w trybie offline czy zapewnie większą prędkość i lepszą optymalizację w wykorzystywaniu sprzętu komputerowego. HTML5 wprowadza nowe elementy sekcji takie jak `<section>`, `<article>`, `<nav>`, `<header>` czy `<footer>`. Ulepszone zostały m.in. formularze, wymuszenie poprawności API (*Application Programming Interface*) czy znaczniki `<input>` i `<output>`, które zyskały nowe atrybuty, takie jak `email` oraz `password`. HTML5 zapewnia również uproszczone odtwarzanie plików audio i wideo. Oferuje dużo prostsze tworzenie i wyświetlanie grafiki przy użyciu znaczników `<canvas>`. Ważną nowością są też tzw. *Web-Workers*, które umożliwiają wielowątkową obsługę przepływu danych.

2.2.4 CSS3

CSS3 (*Cascading Style Sheets*) jest to kolejna wersja kaskadowych arkuszy stylów, wprowadzająca szereg udogodnień i rozszerzająca możliwości interakcji użytkownika ze stroną internetową. W porównaniu do poprzednich wersji, zmiany w CSS3 obejmują między innymi zastosowanie animowanych elementów czy wszelkiego rodzaju efektów graficznych, takich jak gradienty oraz cienie. CSS3 jest kompatybilny ze wszystkimi stylami objętymi standardem CSS2. Nie ma zatem potrzeby modyfikowania starszych stron przy przejściu na wyższą wersję kaskadowych arkuszy stylów. Trzecia wersja CSS zyskała również modułową budowę – specyfikacja zostaje podzielona na wiele różnych dokumentów, dzięki czemu rozwój odrębnych modułów aplikacji odbywa się odrębnie, a kod jest czysty i uporządkowany. Style CSS dodawane są do elementów na podstawie ich pozycji w drzewie dokumentu (*Document Tree*). CSS jest w pełni kompatybilny z językiem HTML, co oznacza, że HTML strukturyzuje treść strony, natomiast CSS formatuje ją w odpowiedni sposób. CSS3 pozwala na całkowitą kontrolę układu graficznego dokumentów z poziomu tylko jednego arkusza stylów. Ponadto, programista ma bardziej precyzyjną kontrolę nad całym układem graficznym [Fra15].

2.2.5 Bootstrap 4

Jedna z najbardziej popularnych bibliotek dla języka HTML, CSS i *JavaScript*. *Bootstrap* to zestaw przydatnych narzędzi wykorzystywany do tworzenia responsywnych interfejsów aplikacji internetowych oraz mobilnych. Bazuje w dużej mierze na gotowych implementacjach HTML i CSS, które kompilowane są bezpośrednio z pliku *Less*. Biblioteka może być wykorzystywana w celu stylizowania formularzy, tekstów, przycisków, elementów menu i wielu innych przydatnych komponentów stron internetowych. Jedną z najważniejszych cech charakteryzujących *Bootstrap* jest to, że wprowadza on system siatek (*Grid*) mający na celu usystematyzowanie położenia elementów na stronie. Wszystko opiera się na dzieleniu strony na rzędzy (*rows*), a rzędy na kolumny (*columns*). Szerokość każdej z kolumn określana jest liczbą, a suma wszystkich w rzędzie powinna wynosić 12. Framework korzysta z języka *JavaScript*.

2.2.6 NodeJS 8

NodeJS jest środowiskiem uruchomieniowym stosowanym do tworzenia wysoce skalowalnych aplikacji internetowych, a w szczególności serwerów WWW napisanych w języku *JavaScript*, dzięki czemu zyskał on sporą popularność. Wykorzystuje on asynchroniczny system wejścia-wyjścia i składa się z silnika stworzonego przez firmę *Google*. *NodeJS* umożliwia sprawne zarządzanie bibliotekami i ich zależnościami poprzez dostarczane oprogramowanie NPM (*Node Package Manager*). Dodatkowo dzięki niemu możemy uruchamiać aplikacje napisane w *JavaScript* na urządzeniu IoT (*Internet of Things*) co jeszcze bardziej potęguje jego skalowalność i powszechność. Obecnie środowisko to używane jest przez światowych gigantów takich jak *Google*, *Microsoft*, *Amazon* czy *Netflix*. Ma swoje podstawy w języku C++ co daje możliwość zapewnienia odpowiedniego poziomu stabilności oraz szybkości pisanych aplikacji.

2.2.7 Express.js

Express.js jest frameworkiem typu *open source* przeznaczonym do współpracy z *Node.js*. Jest zaprojektowany do tworzenia aplikacji internetowych oraz przeznaczonych dla nich API (*Application Programming Interface*). Jest standardem typu *de facto* zastosowań serwerowych bazujących na *Node.js*. *Express.js* pozwala definiować tabele routingu w celu wykonywania różnego typu ak-

cji w oparciu o metodę HTTP. Umożliwia ponadto dynamiczne renderowanie stron napisanych w języku HTML w oparciu o przekazywanie argumentów do szablonów.

2.2.8 Karma

Narzędzie zbudowane w oparciu o serwer *NodeJS* oraz technologię *Socket.io*, służące do automatycznego uruchamiania testów tworzonych w języku *JavaScript* w emulowanym środowisku przeglądarek internetowych. Za pomocą *Karma* możemy uruchamiać testy na różnych środowiskach programistycznych - deweloperskim, produkcyjnym czy testowym. Przy użyciu narzędzia *Istanbul* możliwy jest dostęp do informacji na temat pokrycia implementowanego kodu testami. *Karma* jest pełnoprawnym środowiskiem testowym ułatwiającym szybkie i bezproblemowe testowanie kodu *JavaScript*.

2.2.9 Jasmine

Framework typu *behavior-driven development framework* dający programistom wiele przydatnych funkcji potrzebnych do testowania oprogramowania. Jest zintegrowany ze środowiskiem *Karma* i pozwala na pisanie testów oprogramowania w sposób opisowy. Nie jest on zależny od środowiska *JavaScript* i nie wymaga drzewa DOM (*Document Object Model*). Jasmine pozwala nie tylko na pisanie testów jednostkowych, ale także testów typu *e2e* (*end-to-end*).

2.2.10 Apache Cordova

Cordova to w skrócie API (*Application Programming Interface*) umożliwiające stworzenie natywnej aplikacji używając wyłącznie HTML, CSS oraz kodu *JavaScript* przy jednoczesnym dostępie do komponentów urządzeń mobilnych, takich jak aparat, usługi geolokalizacji czy nawet książki kontaktów. Aplikacje stworzone w ten sposób mogą znaleźć się na urządzeniach mobilnych producentów najbardziej wiodących firm - *iOS*, *Android*, *Windows Phone* czy *BlackBerry*.

2.2.11 LaTeX

Oprogramowanie służące do tworzenia przejrzystych dokumentów tekstowych takich jak na przykład książki czy artykuły. Docelowym plikiem wyjściowym jest najczęściej plik w formacie PDF (*Portable Document Format*). Cechą charakterystyczną jest tutaj fakt, że *LaTeX* ma swój własny język programowania, za pomocą którego tworzone są dokumenty. Do wygenerowania dokumentów przydatne są narzędzia przetwarzające pliki źródłowe i generujące dokumenty wyjściowe. *LaTeX* oferuje dostęp do szeregu pakietów umożliwiających znacznie prostsze i szybsze implementowanie bardziej złożonych elementów plików wyjściowych. Filozofia *LaTeXa* zakłada, aby skupiać się nie na tym jak dokument ma wyglądać, a co ma zawierać. Do użytkownika należy tylko wprowadzenie struktury i zawartości dokumentu.

2.2.12 PBKDF2

PBKDF2 (*Password-Based Key Derivation Function 2*) jest popularnym algorytmem podobnym do algorytmu *BCrypt*, zapewniającym porównywalny stopień bezpieczeństwa. PBKDF2 jest bezpieczną funkcją skrótu stosowaną w celach zabezpieczania bezprzewodowych sieci WiFi (WPA, WPA2). Algorytm ten w celu wygenerowania klucza pochodnego wykorzystuje pseudolosową funkcję, taką jak HMAC (*Hash Message Authentication Code*) powtarzając operacje hashowania przez określoną, dużą liczbę iteracji. Algorytm PBKDF2 jest trudny do złamania za pomocą CPU (*Central Processing Unit*), ale nie wymaga zbyt dużych zasobów pamięciowych i łatwo jest go zrównoleglić za pomocą GPU (*Graphics Processing Unit*). Mimo to, że jest on wciąż stosowany to nie zaleca się stosowania go do nowych projektów.

2.2.13 Argon2

Algorytm *Argon2* został zwycięzcą *Password Hashing Competition* i jako następca *BCrypt* oraz *Scrypt* jest obecnie zalecany do zabezpieczania haseł. *Argon2* zawiera szereg zabezpieczeń przeciwko atakom typu *Brute Force*. W przeciwieństwie do *PBKDF2* algorytm ten wprowadza silną odporność nie tylko na ataki przy użyciu CPU, ale także GPU (wykorzystuje się konkretną

odmianę *Argon2d*). Samo użycie algorytmu jest niezwykle proste. WIele języków programowania oferuje dedykowane bilbioteki, które znacząco ułatwiają używanie *Argon2* w implementacjach.

2.3 Narzędzia

2.3.1 Visual Studio Code 1.28

Visual Studio Code jest wieloplatformowym, prostym w obsłudze IDE (*Integrated Development Environment*) stworzonym przez firmę *Microsoft*. Łączy on w sobie prostotę edytora kodu źródłowego z potężnym środowiskiem deweloperskim oferującym mechanizm *IntelliSense* czy mechanizm debugowania kodu. *Visual Studio Code* oferuje szereg skrótów klawiszowych i snippetów ułatwiających szybsze i bardziej intuicyjne implementowanie oprogramowania. Autorzy udostępnili mnóstwo udogodnień wspierających pracę w zespole, takich jak integracja z systemem kontroli wersji *Git* czy też rozproszone współdzielenie kodu. Oprogramowanie połączone jest bezpośrednio z uaktualnianym na bieżąco repozytorium paczek i pakietów ułatwiających programowanie w danym języku i technologii. Dodatkowo, *VSCode* oferuje przyjazne GUI (*Graphical User Interface*), z przejrzystym eksploratorem drzewa plików i mapą zawartości pliku, ułatwiającą szybsze wykrycie błędów i ostrzeżeń w kodzie [Vsc].

2.3.2 github.com

Serwis internetowy stworzony dla projektów programistycznych, który wykorzystuje system kontroli wersji *Git*. Jego implementacja ma podłoże w języku *Erlang* z wykorzystaniem frameworka *Ruby on Rails*. Github umożliwia darmowy hosting plików oraz płatne, prywatne repozytoria. Platforma oferuje szereg statystyk powiązanych z implementowanym kodem źródłowym, mechanizm typu *bugtracker*, możlwiość pobierania repozytoriów, rozgałęziania (dzielenia) pracy pomiędzy członków zespołu programistycznego czy późniejszego ich łączenia. Dodatkowo serwis ten oferuje usługę zwaną *Github Pages* służącą do szybkiego tworzenia stron internetowych kompilowanych na podstawie kodu zawartego w repozytorium. Dzięki *github.com* mamy możliwość bezpośredniej kontroli nad tworzonym kodem oprogramowania i dostęp do historii pracy co znacząco ułatwia zarządzanie projektem programistycznym.

2.3.3 Auth0

Serwis *Auth0* umożliwia połączenie z dowolną aplikacją w celu zaoferowania usług uwierzytelniania użytkowników. Oferuje on metody logowania i rejestracji w serwisie za pomocą tradycyjnego loginu (adresu e-mail) i hasła, bądź mediów społecznościowych takich jak *Google*, *Facebook* czy *Twitter*. Domyślny protokół używany do integracji serwisu *Auth0* z aplikacją użytkownika i pózniejszego uwierzytelniania to OIDC (*OpenID Connect*). Używa on prostych *tokenów* identyfikacyjnych w formacie JSON (*JavaScript Object Notation*). Wymiana danych odbywa się przy użyciu JWT (*JSON Web Token*), który zawiera wszystkie dane identyfikacyjne użytkownika [Aut].

2.3.4 TexStudio

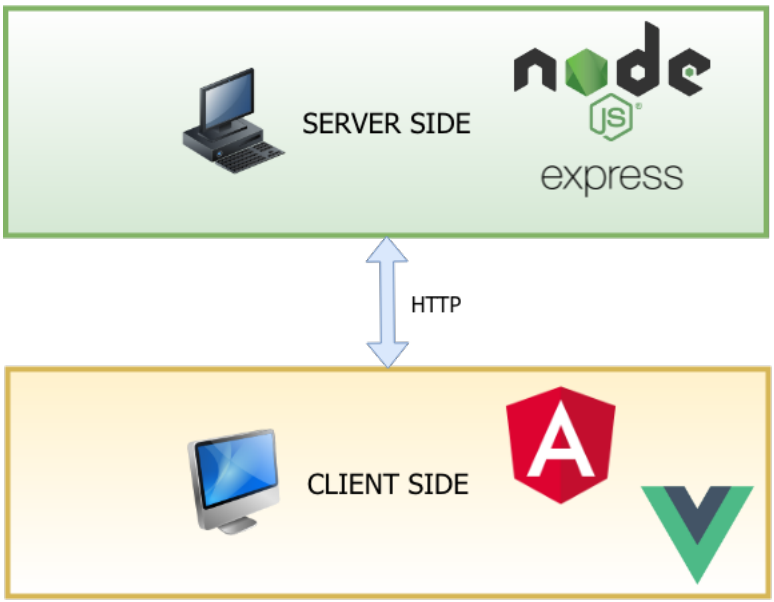
TexStudio jest zintegrowanym środowiskiem służącym do tworzenia dokumentów w języku *LaTeX*. Program posiada szereg funkcji mających na celu ułatwienie tworzenia tekstów, a w ich skład wchodzą między innymi podświetlanie składni, zintegrowana przeglądarka, system sprawdzania referencji czy zintegrowane, zewnętrzne repozytorium pakietów i rozszerzeń języka *LaTeX*.

Rozdział 3

Node.js

3.1 Wprowadzenie

W poniższym rozdziale przedstawione zostaną podstawowe aspekty użycia frameworka *Node.js* z wykorzystaniem zestawu bibliotek oferowanych przez *Express.js* oraz sposób implementacji i analiza działania prostej aplikacji serwerowej, której zadaniem będzie prezentacja uzyskanych wyników. Aplikacja serwerowa pracuje na środowisku lokalnym, na porcie 9000. Wiąże się to z koniecznością przekierowywania wszelkich aplikacji klienckich na ten właśnie port. Dane przechowywane są w plikach dołączonych do projektu. Ogólny schemat aplikacji klient-serwer przedstawiony został na rysunku 31.



Rysunek 31: Ogólny schemat działania aplikacji typu klient-serwer bazującej na serwerze *Node.js* oraz nowoczesnych frameworkach implementacji warstwy prezentacji

3.2 Implementacja aplikacji serwerowej

W poniższym rozdziale przedstawiony zostanie sposób implementacji aplikacji serwerowej bazującej na zestawie bibliotek *Node.js* oraz *Express.js*.

3.2.1 Inicjalizacja serwera

Inicjalizacja serwera rozpoczyna się od wygenerowania klucza i certyfikatu SSL (*Secure Socket Layer*), który używany jest przez protokół HTTPS w celu zwiększenia bezpieczeństwa danych.

Kolejnym krokiem jest zdefiniowanie portu, na którym będzie on nasłuchiwał żądań klienta i rozpoczęcia nasłuchiwania. W tym celu posłużono się frameworkiem *Express.js* i zdefiniowano lokalny serwer HTTPS pracujący na porcie numer 9000.

W celu wygenerowania certyfikatu i klucza RSA (*Rivest-Shamir-Adleman*) posłużono się poniższą komendą:

```
openssl req -newkey rsa:2048 -new -nodes -keyout key.pem -out cert.pem
```

Wygenerowany klucz RSA zawiera 2048 bitów. Rozmiar ten wybrany został z kilku powodów. Przede wszystkim wiele urządzeń nie wspiera większych wartości bitowych. Ponadto, użycie tego klucza w trakcie szyfrowania i uwierzytelniania powoduje znacznie mniejsze zużycie procesora.

Sama, wstępna implementacja serwera HTTPS i przekazanie wygenerowanych plików *key.pem* oraz *cert.pem* do parametrów inicjalizacji serwera zaprezentowana została na poniższych blokach kodu źródłowego:

```
const httpsServer = https.createServer({
key: fs.readFileSync('key.pem'),
cert: fs.readFileSync('cert.pem')
}, app);

httpServer = app.listen(9000, () => {
console.log("HTTP Server running at https://localhost:" +
httpServer.address().port);
});
```

W następnym kroku zdefiniowano komendę *start-server*, za pomocą której uruchamiany będzie serwer. Definicja komendy znajduje się w pliku JSON służącym do zarządzania lokalnymi paczkami NPM (*Node Package Module*) - *package.json*. W celu uruchomienia serwera wystarczy użyć polecenia *npm run start-server*.

```
"start-server": "./node_modules/.bin/ts-node ./server/server.ts --secure",
```

Po uruchomieniu serwera z poziomu konsoli otrzymano komunikat widoczny na rysunku 32.

```
HTTPS Secure Server running at https://localhost:9000
[nodemon] restarting due to changes...
[nodemon] starting `npm run start-server`

> angular-security@0.0.0 start-server /Users/krzysztoffigiel/Desktop/Praca Magis
terska/Angular/Angular-Security/angular-security
> ts-node ./server/server.ts --secure
```

Rysunek 32: Zrzut ekranu konsoli systemu *MacOS* ukazujący komunikaty o poprawnym uruchomieniu procesu nasłuchiwania przez serwer na porcie 9000

Definicja tabel routingu w *Express.js* opiera się na użyciu funkcji `app.route(path)`. W aplikacji serwerowej zdefiniowano następujące ścieżki routingu:

- */api/books*
- */api/signup*
- */api/user*
- */api/logout*
- */api/login*

Przykładowa definicja wpisu tabeli routingu zaprezentowana została na poniższym bloku kodu źródłowego:

```
app.route('/api/books')
.get(readAllBooks);
```

3.2.2 Implementacja modułu rejestracji nowego użytkownika

W następnym kroku zdefiniowano metody odpowiedzialne za obsługę rejestracji nowego użytkownika w aplikacji. Wpis tabeli routingu zawierający ścieżkę do tworzenia nowego użytkownika wygląda następująco:

```
app.route('/api/signup')
.post(createUser);
```

Po zdefiniowaniu wpisu przystąpiono do implementacji metody `createUser(req, res)`. Sama implementacja tej metody opiera się na odebraniu parametrów przesyłanych przez użytkownika w obiekcie `req` i przekazania odpowiednich statusów metody HTTP do klienta (`res`) w zależności czy weryfikacja nowego użytkownika przebiegła pomyślnie czy też nie.

Do walidacji haseł użytkownika posłużono się walidatorem z repozytorium NPM - *password-validator*. Umożliwia on zdefiniowanie reguł tworzenia nowych haseł oraz dodawanie haseł, które znaleźć się mają na tzw. *blacklist* czyli liście haseł niedopuszczalnych, powszechnie używanych. W celu zachowania bezpieczeństwa systemu zdefiniowano reguły i przykład *blacklist* tworzenia nowych haseł przez użytkowników:

```
schema
.is().min(10)                                    // Minimum length 10
.has().uppercase()                               // Must have uppercase letters
.has().lowercase()                               // Must have lowercase letters
.has().digits()                                  // Must have digits
.has().not().spaces()                            // Should not have spaces
.is().not().oneOf(['Passw0rd', 'Password123']); // Blacklist these values
```

Po pomyślnej walidacji przesyłanego przez użytkownika w obiekcie `req` hasła serwer odpowiada komunikatem 200 przesyłając w parametrze `body` dane (*id* oraz *email*) nowego użytkownika. W tym celu posłużono się metodą `res.status(httpCode)`:

```
res.status(200).json({ id: user.id, email: user.email });
```

W przypadku podania hasła nieazgodnego z przyjętymi regułami tworzenia nowych haseł użytkownika serwer odpowiada komunikatem 400 *Bad request*. Wykorzystano metodę `res.status(httpCode)`, a jako zwracany parametr podano listę błędów zwracaną przez pakiet *password-validator*:

```
res.status(400).json({ errors });
```

Jeżeli serwer napotkał wewnętrzny błąd, wówczas wysyła on komunikat o kodzie 500 *Internal Server Error* za pomocą metody `res.sendStatus(httpCode)` bez przekazywania treści błędu w celu zachowania odpowiedniej hermetyzacji aplikacji:

```
res.sendStatus(500);
```

Nowi użytkownicy przechowywani są w bazie danych za pomocą obiektu składającego się z trzech wartości:

```
const user: DbUser = {
id,
email,
passwordDigest
};
```

Przy tworzeniu nowego użytkownika sprawdzany jest dodatkowo warunek czy nie ma już podobnego konta w bazie danych:

```
const usersPerEmail = _.keyBy(_.values(USERS), 'email');

if (usersPerEmail[email]) {
const message = 'User already exists with assigned email address: ' + email;
console.error(message);
throw new Error(message);
}
```

Jeżeli użytkownik istnieje w bazie danych wyświetlany jest stosowny komunikat. W przypadku braku podobnego konta i pomyślnej weryfikacji tworzony i zwracany jest nowy obiekt `user`:

```
this.userCounter++;

const id = this.userCounter++;

const user: DbUser = {
id,
email,
passwordDigest
};

USERS[id] = user;

return user;
```

Hasła użytkownika przechowywane są w bazie danych za pomocą funkcji skrótu *Argon2*. W tym celu wykorzystano specjalny pakiet oferowany przez repozytorium NPM. Hasło podane przez użytkownika przekazywane jest jako parametr funkcji `argon2.hash(password)`. Więcej informacji na temat wykorzystania funkcji *Argon2* zawarte zostało w rozdziale dotyczącym bezpieczeństwa.

3.2.3 Implementacja modułu logowania użytkownika

3.2.4 Implementacja modułu wylogowywania użytkownika

3.2.5 Implementacja mechanizmów podtrzymywania sesji użytkownika

Rozdział 4

Mechanizmy bezpieczeństwa aplikacji internetowych opartych na języku JavaScript

4.1 Wprowadzenie

W poniższym rozdziale przedstawiono najważniejsze mechanizmy bezpieczeństwa wykorzystywane w aplikacjach internetowych opartych na języku *JavaScript*. Działanie mechanizmów poparto przykładami pochodzącymi z aplikacji internetowej napisanej w ramach pracy magisterskiej.

4.1.1 Komunikacja klient-serwer przy użyciu protokołu HTTPS

Protokół HTTPS (*Hypertext Transfer Protocol Secure*) jest szyfrowaną wersją protokołu HTTP (*Hypertext Transfer Protocol*). Komunikacja nie jest oparta bezpośrednio na modelu klient-serwer, tylko na zasadzie szyfrowanego protokołu SSL (*Secure Socket Layer*). Dzięki takiemu rozwiązaniu nie ma możliwości przejęcia czy też naruszenia integralności danych przesyłanych w trakcie transmisji. SSL wykorzystuje tzw. certyfikaty mające na celu poświadczenie wiarygodności domeny, bądź domeny oraz jej właściciela. Dzięki temu użytkownik, który korzysta ze strony WWW ma pewność, że przesyłane informacje nie zostaną przechwycone i nie trafią w nieporządane ręce. W momencie nawiązywania połączenia przez przeglądarkę internetową zabezpieczoną protokołem SSL następuje ustalenie odpowiednich algorytmów oraz kluczy szyfrujących, stosowanych następnie do wymiany danych między przeglądarką a serwerem WWW. Wykorzystywana jest tutaj kryptografia asymetryczna o ustalonej długości klucza.

Inicjalizacja serwera w aplikacji rozpoczyna się od wygenerowania klucza i certyfikatu SSL. Można to wykonać za pomocą następującej komendy:

```
openssl req -newkey rsa:2048 -new -nodes -keyout key.pem -out cert.pem
```

Taka komenda generuje nowy certyfikat oraz klucz prywatny RSA (o rozmiarze podanym w bitach) i zapisuje je do plików *key.pem* oraz *cert.pem*. W tym przypadku jest to 2048 bitowy klucz. Zawartość tych plików zostanie przekazana jako parametr metody `createServer([options], [requestListener])` pakietu NPM o nazwie *https* służącej do zainicjalizowania serwera nasłuchującego na wybranym porcie. Powodem wybrania 2048 bitowego klucza był fakt, iż niektóre urządzenia nie większych wartości bitowych. Ponadto, wykorzystanie tych kluczy podczas szyfrowania i uwierzytelniania pozwala na znacznie mniejsze zużycie procesora.

Zawartość wygenerowanych plików przedstawiono na rysunku 42.

Wygenerowany certyfikat

```
-----BEGIN CERTIFICATE-----
MIIEzTCCA7WgAwIBAgIJAPjE2PVe18vsMA0GCSqGSIb3DQEBBQUAMIGfMQswCQYD
VQQGEwJCRTEQMA4GA1UECBMHQmVsZ2l1bTEQMA4GA1UEBxMHQmVsZ2l1bTEUMBIG
A1UEChMLQW5ndWxhclVuaXYxFDASBgNVBAsTC0FuZ3VsYXJVbmlZMRQwEgYDVQQD
EwtBbmd1bGFyVW5pdjEqMCgGCSqGSIb3DQEJARYbYWRtaW5AYW5ndWxhci11bml2
ZXJzaXR5LmlvMB4XDTE3MDcxODA4NTkyMVoXDTIwMDcxNzA4NTkyMVowgZ8xCzAJ
BgNVBAYTAkJFMRAwDgYDVQQIEwdCZWxnaXVtMRAwDgYDVQQHEwdCZWxnaXVtMRQw
EgYDVQQKEwtBbmd1bGFyVW5pdjEUMBIGA1UECxMLQW5ndWxhclVuaXYxFDASBgNV
BAMTC0FuZ3VsYXJVbml2MSowKAYJKoZIhvcNAQkBFhthZG1pbkBhbmd1bGFyLXVu
aXZlcnNpdHkuaWBwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDOY2DB
Yjc3IU4Y8AiVHoYnsXMvYCWVjHgaB30iH5VrSKNkRBHXr7uaafUYcmz5rEDjTgaM
l19rF5t6sC1bszDA9H0rz1K6TlwbwkCRpcVVk1tr4rfKyBvh810phDuiDjvM1E0C
O2LopdDJTCbwtPV2/5s36qUdAZp8967JfRII/67uxpCuIKzMVYYD10WFwfjRLuzC
MUN9NCdqjz17DNcumirwMEYbO0/dqHaNrX7EBISToidUfI34iWaB9VIW0/y7gzaP
MrplmdfRho8XMA+Dglx7ucgNqnvfij7MOW3p/UuaeYazvq2K5bNU8dDxPrDXUqiL
19FB9I4ObP0rGYBlAgMBAAGjggEIMIIBBDAdBgNVHQ4EFgQUAbMfTxk25XWoQkxl
GiiMFsNEwXMwgdQGA1UdIwSBzDCByYAUAbMfTxk25XWoQkxlGiiMFsNEwXOhgaWk
gaIwgZ8xCzAJBgNVBAYTAkJFMRAwDgYDVQQIEwdCZWxnaXVtMRAwDgYDVQQHEwdC
ZWxnaXVtMRQwEgYDVQQKEwtBbmd1bGFyVW5pdjEUMBIGA1UECxMLQW5ndWxhclVu
aXYxFDASBgNVBAMTC0FuZ3VsYXJVbml2MSowKAYJKoZIhvcNAQkBFhthZG1pbkBh
bmd1bGFyLXVuaXZlcnNpdHkuaW+CCQD4xNj1XtfL7DAMBgNVHRMEBTADAQH/MA0G
CSqGSIb3DQEBBQUAA4IBAQBrYsSH/yG0uaf1ZEG/Bh1cUEgWoYJMh5Kq/TzNur5r
3JnManXN6dDvj/hny2AwCf4QKuT+li+eUXw/U46SRAmWg5I0aeyGyQJFepSU/dYP
pHeHf4mf2vmt0qR+0/EijNYsDfxysFxW7C9CnKjCeJKWLuH4vpcuBdPSgyuddKz/
URTqX8PoYhJacI55Mq2ypcGpHtuoN2Rx2idj/DjVoSc3UbIaU6mjyuZnLac8K3jW
BrAArZgQ54Lx9pKc1GT3fuLOvJ7n+dEBp1CUgkrp3vLEbA2O0EyvhsvtigR6uVK/
55G6tCZcoJolHB2Dh2vBaNgZKbsYoLu5zyVxPcb/Fs/H
-----END CERTIFICATE-----
```

Wygenerowany klucz prywatny RSA

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAzmNjvGI3NyFOGPAIlR6GJ7FzL2AllYx4Ggd9In+Va0ijZEQR
16+7mnn1GHJs+axA404GjJdfaxeberAtW7MwwPR9K89Suk5cG8JAkaXFVZNba+K3
ysgb4fNdKYQ7og47zNRNAjti6KXQyUwm8LT1dv+bN+qlHQGaQfeuyX0SCP+u7saQ
riCszFwGA9dFhcH40S7swjFDfTQnao89ewzXLpoq8D8GGztP3ah2ja1+xASEk6In
VHyN+IlmgfVSFtPBu4M2jzK6ZZnX0YaPFzAPg4Jce7nIDap734o+zDlt6f1LmnmG
s76tiuWzVPHQ8T6w11II19fRQfSODmzzqxmAdQIDAQABAoIBACPUCy9b2Xv/kyAv
Iz5H00A2xOk3hq7EeOED5cwzsWuza3Hvgy1vO2Ed/cdlrddV+Nm85NQpPw5HRd1x
kh42KEDG7efaEOF40vKasTzXiz9E2n0LGZ7hSD3cUpLveq6yS91OCaRWjOAYUPtw
9d2ZfHdQ/jAh0tILQi0rgzNFPAlFkYGrq6/7P+FG//F99XV8KSiF/libpIkyx0JF
u4kajF/XprhExr1wcwkQaLsmi0ispvOE9Ge2QNyRV4b5mvLvpQndla7MxusItegO
sMU1HpMr0/cTGblpYbOkad2AqXR/yQ9x4IIdrv6+vl52GYMuXwxCjeJVyCB0sTXq
9VcZs6ECgYEA/CJ/NV8ZfQi0zpzOApIQv4cSk1WZh7LRzYw1EkOgwtIcnCVs0rnB
32SZblkzmisbJhayqsEUuU4bp5mVeI3vfdnHE8rP73onPhOnD/LvLkSuNbJfqTyW
yzP7QG71pWYrSBaIMchQ5XyZVebclym2/LF1xM5QTvaADpWfL4JsG70CgYEA0Y1c
RmmgVclrLiePwRur37IKuBmRvrJj9fusAt/6tL5Tmi377etWSADpuqbvbmRN79gX
YjBiThDImtSjOJdwDP1xWdHanqQUxT4NBWkcUdvs5m8v2DIC2z4WA5BmHx6l5euy
3lYxvq+BVL1j6gcTucJSkw+QEatQ0Wik1IQlJxkCgYAhrq4VwC2UgXw9+b6fo0lr
Fq1Jl27aLO5QKkHQvc4VGKKPzYQbJouav0Kr9iipmMaYsQGPRBnzo1aYbjEEmNl8
k4I/x6E9mbDyXwe7XrXLFYu8Sqby4dA3NyuCcRBJ1ztC9xecpuj/r0SAsEkZwA3c
9gwIjZLyg4WZ7S0F2MN+dQKBgBUzz11UfKOR2EFgQVxRl29XzVw4xeVcmV6fK0VT
MkOtajoUYwQsppsBdGmtUFaEEZqt65fEu32YlyOLVkR4lh7KWfizkgIxKeTXJWwh
uXrSJMJoB5hBXyadvo53ws1tRW3hDkUOxZO8oEJZs01PkABgmAhHL9WJlvCJPPw/
rp6pAoGBAIZ6BHz/nsJiODrEj4ujlSV+02w0rd2MP+2F6wLxIGJ7Lt0dUxEDc1lK
J76OWdEASR+zvkN9HO7trKqshQBljKK7J9TcjX1gie9zloRDeLMtwlH6Qmw+N/c9
oPlXT1oaMejB/9Wtii1rFsM/X+WvkohHbHquO8ysIlffQ5R1f3Ti
-----END RSA PRIVATE KEY-----
```

Rysunek 41: Zdjęcie przedstawiające wygenerowany za pomocą komendy *openssl -newkey* certyfikat SSL oraz klucz prywatny RSA.

Na rysunku poniżej przedstawiono ogólny schemat działania protokołu HTTPS z wykorzystaniem protokołu SSL podczas wymiany zaszyfrowanych danych między klientem a serwerem:

Rysunek 42: Schemat działania protokołu HTTPS wykorzystującego protokół SSL w trakcie wymiany zaszyfrowanych danych między klientem a serwerem

4.1.2 Zasady tworzenia haseł użytkowników

Hasła są obecnie jedną z najpopularniejszych technik uwierzytelniania w Internecie. Jest to metoda bardzo bezpieczna pod warunkiem spełnienia określonych kryteriów. Hasło użytkownika powinno charakteryzować się tym iż nie jest ono ciągiem składającym się z mniej niż dziesięciu znaków. Dzięki takiemu rozwiązaniu liczba możliwych kombinacji (na przykład przy ataku słownikowym) diametralnie wzrasta. Kolejną kwestią jest wykorzystywanie przez użytkowników różnych typów znaków alfanumerycznych - dużych i małych liter, znaków specjalnych oraz cyfr. Znacznie podnosi to poziom skomplikowania zabezpieczenia.

Twórcy aplikacji internetowych mają dostęp do szeregu bibliotek i pakietów umożliwiających definiowanie reguł tworzenia bezpiecznych haseł użytkowników. W projekcie wykorzystano pakiet NPM o nazwie *password-validator*. Pozwala on na proste definiowanie zasad kreowania haseł i dostarcza mechanizmy weryfikacji czy powyższe zasady zostały spełnione.

W przykładowym projekcie posłużono się następującymi definicjami reguł:

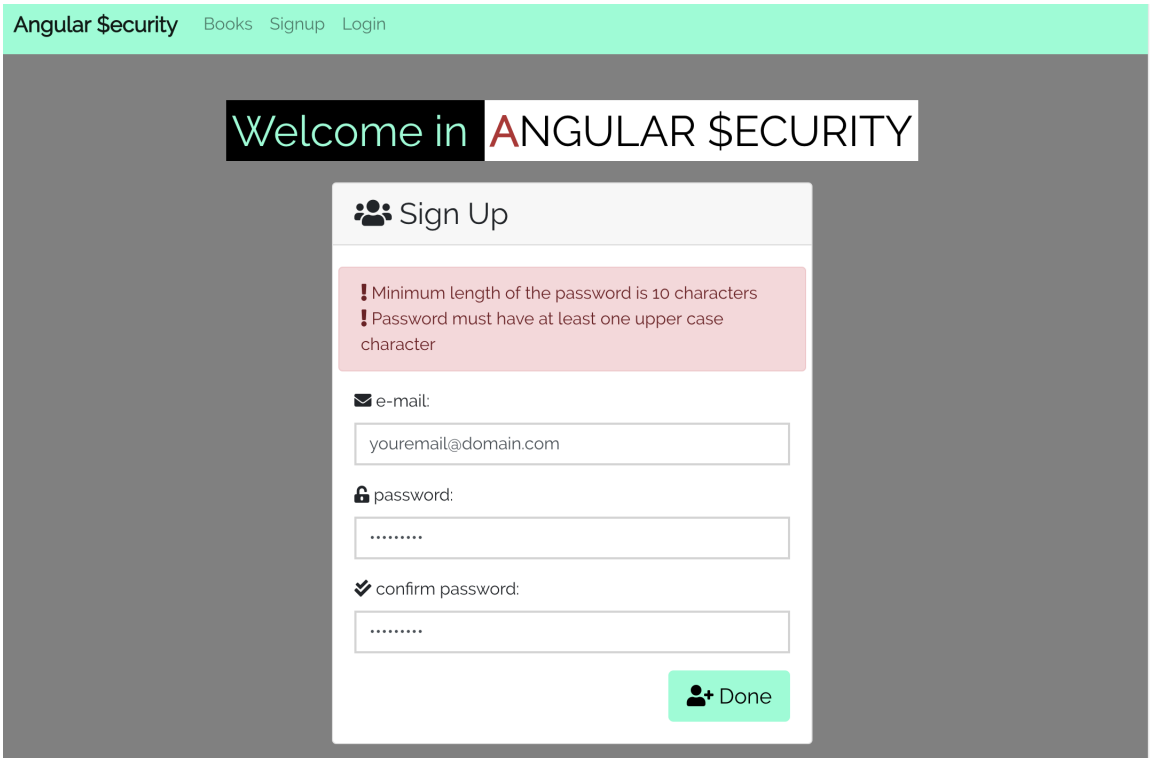
```
schema
```

```
.is().min(10)                                    // Minimum length 10
.has().uppercase()                               // Must have uppercase letters
.has().lowercase()                               // Must have lowercase letters
.has().digits()                                  // Must have digits
.has().not().spaces()                            // Should not have spaces
.is().not().oneOf(['Passw0rd', 'Password123', '123456', '123456789', 'qwerty']);
// Blacklist these values
```

Jak widać, istnieje również możliwość zdefiniowania tzw. *blacklisty* zawierającej najczęściej używane hasła, będące łatwym elementem do wykorzystania podczas na przykład ataku słownikowego. Do walidacji poprawności przesyłanych przez użytkownika haseł wykorzystano metodę oferowaną przez pakiet *password-validator*:

```
schema.validate(password, options?)
```

W przypadku wpisania przez użytkownika hasła nie spełniającego wymagań zdeklarowanych w poniższym schemacie następuje walidacja formularza rejestracji i powiadomienie użytkownika jakie błędy popełnił:



Rysunek 43: Ekran aplikacji pokazowej prezentujący błędną walidację formularza w przypadku podania przez użytkownika haseł, które nie spełniają wymagań zdefiniowanych w schemacie bezpiecznego hasła.

4.1.3 Sposób przechowywania haseł użytkowników

Hasła użytkowników aplikacji przechowywane są w bazie danych w postaci jednokierunkowej funkcji skrótu *Argon2*, która jest zwycięzcą konkursu *Password Hashing Competition* z połowy 2015 roku. Funkcja ta jest rekomendowana przez OWASP (*Open Web Application Security Project*). Jest to globalna, profesjonalna fundacja, która działa charytatywnie. OWASP jest otwarty dla każdego, kto interesuje się bezpieczeństwem nowoczesnych aplikacji internetowych. Organizacja działa na rzecz publikowania artykułów, metodologii, narzędzi i dokumentacji związanych z ochroną danych.

Funkcja *Argon2* może być używana do mieszania haseł, przechowywania danych uwierzytelniających, pochodnych kluczy lub innych aplikacji. W przeciwieństwie do funkcji *bcrypt* czy *PBKDF2*,

Argon2 cechuje się wysoką adaptacyjnością. Możliwe jest tu bowiem ustawienie aż trzech parametrów: liczby iteracji wykonywanych operacji kryptograficznych, wielkości wykorzystywanej pamięci oraz zrównoleglenia, czyli liczby równoległych wątków działających w tle. Ma ona bardzo prostą konstrukcję umożliwiającą mniejsze zużycie pamięci i efektywne wykorzystanie wielu jednostek obliczeniowych w celu przeprowadzania działań kryptograficznych. Jedną z ciekawych funkcji jest również możliwość sprecyzowania dodawanej soli, długości wyjściowego skrótu czy też możliwość dodania dodatkowego, opcjonalnego klucza. *Argon2* posiada trzy odmiany: *Argon2i*, *Argon2d* oraz *Argon2id*. Funkcja *Argon2d* jest znacznie szybsza od pozostałych i jednocześnie odporna na ataki typu *GPU Cracking*. *Argon2i* charakteryzuje się znacznie większą liczbą odwołań do pamięci co powoduje ją wolniejszą od pozostałych. Zaleca się używanie jej do hashowania haseł. Ostatnia z odmian - *Argon2id* - jest hybrydą dwóch wersji: *Argon2i* oraz *Argon2d*. Czyni ją to odporną na ataki typu *side-channel* i zwiększa jej bezpieczeństwo na metody naruszania zabezpieczeń z użyciem kart graficznych. Ogólna, bardzo uproszczona postać funkcji *Argon2* wygląda następująco:

```
ARGON2(password, salt, parallelism, length, memory, count, key)
```

Jak widać, oprócz przekazania takich parametrów jak hasło czy dodawana sól istnieje równiez możliwość oznaczenia poziomu zrównoleglenia obliczeń (`parallelism`), określenia ilości pamięci niezbędnej do obliczenia skrótu (`memory`), określenia opcjonalnego klucza (`key`) czy też wielkości ciągu wyjściowego (`count`). Cały algorytm oparty jest o kryptograficzną funkcję skrótu *BLAKE2*, która jest jednym z finalistów konkursu *SHA-3*.

W autorskiej aplikacji wykorzystano pakiet NPM o nazwie *node-argon2*. Oferuje on większość możliwości, które dostarcza funkcja *argon2*. Funkcja hashowania hasła użytkownika wygląda następująco:

```
async function createUserAndSession(res: Response, credentials) {
const passwordDigest = await argon2.hash(credentials.password);
...
}
```

Jak widać jako parametr funkcji przekazywane jest hasło podane przez użytkownika w trackie rejestracji do serwisu. Rezultat hashowania hasła widoczny jest na zdjęciu 44.

```
HTTPS Secure Server running at https://localhost:9000
Users:  { '1':
   { id: 1,
     email: 'testemail@email.com',
     passwordDigest: '$argon2i$v=19$m=4096,t=3,p=1$VEPI2GGbiaojCgdmU63/RA$vTy8KD
9/v1X8Umm5FIiamRcw9SgVubvBIa5pHkRuCiA' } }
sessionId:  f64ebf9b348659345d6ebd1992fec55ed9ef13f12ccbc1c04ab0671bb3d4cbda
```

Rysunek 44: Okno konsoli serwera po pomyślnej rejestracji nowego użytkownika z wyświetlonym skrótem hasła podanego przez osobę rejestrującą się do serwisu..

Funkcja `argon2.hash(password, options)` umożliwia określenie rodzaju funkcji *argon2*. Wystarczy w obiekcie options przekazać wartość `type: typeOfArgonFunction`. Przykładowy rezultat hashowania hasła *PAssw0rd12* oraz kod źródłowy przedstawione zostały poniżej:

```
argon2.hash(password, {type: argon2.argon2id}).then(hash => {
console.log('Argon2id password: ', hash);
}, (err) => {
console.err('An error occured while password hashing: ', err);
});
```

Pakiet *node-argon2* oferuje również możliwość weryfikowania hasła. Służy do tego metoda `argon2.verify('<big long hash>', 'password')`. Rezultat weryfikacji hasła *PAssw0rd12* widoczny jest na zdjęciu 46.

W momencie podstawienia innego skrótu (w tym przypadku skrótu wygenerowanego przez funkcję *argon2i*) metoda program informuje użytkownika o błędnym dopasowaniu hasła.

```
> angular-security@0.0.0 hash /Users/krzysztoffigiel/Desktop/Praca Magisterska/A
ngular/Angular-Security/angular-security
> node ./demos/hash.js

The result of "PAssw0rd12" password hashing is:
 $argon2id$v=19$m=4096,t=3,p=1$EDu5kEXhEm8MsGKQAPmnVg$V0sR8ALPZsCypMlSwZBvI95mR1
KsJv57RTRLnsDICAw
```

Rysunek 45: Rezultat hashowania hasła *PAssw0rd12* za pomocą funkcji skrótu *Argon2id*.

```
The result of "PAssw0rd12" password hashing is:
 $argon2id$v=19$m=4096,t=3,p=1$oXzIWFp/kpPA0VubMExDbw$+0kcsZlzSTxBMEFBKCzRdriVpa
vyVf9xV9xWRR+8rY4

Matching with:  $argon2i$v=19$m=4096,t=3,p=1$VEPI2GGbiaojCgdmU63/RA$vTy8KD9/v1X8
Umm5FIiamRcw9SgVubvBIa5pHkRuCiA ...
Password not match!!!
```

Rysunek 46: Rezultat wykorzystania funkcji sprawdzającej błędne dopasowanie hasła do funkcji skrótu stworzonego za pomocą *Argon2id*.

4.1.4 Mechanizmy podtrzymywania sesji użytkownika

4.1.5 JSON Web Tokens

4.1.6 Metody ochrony aplikacji internetowych przed atakami typu CSRF

4.1.7 Metody ochrony aplikacji internetowych przed atakami typu XSS

4.1.8 Metody ochrony aplikacji internetowych przed atakami typu CSS

4.1.9 Walidacja formularzy

4.1.10 Autoryzacja oparta na rolach użytkowników (RBAC)

4.1.11 Testy aplikacji internetowych z użyciem frameworków Jasmine oraz Karma

Rozdział 5

Baza danych

5.1 Opis tabeli bazy danych

W poniższej sekcji opisane zostaną tabele bazy danych wraz z polami w nich zawartymi. Określony zostanie typ danego pola oraz krótki opis jego przeznaczenia.

5.1.1 Person

Tabela *person* – zawiera dane dotyczące użytkownika aplikacji:

- *id* – pole typu *INT* zawierające unikalny identyfikator pola w bazie danych,
- *name* – pole typu *VARCHAR* przechowujące imię użytkownika,
- *surname* – pole typu *VARCHAR* przechowujące nazwisko użytkownika,
- *pesel* – pole typu *VARCHAR* przechowujące PESEL użytkownika,
- *guardian* – pole typu *VARCHAR* mówiące o tym, czy dany użytkownik posiada opiekuna,
- *street* – pole typu *VARCHAR* przechowujące nazwę ulicy, na której mieszka użytkownik,
- *homeNumber* – pole typu *VARCHAR* przechowujące numer domu, pod którym mieszka użytkownik,
- *city* – pole typu *VARCHAR* przechowujące nazwę miasta, w którym mieszka użytkownik,
- *postalCode* – pole typu *VARCHAR* przechowujące kod pocztowy miasta, w którym mieszka użytkownik,
- *telephoneNumber* – pole typu *VARCHAR* przechowujące numer telefonu użytkownika,
- *actualGlucometer* – pole typu *INT* będące kluczem obcym z tabeli *glucometer*, zawierające *id* aktualnie posiadanego glukometru,
- *previousGlucometer* – pole typu *INT* będące kluczem obcym z tabeli *glucometer*, zawierające *id* poprzedniego glukometru,
- *serialNumber* – pole typu *VARCHAR* przechowujące numer seryjny glukometru, którego aktualnie używa użytkownik.

5.1.2 Medical Data

Tabela *medical_data* – zawiera informacje dotyczące danych medycznych użytkownika:

- *id* – pole typu *INT* zawierające unikalny identyfikator pola w bazie danych,
- *yearOfIllness* – pole typu *TEXT* przechowujące informację na temat roku zachorowania użytkownika na cukrzycę,
- *otherMedicines* – pole typu *VARCHAR* przechowujące informacje na temat innych, przyjmowanych leków,

- *bmi* – pole typu *DECIMAL* przechowujące wartość współczynnika BMI *(Body Mass Index)* użytkownika,
- *height* – pole typu *INT* przechowujące informacje na temat wzrostu użytkownika w cm,
- *weight* – pole typu *VARCHAR* przechowujące informacje na temat wagi użytkownika,
- *hbValue* – pole typu *DECIMAL* przechowujące informacje na temat wartości współczynnika HbA1c użytkownika,
- *diabetesTypeId* – pole typu *INT* będące kluczem obcym z tabeli *diabetes_type*, zawierające *id* typu cukrzycy, na którą cierpi użytkownik,
- *insulinId* – pole typu *INT* będące kluczem obcym z tabeli *insulin*, zawierające *id* typu insuliny, którą zażywa użytkownik,
- *tabetsId* – pole typu *INT* będące kluczem obcym z tabeli *tablets*, zawierające *id* leku, który przyjmuje użytkownik.

5.1.3 Glucometer

Tabela *glucometer* – zawiera informacje dotyczące dostępnych glukometrów w bazie danych:

- *id* – pole typu *INT* zawierające unikalny identyfikator pola w bazie danych,
- *glucometerName* – pole typu *VARCHAR* zawierające nazwę danego glukometru.

5.1.4 Medical Data has Tablets

Tabela *medical_data_has_tablets* – zawiera informacje dotyczące przyjmowanych przez danego użytkownika leków:

- *id* – pole typu *INT* zawierające unikalny identyfikator pola w bazie danych,
- *medicalDataId* – pole typu *INT* będące kluczem obcym z tabeli *medical_data*, zawierające *id* danych medycznych danego użytkownika,
- *tabetsId* – pole typu *INT* będące kluczem obcym z tabeli *tablets*, zawierające *id* leku, który przyjmuje użytkownik.

5.1.5 Tablets

Tabela *tablets* – zawiera informacje dotyczące dostępnych leków w bazie danych:

- *id* – pole typu *INT* zawierające unikalny identyfikator pola w bazie danych,
- *datbletName* – pole typu *VARCHAR* zawierające nazwę danego medykamentu.

5.1.6 Medical Data has Insulin

Tabela *medical_data_hasinsulin* – zawiera informacje dotyczące insuliny przyjmowanej przez danego użytkownika:

- *id* – pole typu *INT* zawierające unikalny identyfikator pola w bazie danych,
- *medicalDataId* – pole typu *INT* będące kluczem obcym z tabeli *medical_data*, zawierające *id* danych medycznych danego użytkownika,
- *insulinId* – pole typu *INT* będące kluczem obcym z tabeli *insulin*, zawierające *id* typu insuliny, którą zażywa użytkownik.

5.1.7 Diabetes Type

Tabela *diabetes_type* – zawiera informacje dostępnych w bazie danych typów cukrzycy:

- *id* – pole typu *INT* zawierające unikalny identyfikator pola w bazie danych,
- *typeName* – pole typu *VARCHAR* zawierające nazwę danego typu cukrzycy.

5.1.8 Insulin

Tabela *insulin* – zawiera informacje dotyczące dostępnych w bazie danych rodzajów insuliny:

- *id* – pole typu *INT* zawierające unikalny identyfikator pola w bazie danych,
- *insulinName* – pole typu *VARCHAR* zawierające nazwę danego typu insuliny.

5.1.9 Measurement

Tabela *measurement* – zawiera informacje dotyczące pomiarów dokonywanych przez użytkowników:

- *id* – pole typu *INT* zawierające unikalny identyfikator pola w bazie danych,
- *date* – pole typu *DATETIME* zawierające datę aktualnie dokonanego przez użytkownika pomiaru,
- *glicemy* – pole typu *INT* zawierające wartość glikemii, którą wprowadza użytkownik,
- *moment* – pole typu *TEXT* zawierające informację na temat momentu dokonanego pomiaru,
- *note* – pole typu *VARCHAR* zawierające notatkę dotyczącą pomiaru dokonanego przez użytkownika.

5.1.10 Measurement has Insulin

Tabela *measure_has_person* – grupuje informacje dotyczące danego pomiaru, wykonanego przez daną osobę:

- *id* – pole typu *INT* zawierające unikalny identyfikator pola w bazie danych,
- *measurementId* – pole typu *INT* będące kluczem obcym z tabeli *measurement*, zawierające *id* pomiaru dokonanego przez użytkownika,
- *personId* – pole typu *INT* będące kluczem obcym z tabeli *person*, zawierające *id* osoby, dokonującej pomiaru.

5.1.11 Glycemia Ranges

Tabela *glycemia_ranges* – zawiera informacje dotyczące zakresów glikemii użytkownika:

- *id* – pole typu *INT* zawierające unikalny identyfikator pola w bazie danych,
- *hipoglicemy* – pole typu *INT* zawierające wartość hipoglikemii użytkownika,
- *hiperglicemy* – pole typu *INT* zawierające wartość hiperglikemii mierzonej przez użytkownika,
- *hiperglicemyMeal* – pole typu *INT* zawierające wartość hiperglikemii, mierzonej przez użytkownika po posiłku.

5.1.12 Products

Tabela *products* – zawiera informacje dotyczące dostępnych w bazie danych produktów:

- *id* – pole typu *INT* zawierające unikalny identyfikator pola w bazie danych,
- *productName* – pole typu *VARCHAR* przechowujące nazwę danego produktu,
- *kcal* – pole typu *INT* zawierające ilość kalorii, które zawiera dany produkt,
- *protein* – pole typu *DECIMAL* zawierające ilość białka, które zawiera dany produkt,
- *fat* – pole typu *DECIMAL* zawierające ilość tłuszczu, który zawiera dany produkt,
- *carbohydrates* – pole typu *DECIMAL* zawierające ilość węglowodanów, które zawiera dany produkt,
- *fiber* – pole typu *DECIMAL* zawierające ilość błonnika, który zawiera dany produkt,
- *portion* – pole typu *DECIMAL* zawierające wielkość porcji danego produktu.

5.2 Model bazy danych

Rysunek 51 przedstawia schemat EER (*Enhanced Entity–Relationship*) modelu bazy danych użytego do gromadzenia informacji w systemie. Dla każdej tabeli został określony klucz główny, który może być używany w innych tabelach jako klucz obcy. Każde z pól w bazie danych zawiera określony typ, w zależności od tego jakie informacje ma dane pole zawierać. Tabele zawarte w bazie połączone są między sobą relacjami:

- jeden-do-jeden,
- jeden-do-wielu,
- wiele-do-wielu.

W przypadku relacji wiele-do-wielu wykorzystana została tabela grupująca klucze obce z obydwu łączonych w tę relację tabel.

Schemat EER bazy danych został wygenerowany automatycznie za pomocą programu *MySQL Workbench* w wersji 6.3 za pomocą narzędzia *Reverse Engineer*.

Zapytania do bazy danych generowane są w sposób automatyczny za pomocą ORM *Sequelize* na podstawie odwzorowanego w kodzie źródłowym modelu bazy danych. Cała baza danych w programie została wygenerowana w konwecji *Code First* – najpierw stworzony był model, a następnie na podstawie tego modelu odpowiadająca mu tabela bazy danych. Poniższy kod przedstawia przykładową implementację modelu tabeli *products*:

```
module.exports = function(sequelize, DataTypes) {
return sequelize.define('products', {
id: {
type: DataTypes.INTEGER(11),
allowNull: false,
primaryKey: true,
autoIncrement: true
},
productName: {
type: DataTypes.STRING(45),
allowNull: false
},
kcal: {
type: DataTypes.INTEGER(11),
allowNull: false
},
protein: {
type: DataTypes.DECIMAL,
allowNull: false
},
fat: {
type: DataTypes.DECIMAL,
allowNull: false
},
carbohydrates: {
type: DataTypes.DECIMAL,
allowNull: false
},
fiber: {
type: DataTypes.DECIMAL,
allowNull: true
},
portion: {
type: DataTypes.DECIMAL,
allowNull: false
}
```

```
}, {
tableName: 'products',
timestamps : false
});
};
}
```

Początkowo eksportowany jest model przy użyciu funkcji ORM *Sequelize*. Zwracana wartość zawiera odwzorowanie tabeli produkt, na podstawie którego generowane są później automatyczne zapytania do bazy danych. Każde pole tabeli opisywane jest w następujący sposób:

```
nazwa_pola: {
type: typ_pola,
...
//odpowiednie opcje i ograniczenia
}
```

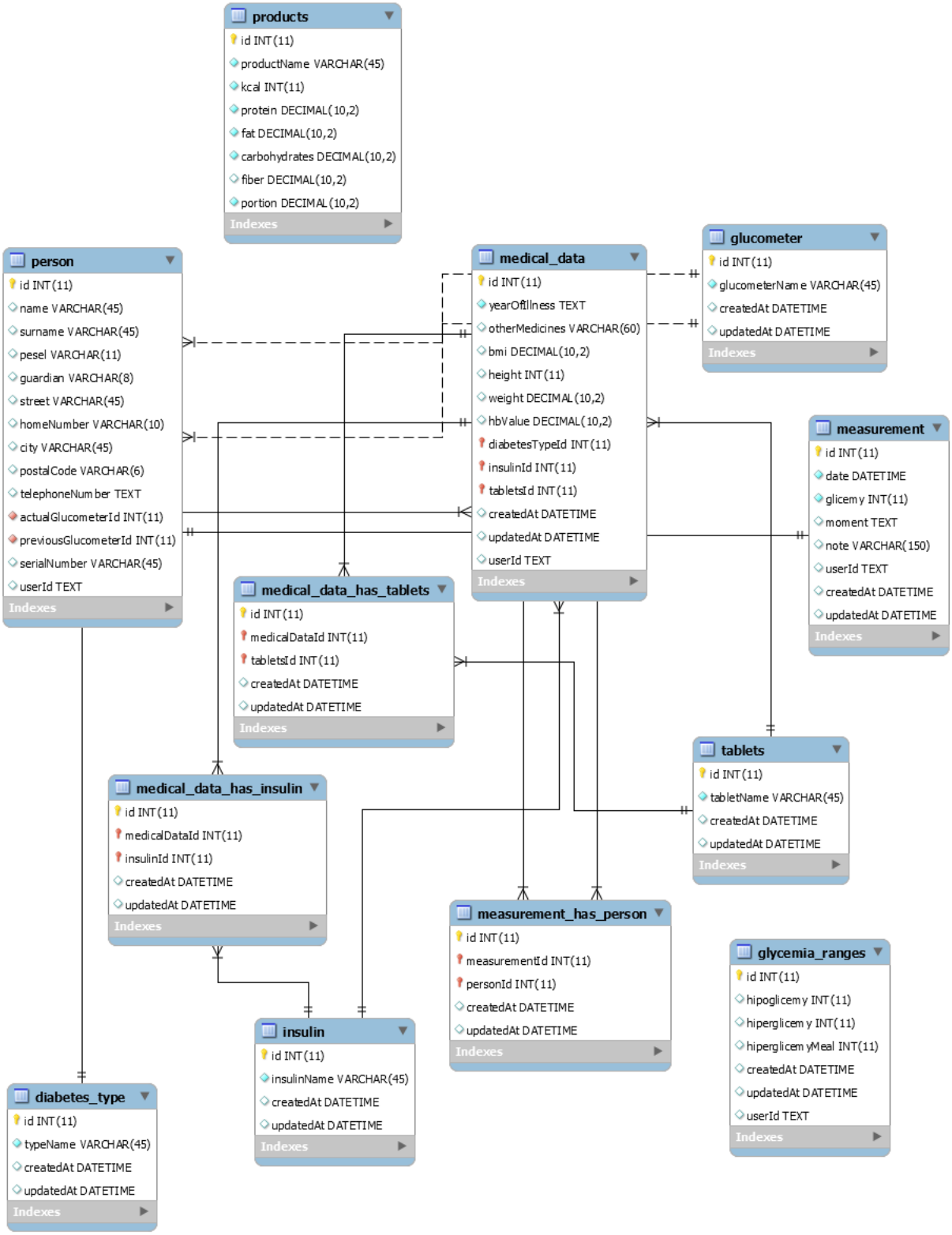
Ograniczenia jakie zostały użyte do zdefiniowania klucza głównego to:

```
allowNull: false;
primaryKey: true;
autoIncrement: true;
```

Pierwsze z nich określa czy pole tabeli może być wartością pustą (*NULL*). Klucz główny nie może być wartością pustą, dlatego wartość ta jest ustawiona na *false*. Następna wartość określa czy dane pole jest kluczem głównym. W tym przypadku przyjmuje ono wartość *true*. Ostatnim z pól jest określenie autoinkrementowania wartości *id*, dlatego również przyjmuje ono wartość *true*. Ostatni znacznik:

```
timestamps: false
```

określa anulowanie możliwości korzystania z automatycznych znaczników czasu tworzonych w momencie edycji *UPDATE* tabeli.



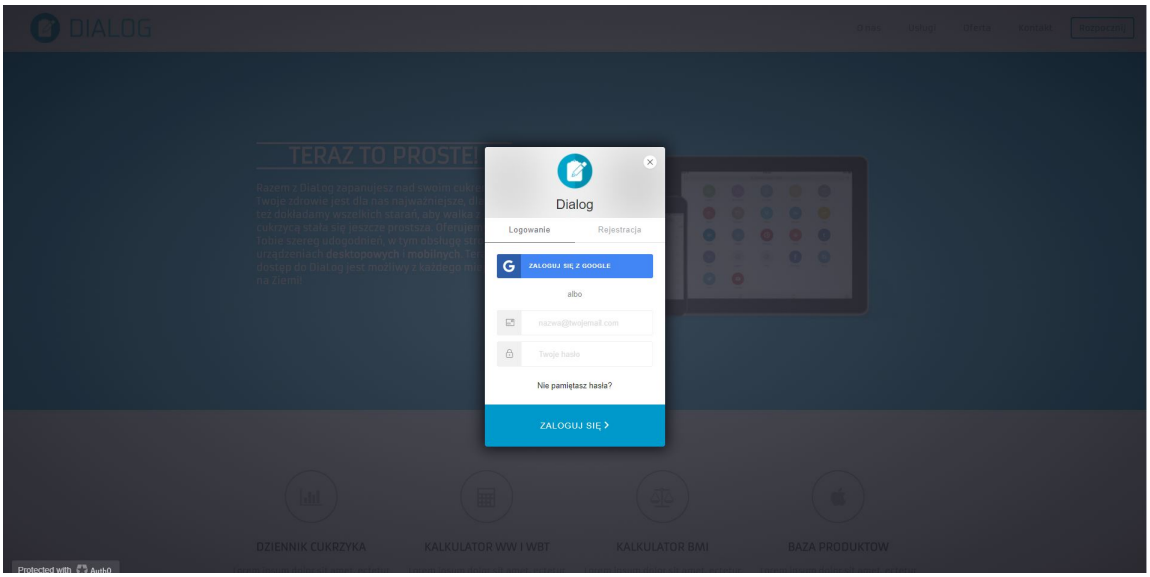
Rysunek 51: Diagram EER bazy danych aplikacji

Rozdział 6

Panel administratora

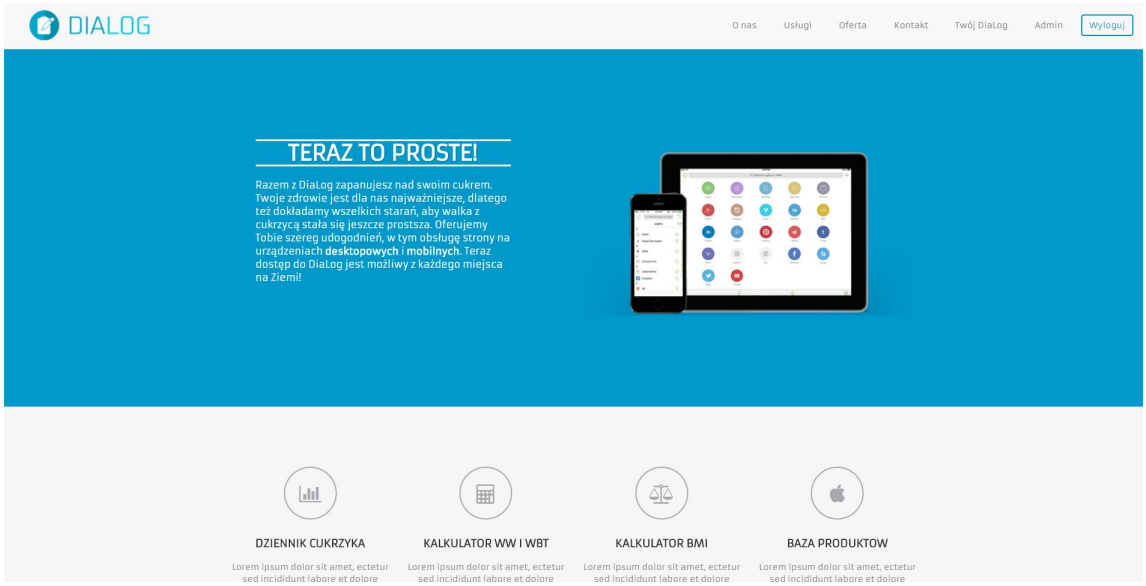
6.1 Logowanie do panelu administratora

W poniższej sekcji przedstawiony zostanie proces logowania do panelu administratora w aplikacji. W pierwszej kolejności użytkownik musi wybrać opcję *Rozpocznij*, dostępną z poziomu górnego menu na stronie głównej, a następnie wybrać przycisk *Logowanie z Google*. Poniższy rysunek 61 przedstawia okno logowania administratora.



Rysunek 61: Okno logowania administratora

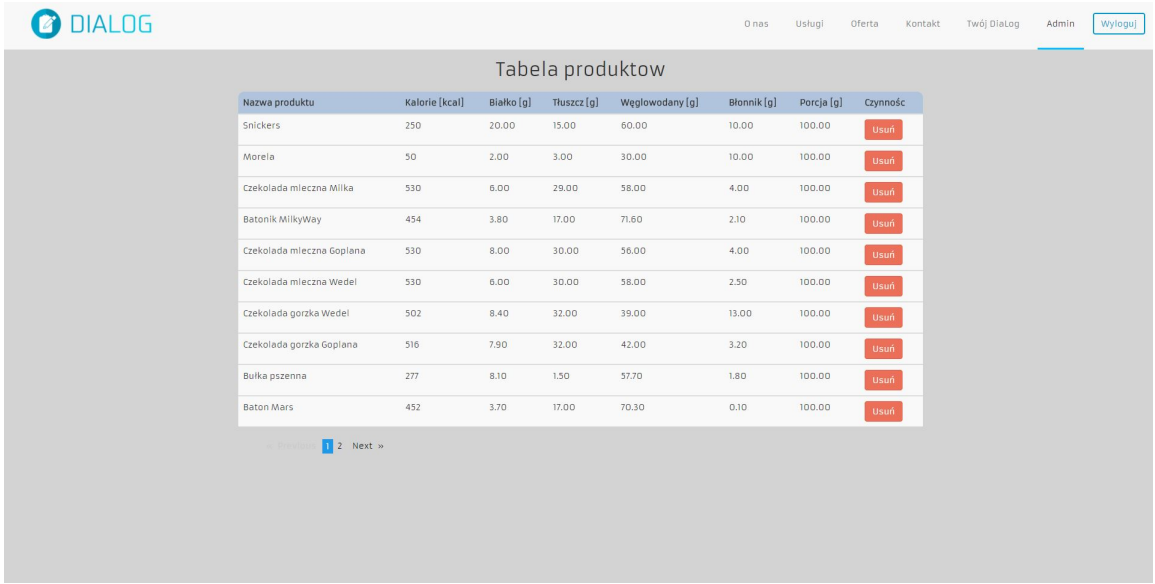
Po wybraniu przycisku *Logowanie z Google* należy zalogować się na konto z przypisanymi uprawnieniami administracyjnymi. Pojawi się wówczas dodatkowa pozycja w menu górnym – *Admin*, co przedstawione zostało na rysunku 62.



Rysunek 62: Okno główne aplikacji z dostępną opcją panelu administratora

6.2 Użytkowanie panelu administratora

Po kliknięciu przycisku *Admin* użytkownik uzyskuje dostęp do Panelu Administratora. Panel ten został stworzony wyłącznie po to, aby usuwać błędnie dodane, bądź zduplikowane produkty dodane do bazy danych produktów. Jest to niezbędny element w przypadku funkcjonalności danej aplikacji rozszerzanych przez użytkowników. Dzięki takiemu rozwiązaniu Administrator w prosty sposób może kontrolować treść zawartą na stronie i w razie potrzeby usuwać ją. Panel administratora przedstawiony został na rysunku 63. Panel został zaprojektowany w postaci tabeli tak, aby w prosty i przejrzysty sposób zaprezentować wszystkie dostępne dane. Ponieważ baza produktów może być stale rozszerzana, a ich ilość może sięgać nawet kilkuset elementów, zastosowana została paginacja tabeli, dzieląca jej zawartość na strony zawierające po dziesięć elementów.



Rysunek 63: Panel administratora aplikacji

Tabela podzielona jest na osiem kolumn:

1. *Nazwa produktu*,

2. *Kalorie [kcal]*,
3. *Białko [g]*,
4. *Tłuszcz [g]*,
5. *Węglowodany [g]*,
6. *Błonnik [g]*,
7. *Porcja [g]*,
8. *Czynność*.

W kolumnie *Czynność* przy każdym z wierszy dostępny jest przycisk *Usuń*, dzięki któremu Administrator może usunąć wybrany produkt. Usuwanie produktu odbywa się w sposób dynamiczny, co oznacza, że produkt po usunięciu od razu (bez odświeżenia strony) znika z listy dostępnych produktów i nie jest on dostępny nawet z poziomu tabeli produktów dostępnej w zakładce *Kalkulatory* z poziomu panelu użytkownika zalogowanego.

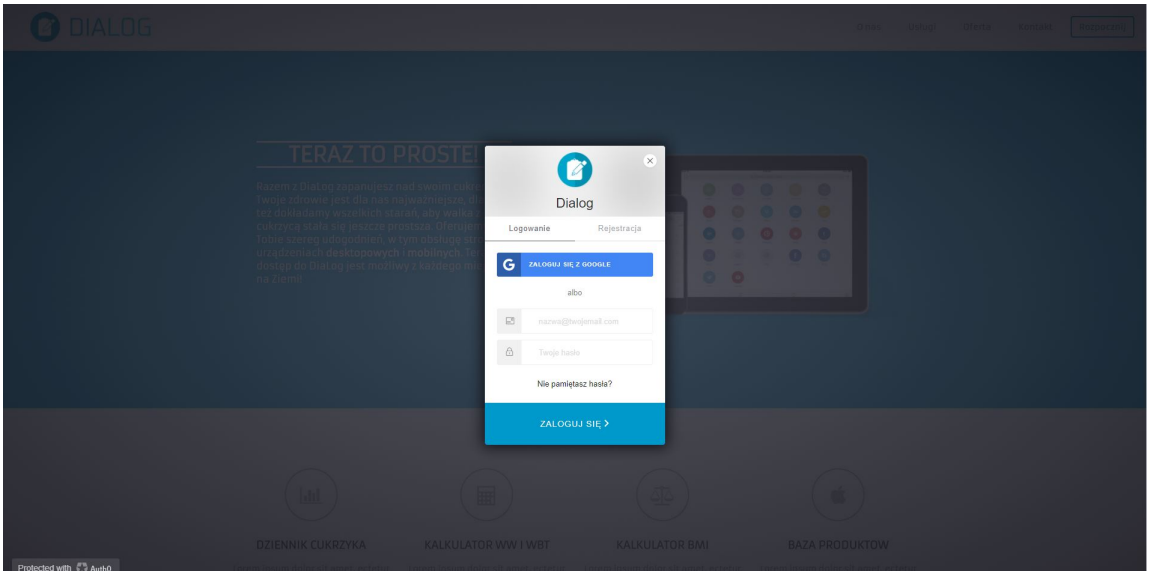
Administrator, tak jak każdy zalogowany użytkownik, ma również możliwość korzystania ze wszystkich funkcjonalności systemu przeznaczonych dla tej grupy użytkowników – zakładka *Twój DiaLog*.

Rozdział 7

Panel użytkownika zarejestrowanego

7.1 Logowanie do panelu użytkownika

W poniższej sekcji przedstawiony zostanie proces logowania do panelu użytkownika w aplikacji. W pierwszej kolejności użytkownik musi wybrać opcję *Rozpocznij*, dostępną z poziomu górnego menu na stronie głównej, a następnie wpisać login i hasło, bądź wybrać przycisk *Zaloguj się z Google*. Okno logowania przedstawione jest na rysunku 71.



Rysunek 71: Okno logowania użytkownika

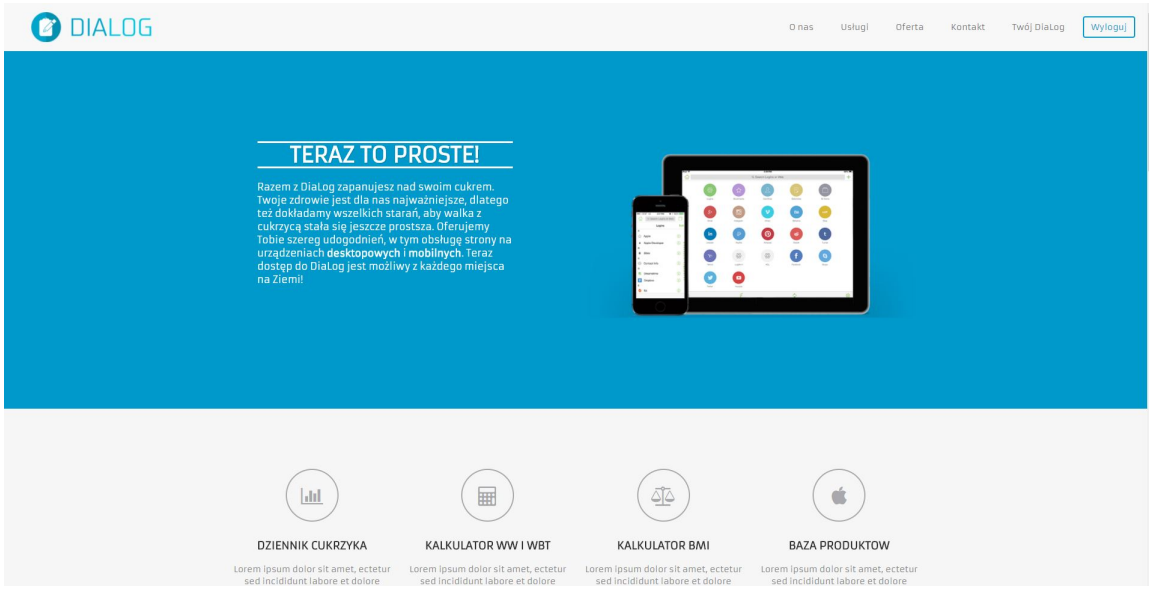
Po wpisaniu danych służących do logowania użytkownik uzyskuje dostęp do nowej pozycji w menu górnym – *Twój DiaLog*. Menu górne dostępne po zalogowaniu użytkownika zostało ukazane na rysunku 72.

7.2 Użytkowanie panelu użytkownika zarejestrowanego

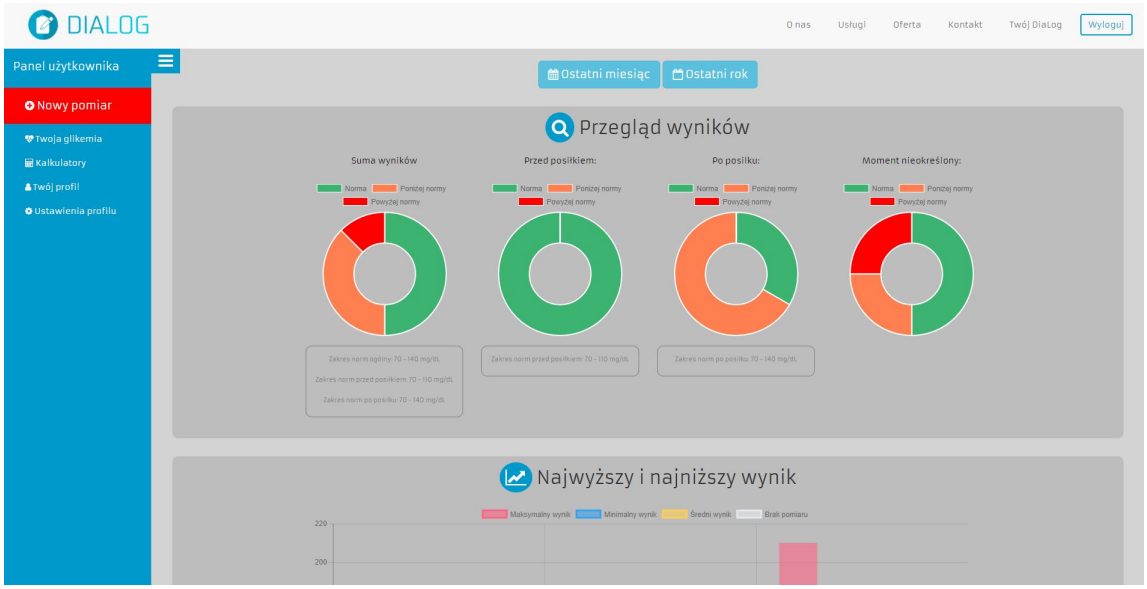
Po uzyskaniu dostępu do zakładki *Twój DiaLog* i kliknięciu jej, użytkownik uzyskuje dostęp do strony głównej panelu użytkownika zarejestrowanego będącej jednocześnie komponentem podsumowującym pomiary wprowadzone przez użytkownika. Początkowo, po rejestracji wykresy i tabela zestawień są puste, gdyż nie zawierają żadnych danych. Strona główna po kliknięciu pozycji górnego menu *Twój DiaLog* została ukazana na rysunku 73.

7.2.1 Twoja glikemia

Zawartość dostępna z poziomu zakładki *Twoja glikemia* wyświetlana jest jako strona główna po kliknięciu zakładki *Twój DiaLog*. Zawiera ona wykresy kołowe podsumowujące ilość dodanych



Rysunek 72: Menu górne rozszerzone o dodatkową pozycję – *Twój DiaLog*, dostępne po zalogowaniu użytkownika



Rysunek 73: Strona główna dostępna po kliknięciu przycisku menu górnego *Twój DiaLog*, dostępnego po zalogowaniu użytkownika

pomiarów glikemii. Każdy z czterech wykresów kołowych dotyczy odrębnego rodzaju pomiaru. Rodzaje pomiarów glikemii rozróżniane są w następujący sposób:

- Przed posiłkiem,
- Po posiłku,
- Moment nieokreślony.

Pierwszy z wykresów jest sumą ilości pomiarów z pozostałych trzech wykresów. Ponadto, każdy z wykresów rozróżnia normy zakresu glikemii odpowiednie dla danego rodzaju pomiaru:

- Przed posiłkiem:
  - Norma ogólna: 70 - 140 mg/dL.

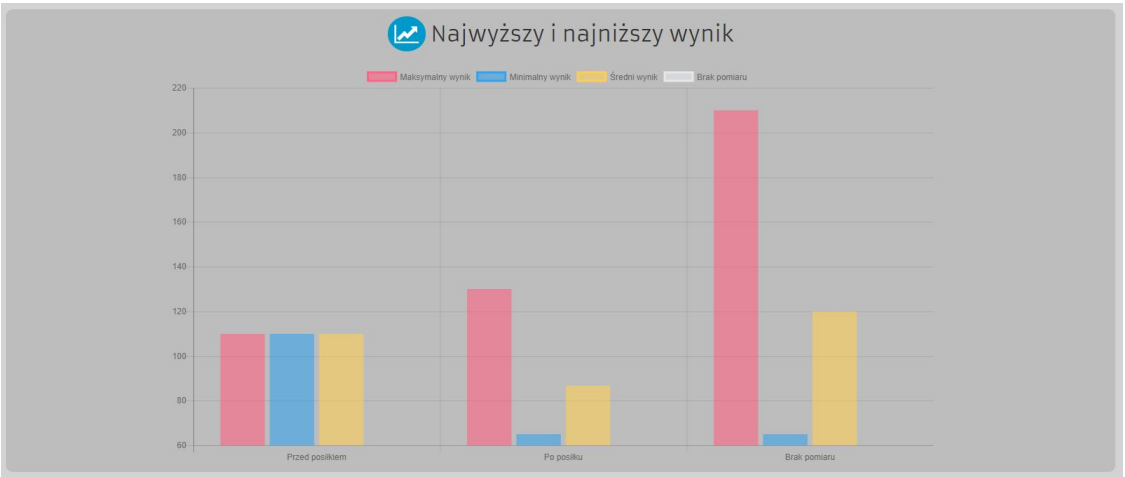
- Po posiłku,
  - Zakres norm przed posiłkiem: 70 - 110 mg/dL.
- Moment nieokreślony.
  - Zakres norm po posiłku: 70 - 140 mg/dL.

Dla lepszej przejrzystości, do zaznaczenia norm na wykresie zostały użyte charakterystyczne kolory. Przy przekroczeniu normy wykres przyjmuje kolor czerwony. Jeżeli pomiar glikemii mieścił się w normie, wykres zostaje zaznaczony kolorem zielonym. W przypadku wartości glikemii znajdującej się poniżej normy wykres przyjmuje kolor pomarańczowy. Dodatkowo, po najechaniu kursorem na daną część wykresu użytkownikowi wyświetla się dymek z podpowiedzią ile pomiarów znajduje się w danym zakresie norm.

Rysunek 74: Wykresy kołowe dostępne z poziomu zakładki *Twoja glikemia*

.

Kolejnym elementem dostępnym z poziomu zakładki *Twoja glikemia* są wykresy słupkowe. Zestawiają one maksymalny, minimalny oraz średni wynik dla każdego z rodzajów pomiarów glikemii. Po najechaniu na dany słupek wykresu użytkownik ma dostęp do informacji o wartości danego wyniku.



Rysunek 75: Wykresy słupkowe dostępne z poziomu zakładki *Twoja glikemia*

.

Ostatnim z elementów zakładki *Twoja glikemia* jest tabela zestawień. Podsumowuje ona (w zależności od pory dnia – rano, południe, wieczór, noc) następujące dane:

- Liczba wyników,

- Najwyższy wynik (mg/dL),
- Najniższy wynik (mg/dL),
- % wyników powyżej normy,
- % wyników poniżej normy,
- % wyników w normie.

Ostatnia kolumna tabeli przedstawia sumę wyników danego wiersza.

Czynnik główny	Rano	Południe	Wieczór	Noc	Łącznie
Liczba wyników:	1	1	6	3	11
Najwyższy wynik (mgdL):	76	250	220	180	---
Najniższy wynik (mgdL):	76	250	65	45	---
% powyżej normy:	0.00%	100%	50.0%	33.3%	45.5%
% poniżej normy:	0.00%	0.00%	16.7%	33.3%	18.2%
% w normie:	100%	0.00%	33.3%	33.3%	36.4%

Rysunek 76: Tabela zestawień pomiarów glikemii dostępna z poziomu zakładki *Twoja glikemia*

.

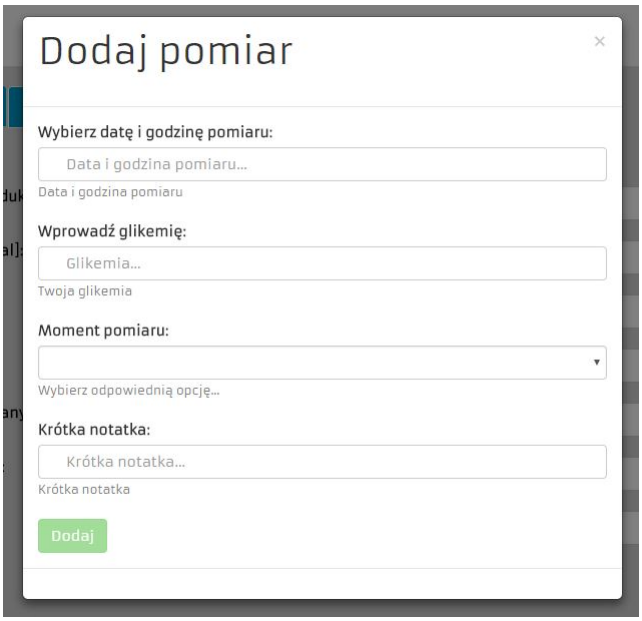
Użytkownik ma możliwość wybrania przedziału czasu, na podstawie którego wyświetlane mają być dane. Dostępne są dwie opcje – *Ostatni miesiąc* oraz *Ostatni rok*. Wyboru dokonuje się poprzez kliknięcie jednego z dwóch dostępnych przycisków na szczycie podstrony.

7.2.2 Dodaj pomiar

Zakładka *Dodaj pomiar* otwiera okno modalne dodawania nowego pomiaru (rys. 77). Z poziomu tego okna użytkownik wprowadza dane, które zestawiane są później na stronie komponentu *Twoja glikemia* w postaci wykresów i tabeli zestawień:

- Data i godzina pomiaru,
- Wartość glikemii,
- Moment pomiaru,
- Krótka notatka.

Po naciśnięciu przycisku *Dodaj* zostaje dodany do bazy danych pomiarów nowy pomiar.



Rysunek 77: Okno modalne dodawania nowego pomiaru

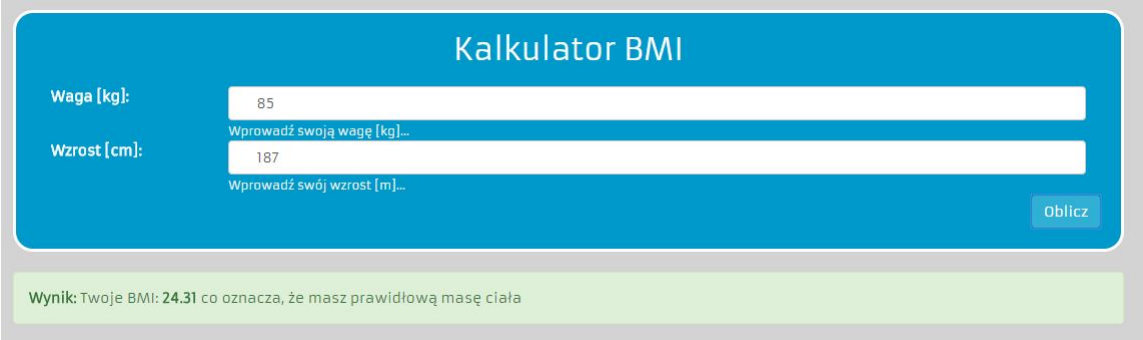
7.2.3 Kalkulatory

Zakładka *Kalkulatory* pozwala użytkownikowi na uzyskanie dostępu do kalkulatora *BMI (Body Mass Index)* oraz do kalkulatora wymienników węglowodanowych oraz białkowo -tłuszczowych, dostępnego wraz z tabelą makroskładników produktów spożywczych pochodzących z bazy danych produktów. Kalkulator *BMI (Body Mass Index)* pozwala obliczyć współczynnik masy do wzrostu. Składa się on z dwóch pól typu *Input*. W pierwszym z nich użytkownik wpisuje swoją wagę, a w następnym wzrost (rys. 78). Po kliknięciu przycisku *Oblicz* użytkownik otrzymuje informację o uzyskanym wyniku współczynnika *BMI* oraz podpowiedź do jakiej grupy osób (ze względu na wagę) się zalicza (rys. 79):

- osoby z niedowagą,
- osoby z wagą prawidłową,
- osoby z nadwagą,
- osoby z otyłością I stopnia,
- osoby z otyłością II stopnia,
- osoby z otyłością III stopnia.

Rysunek 78: Interfejs kalkulatora BMI

.



Rysunek 79: Interfejs kalkulatora BMI po uzyskaniu wyniku

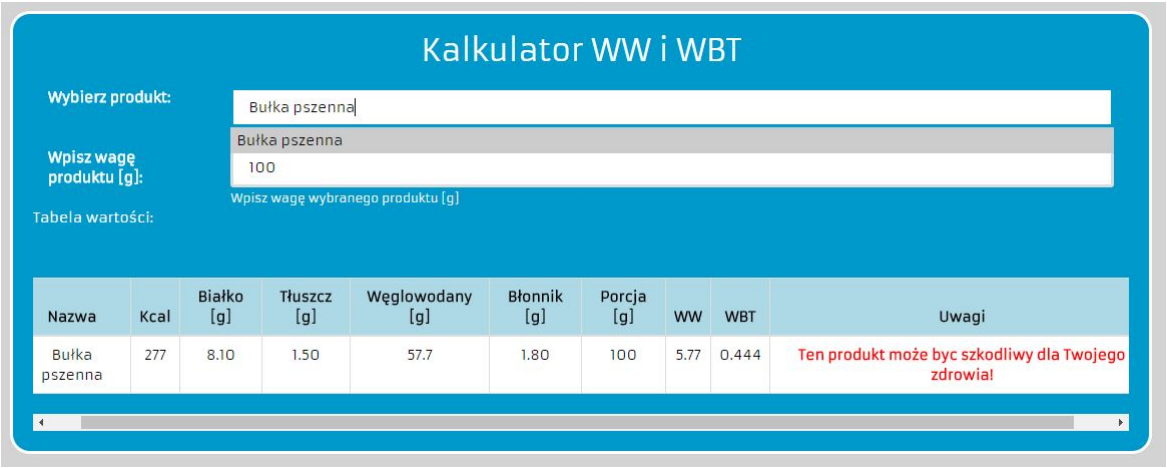
.

Kolejną funkcjonalnością dostępną z poziomu zakładki *Kalkulatory* menu bocznego jest kalkulator wymienników węglowodanowych i wymienników białkowo-tłuszczowych na podstawie tabeli makroskładników produktów dostępnych z poziomu tabeli *products*, która może być stale rozszerzana przez zalogowanych użytkowników aplikacji (rys. 710). Wynik przedstawiany jest w postaci kolumn dołączonych do tabeli makroskładników danego produktu. Aby obliczyć wartość wymienników węglodowanowych i wymienników białkowo-tłuszczowych należy wybrać dany produkt poprzez wpisanie ciągu (bądź części ciągu) znaków w polu formularza o nazwie *Wybierz produkt.* Do tego pola przypięty jest inteligentny moduł wyszukiwania znajdujący produkty po słowach kluczowych. Po wpisaniu części wyrazu moduł podpowiada użytkownikowi najbardziej prawdopodobne wyniki w postaci listy rozwijanej. Dodatkowo użytkownik ma możliwość podania porcji produktu w polu *Wpisz wagę produktu [g].* Wartości makroskładników i wyników obliczonych przez kalkulatory są wówczas dynamicznie zmieniane w zależności od wpisanej porcji produktu. Domyślna wartość pola to 100 [g].

Dodatkowo, jako ostatnia kolumna tabeli dodana została informacja czy produkt może być szkodliwy dla zdrowia osoby chorej na cukrzycę czy też nie. System rekomendacji oparty jest na ilości węglowodanów zawartych w produkcie. Jeżeli w 100 gramach produktu, 50 gram to węglowodany, wówczas produkt może okazać się szkodliwy w diecie cukrzyka (rys. 711).

Rysunek 710: Interfejs kalkulatora wymienników węglowodanowych i białkowo-tłuszczowych

.



Rysunek 711: Interfejs kalkulatora wymienników węglowodanowych i białkowo-tłuszczowych po uzyskaniu wyniku

.

Użytkownik zarejestrowany ma również możliwość dodawania produktów do bazy danych produktów. Formularz dodawania nowego produktu dostępny jest również z poziomu zakładki *Twoja glikemia* – karta *Dodaj produkt* (rys. 712).

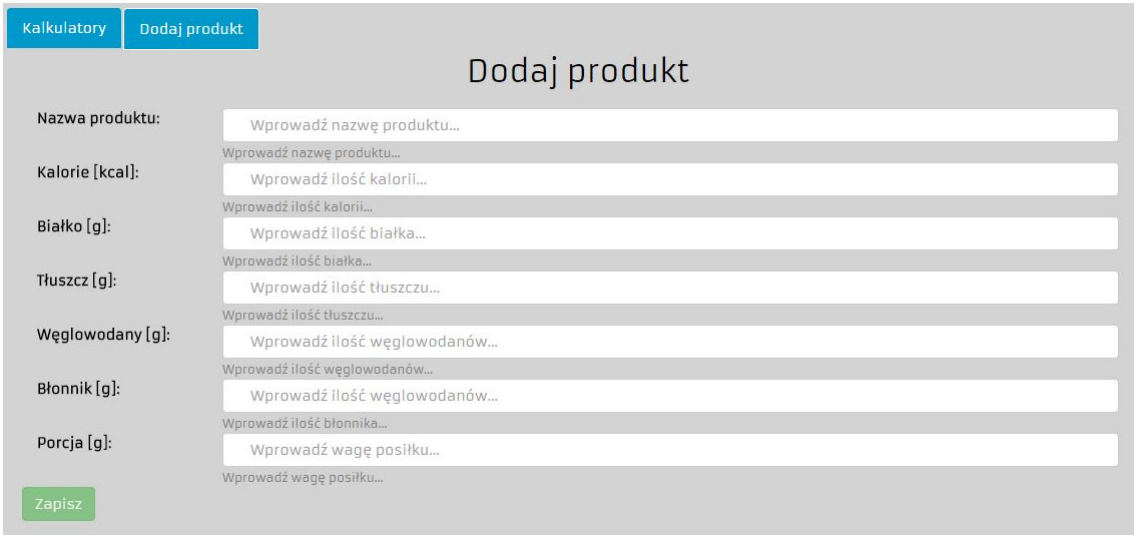
Po wypełnieniu wszystkich pól:

- Nazwa produktu,
- Kalorie [kcal],
- Białko [g],
- Tłuszcz [g],
- Węglowodany [g],
- Błonnik [g],
- Porcja [g]

produkt jest dynamicznie dodawany do bazy danych i bezpośrednio dostępny z pozycji modułu kalkulatora do obliczania wymienników węglowodanowych i białkowo-tłuszczowych.

7.2.4 Twój profil

Z poziomu zakładki *Twój profil* użytkownik ma możliwość dodania informacji zarówno o profilu (rys. 713) jak i danych medycznych (rys. 714). Czynność ta odbywa się przy pomocy dwóch formularzy z odpowiednimi polami:

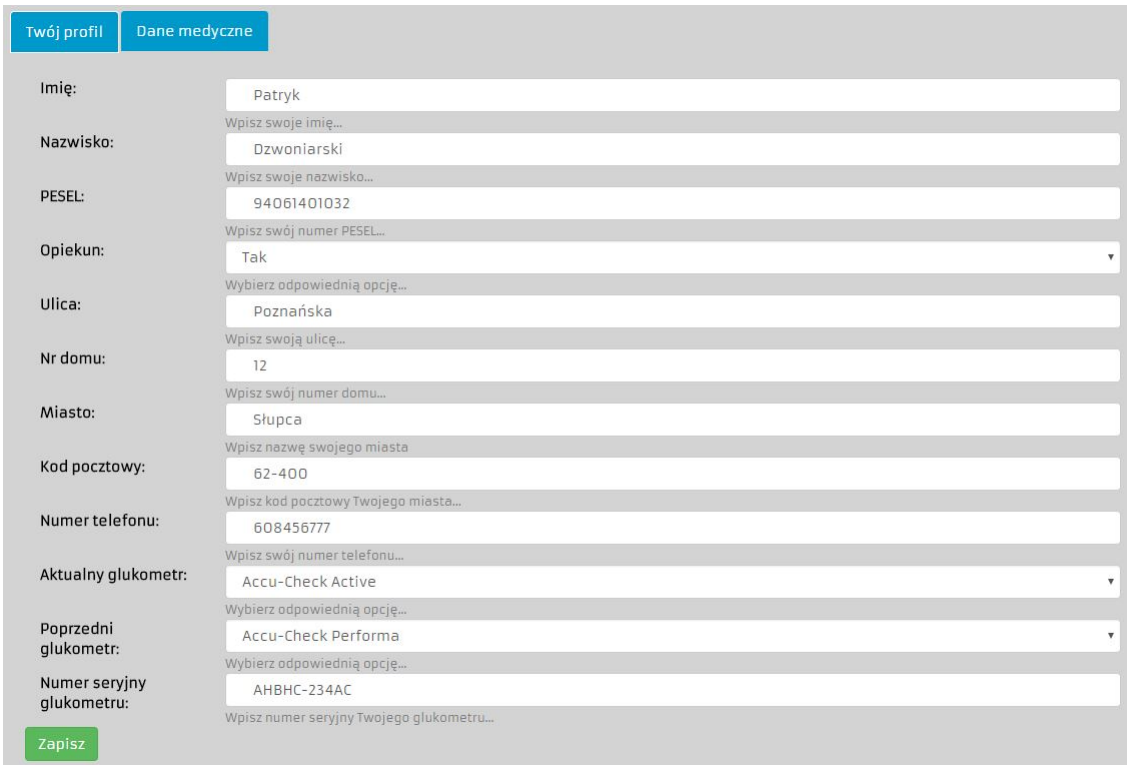


Rysunek 712: Formularz dodawania nowego produktu

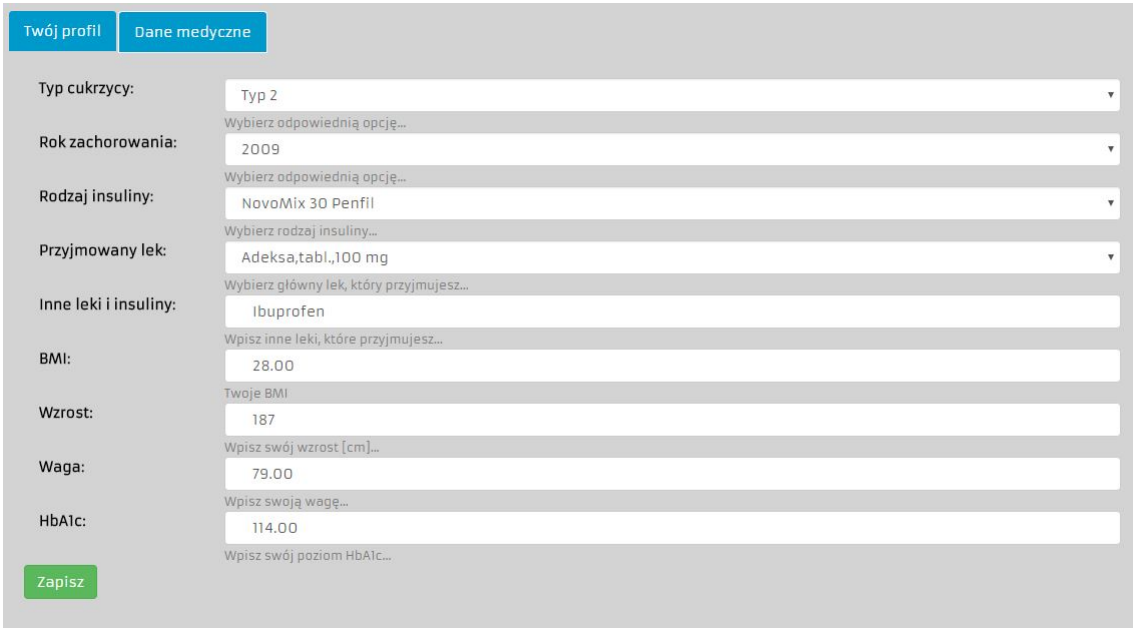
.

- Dla karty *Twój profil*
  - imię,
  - nazwisko,
  - PESEL,
  - opiekun – lista rozwijana,
  - ulica,
  - nr domu,
  - miasto,
  - kod pocztowy,
  - numer telefonu,
  - aktualny glukometr – lista rozwijana,
  - poprzedni glukometr – lista rozwijana,
  - numer seryjny glukometru.
- Dla karty *Dane medyczne*
  - typ cukrzycy – lista rozwijana,
  - rok zachorowania – lista rozwijana,
  - rodzaj insuliny – lista rozwijana,
  - przyjmowany lek – lista rozwijana,
  - inne leki i insuliny,
  - BMI,
  - wzrost,
  - waga,
  - hBA1c.

Przy pierwszym wpisaniu danych do wszystkich pól formularza i zatwierdzeniu ich przyciskiem *Zapisz*, zostanie wykonana operacja zapisania danych do obydwu tabel bazy danych. Przy każdorazowym otwarciu zakładki *Twój profil* dane te będą czytane z bazy danych do pól formularza. W przypadku chęci zaktualizowania informacji użytkownik nadpisuje pola formularza nowymi danymi i wykonywana jest operacja *UPDATE* zarówno w przypadku jednej, jak i drugiej tabeli.



Rysunek 713: Formularz dodawania i aktualizacji informacji o profilu



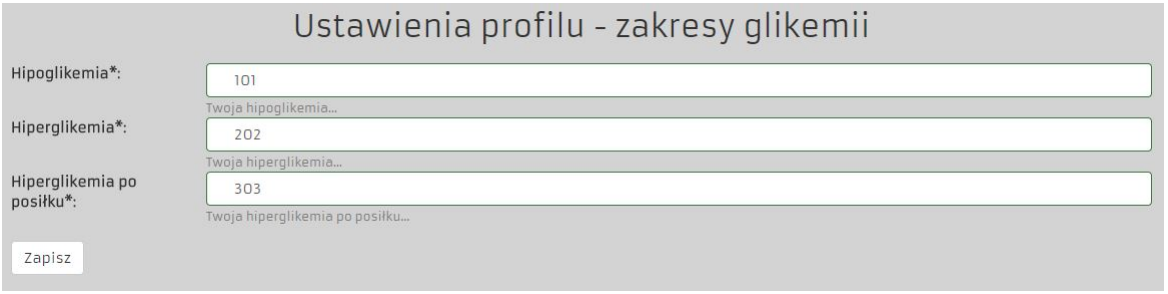
Rysunek 714: Formularz dodawania i aktualizacji danych medycznych użytkownika profilu

7.2.5 Ustawienia profilu

W zakładce *Ustawienia profilu* użytkownik ma możliwość dodania, bądź zaktualizowania bieżących zakresów glikemii (rys. 715). Komponent składa się z trzech pól formularza:

- Hipoglikemia,

- Hiperglikemia,
- Hiperglikemia po posiłku.



Rysunek 715: Formularz dodawania i aktualizacji danych dotyczących zakresów glikemii użytkownika

Każde pole może być w dowolnym czasie aktualizowanie (operacja *UPDATE*) w bazie danych. Ponadto dane wczytywane są bezpośrednio do pól formularza przy użyciu metody *GET*.

Rozdział 8

Panel użytkownika niezarejestrowanego oraz niezalogowanego

8.1 Okno rejestracji użytkownika

W poniższej sekcji ukazany zostanie interfejs programu umożliwiający użytkownikowi niezarejestrowanemu w systemie dokonania rejestracji. Rejestracja przebiegać może w dwóch wariantach:

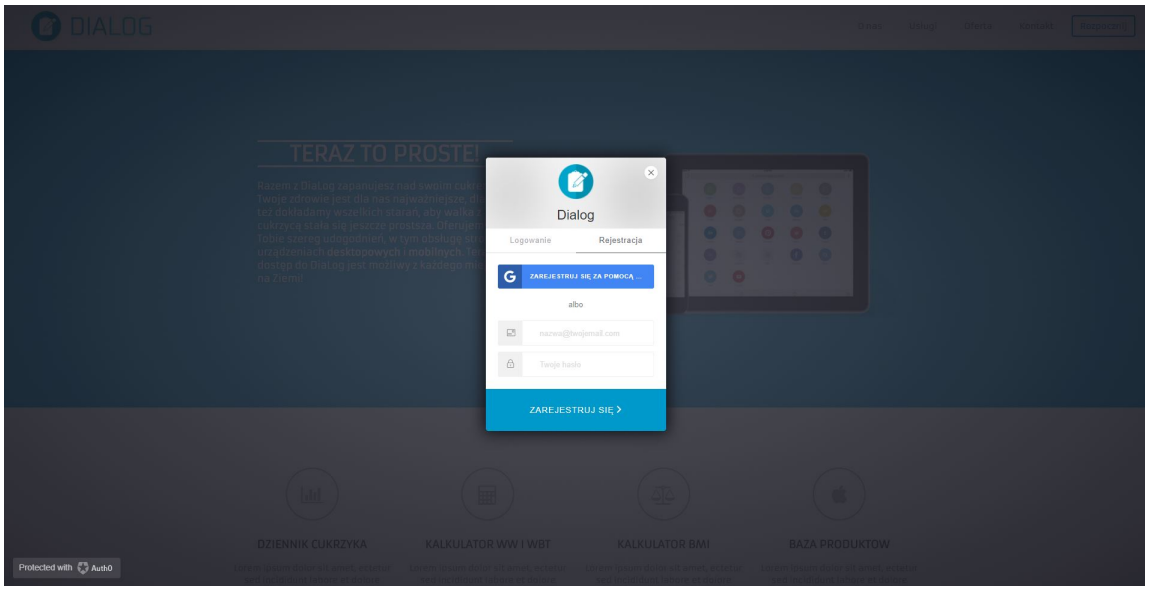
- Rejestracja przy użyciu konta *Google*,
- Rejestracja tradycyjna – podanie adreu e-mail oraz hasła.

8.1.1 Rejestracja przy użyciu konta *Google*

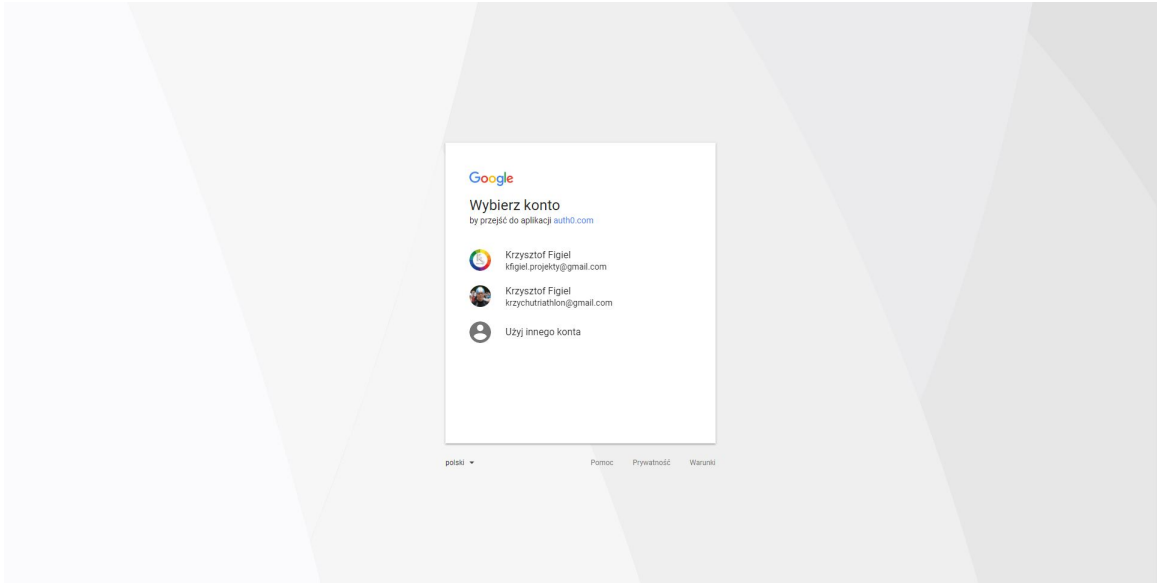
Opcja rejestracji i logowania za pomocą konta *Google* znacznie ułatwia i przyspiesza proces dodawania nowego użytkownika. Pozwala ona zwiększyć odsetek nowych użytkowników systemu. Kolejnym powodem, który przemawia za tym, aby stosować logowanie przy użyciu danych pochodzących z portali społecznościowych jest fakt mówiący o tym, że adres e-mail, którym posługuje się dana osoba w momencie takiej rejestracji został już wcześniej zweryfikowany przez dostawcę sieci społecznościowej. Oznacza to, że w momencie logowania przy użyciu konta społecznościowego otrzymujemy rzetelne informacje, a nie fałszywe adresy, które użytkownicy wykorzystują zazwyczaj do zarejestrowania się na stronach internetowych.

Cały proces rejestracji i późniejszego logowania się do aplikacji przy użyciu konta *Google* przebiega w następujący sposób:

- użytkownik wybiera opcję rejestracji w systemie i klika przycisk *Zaloguj się za pomocą konta Google* (rys. 81) i wpisuje dane logowania do konta *Google*, bądź w przypadku kiedy jest zalogowany globalnie, wybiera odpowiednie konto (rys. 82),
- żądanie rejestracji/logowania wysyłane jest do dostawcy sieci społecznościowej *Google*,
- w momencie gdy dostawca sieci społecznościowej *Google* potwierdzi tożsamość użytkownika, bieżący użytkownik uzyskuje dostęp do aplikacji.



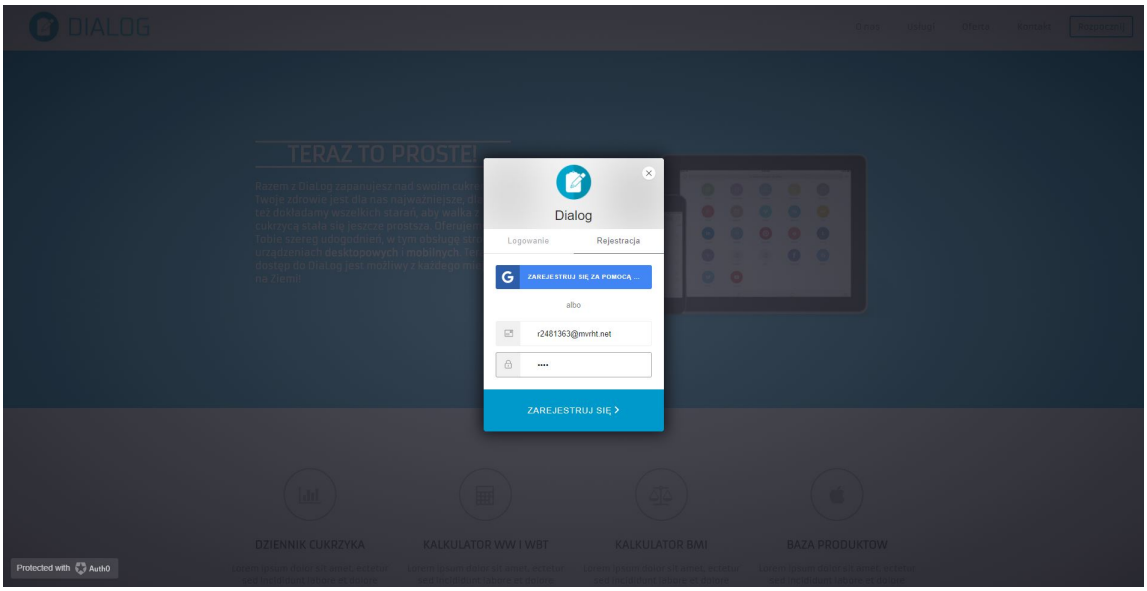
Rysunek 81: Okno rejestracji użytkownika do systemu



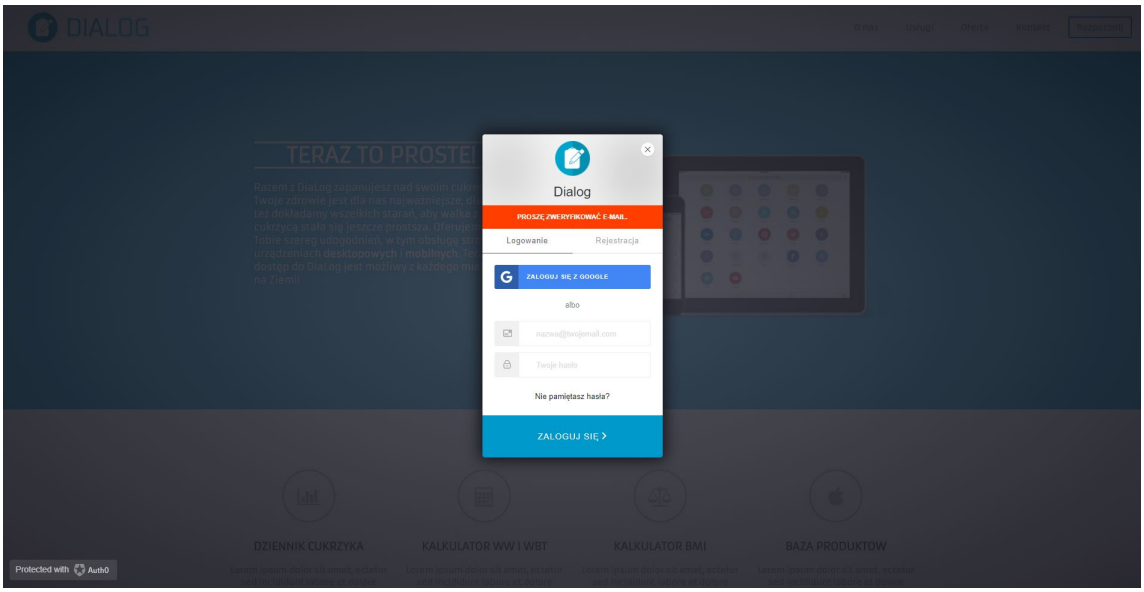
Rysunek 82: Okno wyboru konta *Google*

8.1.2 Rejestracja tradycyjna – podanie adreu e-mail oraz hasła

W przypadku rejestracji tradycyjnej dane podane przez użytkownika (adres e-mail oraz hasło) zostają zapisane w zewnętrznej bazie danych dostawcy usług autoryzacji – *Auth0*. Po kliknięciu przycisku *Zarejestruj się* (rys. 83) użytkownik zostaje powiadomiony o konieczności potwierdzenia autentyczności konta poprzez kliknięcie w link aktywacyjny wysłany automatycznie w wiadomości e-mail przez serwis *Auth0* (rys. 84). Po kliknięciu w link aktywacyjny możliwe jest już zalogowanie się.



Rysunek 83: Okno rejestracji tradycyjnej

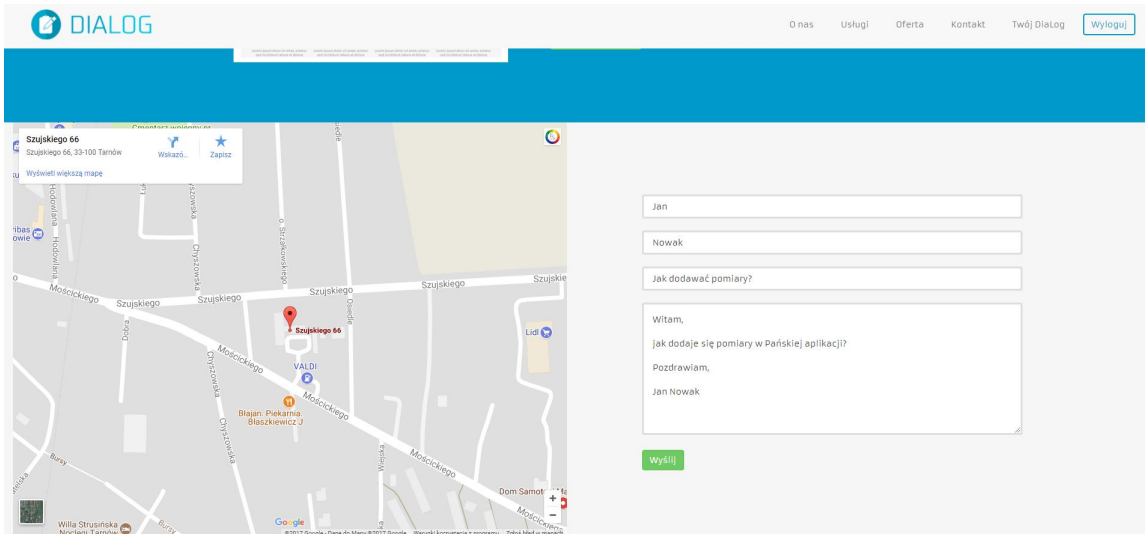


Rysunek 84: Okno z prośbą o potwierdzenie adresu e-mail

Po poprawnej rejestracji i późniejszym zalogowaniu się użytkownik ma dostęp do panelu użytkownika zarejestrowanego. Z poziomu panelu użytkownika niezalogowanego możliwe jest również uzyskanie informacji na temat funkcjonalności strony oraz wysłanie maila z zapytaniem do administracji za pomocą wbudowanego formularza z podpiętym modułem do wysyłania wiadomości e-mail.

Użytkownik może w każdej chwili wypełnić wymagane pola oraz wysłać wiadomość do administracji z dowolnym zapytaniem, klikając przycisk *Wyślij* (rys. 85):

- Imię,
- Nazwisko,
- Temat,
- Wiadomość



Rysunek 85: Formularz do wysyłania maili do administracji

Rozdział 9

Bezpieczeństwo aplikacji

Do przechowywania wrażliwych danych użytkownika, takich jak hasło wykorzystywane są mechanizmy oferowane przez usługodawcę *Auth0*. Dane przechowywane w bazie tego serwisu nie mają postaci zwykłego tekstu, a przechowywane za pomocą skrótu *BCrypt*.

BCrypt jest funkcją skrótu kryptograficznego, która została stworzona w celu przechowywania haseł statycznych, czyli haseł znanych wyłącznie osobie, która chce się uwierzytelnać, a nie dowolnych danych binarnych. *BCrypt* wymaga stosowania soli, co wyróżnia ją od innych funkcji skrótu. Sól w algorytmie *BCrypt* jest złożona z nastepujących elementów:

- *version* oznaczające wersję algorytmu *BCrypt*,
- *rounds* jest to liczba z przedziału 04-99, która określa tzw. *work factor* algorytmu, domyślna wartość tego pola to $12,
- *saltaddon* – są to losowe 22 znaki, które mają za zadanie powiększać sól weryfikacja tego ciągu przebiega przy użyciu wyrażenia regularnego [./A-Za-z0-9] – znaki te mogą być wylosowane przez użytkownika, bądź przez specjalnie zaprojektowany do tego celu algorytm.

W *Auth0* zarówno dane *REST-owe*, jak i przekazywane w ruchu sieciowym są szyfrowane. Cała komunikacja sieciowa wykorzystuje protokół TLS (*Transport Layer Security*) w wersji 1.2 (będący rozwinięciem protokołu SSL), z co najmniej 128-bitowym szyfrowaniem AES (*Advanced Encryption Standard*). Do wymiany kluczy wykorzystywany jest mechanizm *ECDHE_RSA* oparty na protokole *Diffiego-Hellmana* z podpisem generowanym przy użyciu kryptograficznego algorytmu asymetrycznego RSA (*Rivest–Shamir–Adleman*).

Usługi świadczone przez serwis *Auth0* zaprojektowane są z myślą o wysokiej dostępności i odporności. Aplikacje korzystające z *Auth0* są częściowo zabezpieczone przed atakami typu *Odmowa usługi* czy *Uwierzytelnianie*. Mają one wbudowane funkcje ograniczania szybkości i automatycznego blokowania. Ponadto konta użytkowników zabezpieczone są za pomocą domyślnie wbudowanego modułu weryfikacji autentyczności użytkownika przy użyciu adresu e-mail. Każdy użytkownik systemu otrzymuje unikalny JWT (*JSON Web Token*), który pozwala na rozróżnianie i sprawdzanie autentyczności użytkowników aplikacji. Aplikacja została zabezpieczona przed skopiowaniem linku do panelu użytkownika zalogowanego i wklejeniu go do innego okna przeglądarki w celu uzyskania dostępu. W momencie takiej próby użytkownik zostaje poinformowany, że nie ma uprawnień do przeglądania treści danej strony (rys. 91).

Rysunek 91: Błąd przy próbie uzyskania dostępu do panelu użytkownika zalogowanego poprzez link URL

Jak widać u góry (w menu górnym) nie ma dostępu do zakładki *Twój dialog*, co oznacza, że użytkownik jest niezalogowany i próbuje uzyskać dostęp do linku URL udostępnianego wyłącznie użytkownikowi zalogowanemu.

Podobna sytuacja tyczy się panelu administratora. Użytkownik nie posiadający w serwisie *Auth0* roli *Admin* nie posiada dostępu do funkcjonalności dla niej przeznaczonych. To oznacza, że nie da się uzyskać bezpośredniego dostępu do podstrony panelu administratora poprzez wklejenie linku URL do okna wyszukiwarki. Użytkownik przy takiej próbie zostaje natychmiastowo informowany właściwym komunikatem (rys. 92).

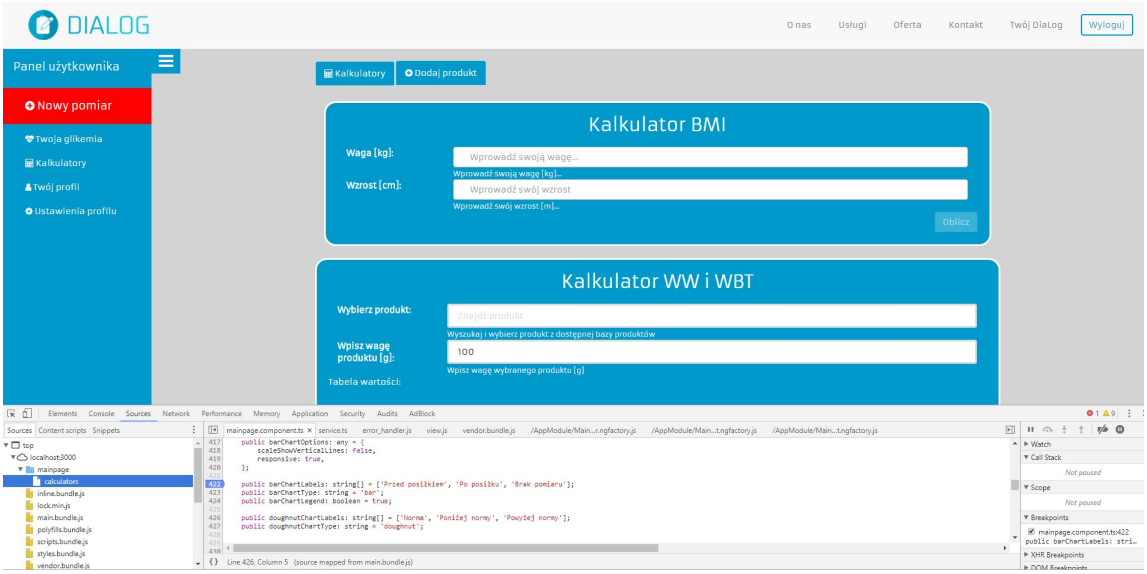
Rysunek 92: Błąd przy próbie uzyskania dostępu do panelu administratora poprzez link URL

Rozdział 10

Testy

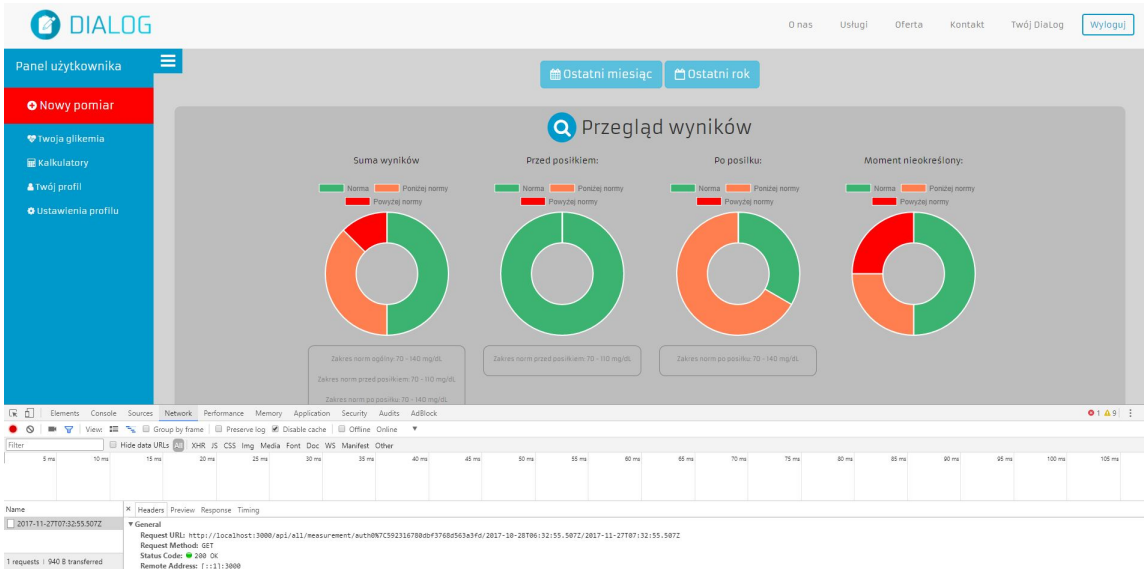
10.1 Debugowanie aplikacji i dostęp do informacji o jej działaniu

Do wykonywania debugowania aplikacji w celu znalezienia błędów użyto wtyczki do programu *Visual Studio Code* oraz wbudowanego w przeglądarkę internetową *Google Chrome*, debuggera dostępnego z poziomu panelu narzędzi deweloperskich (zakładka *Sources*) widoczna na rysunku 101 . W tym miejscu możliwe jest podejrzenie i debugowanie skryptów aplikacji. Programista ma również możliwość śledzenia wybranych wyrażeń (pole *Watch expressions*), przeglądania stosu aplikacji (pole *Call Stack*), ustawiania tzw. *breakpointów* w momencie debugowania, przechodzenia kodu krok po kroku i wiele, wiele innych.



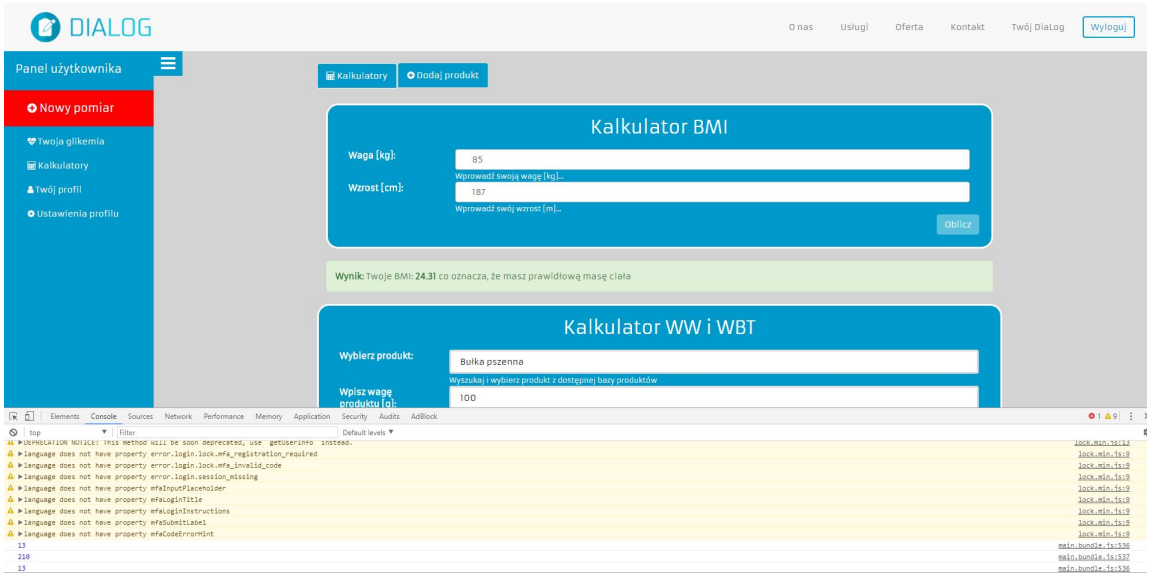
Rysunek 101: Opcje debugowania i podglądu kodu dostępne z poziomu zakładki *Sources* narzędzi deweloperskich przeglądarki *Google Chrome*

Do podglądu poprawności wysyłanych żądań za pomocą metod HTTP użyto narzędzi dostępnych z poziomu zakładki *Network*. Oferuje ona podgląd adresów URL (*Uniform Resource Locator*) generowanych do pobrania danych w formacie JSON oraz treści tablicy obiektu zawierającego te dane (rys. 102).



Rysunek 102: Opcje podglądu przesyłanych żądań dostępne z poziomu zakładki *Network* narzędzi deweloperskich przeglądarki *Google Chrome*

W celu identyfikacji rodzaju błędu posłużono się oknem konsoli dostępnym z poziomu zakładki *Console*. Oferuje ona dokładny opis zaistniałego błędu wraz ze ścieżką lokalizacji do pliku z błędnym fragmentem kodu, a nawet numeru linijki, w której dany błąd wystąpił. W przypadku użycia metod HTTP (*Hypertext Transfer Protocol*) wyświetlany jest odpowiedni numer błędu zgodnie ze specyfiacją kodów błędów HTTP. W oknie konsoli możliwy jest również podgląd wartości przechowywanych przez zmienne w projekcie oraz dostęp do treści ostrzeżeń w trakcie działania aplikacji (rys. 103).



Rysunek 103: Opcje podglądu błędów, ostrzeżeń i wartości zmiennych dostępne z poziomu zakładki *Console* narzędzi deweloperskich przeglądarki *Google Chrome*

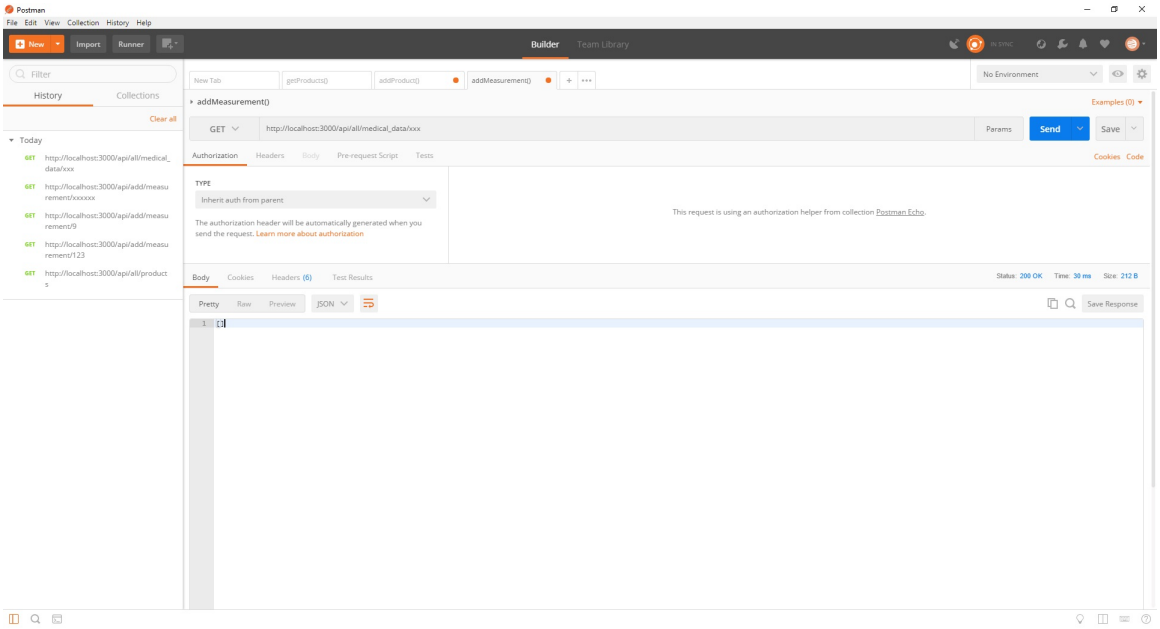
10.2 Testy funkcjonalne aplikacji

Aplikację skonstruowano pod kątem UX (*User Experience*), a kod napisano w oparciu o podstawowe zasady budowania struktury projektu tworzonego z wykorzystaniem frameworku *Angular*

2 [Ans]. Dołożono wszelkich starań, aby ilość danych umieszczanych w ramach jednego widoku nie była zbyt duża, ponieważ w przypadku *Angulara 2* znacząco obniża to wydajność aplikacji.

W ramach testów funkcjonalnych aplikacji przeprowadzono podstawowe czynności mające na celu zapewnienie stabilności jej działania. Zostały one wykonane zgodnie z założeniami funkcjonalnymi, jakie powinien spełniać program. Pierwszym krokiem było sprawdzenie poprawności wysyłanych do serwera żądań (*Requests*) w celu pobrania danych (*Response Data*). Posłużono się oprogramowaniem *Postman*.

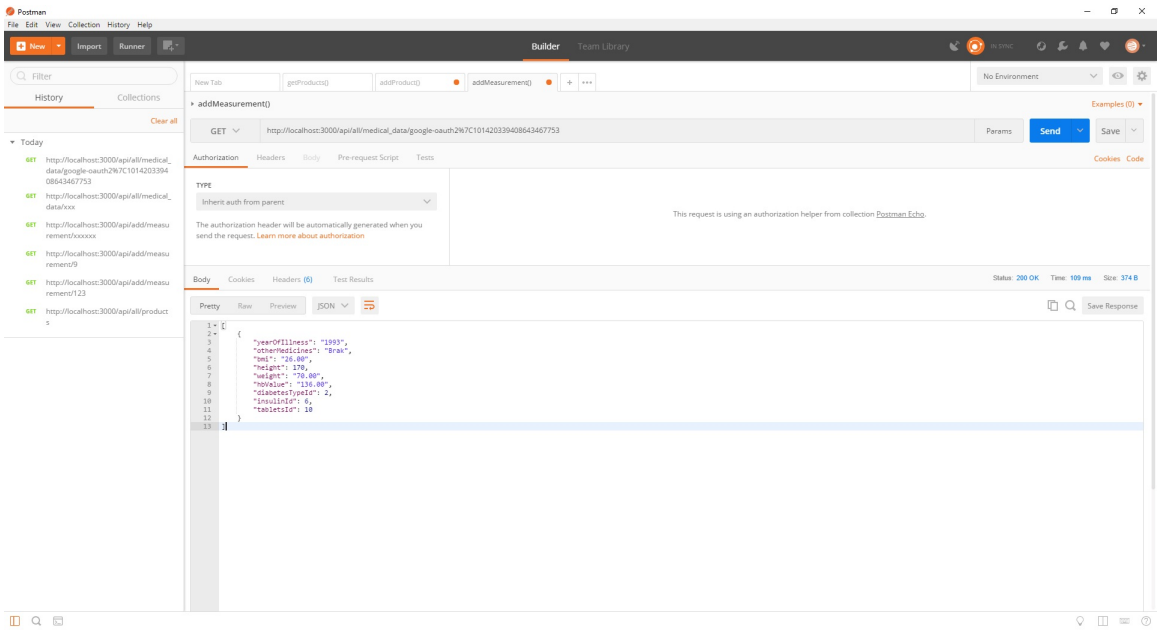
Każda z metod HTTP (*Hypertext Transfer Protocol*) służących do komunikacji z serwerem w celu pobrania lub wysłania danych została sprawdzona poprzez wprowadzenie najpierw błędnych, a następnie poprawnych parametrów w nagłówku, bądź w adresie URL (*Uniform Resource Locator*) metody. Przykładowe wyniki testu przedstawiono na rysunkach 104 oraz 105.



Rysunek 104: Próba pobrania danych medycznych za pomocą oprogramowania *Postman* użytkownika o błędnym identyfikatorze

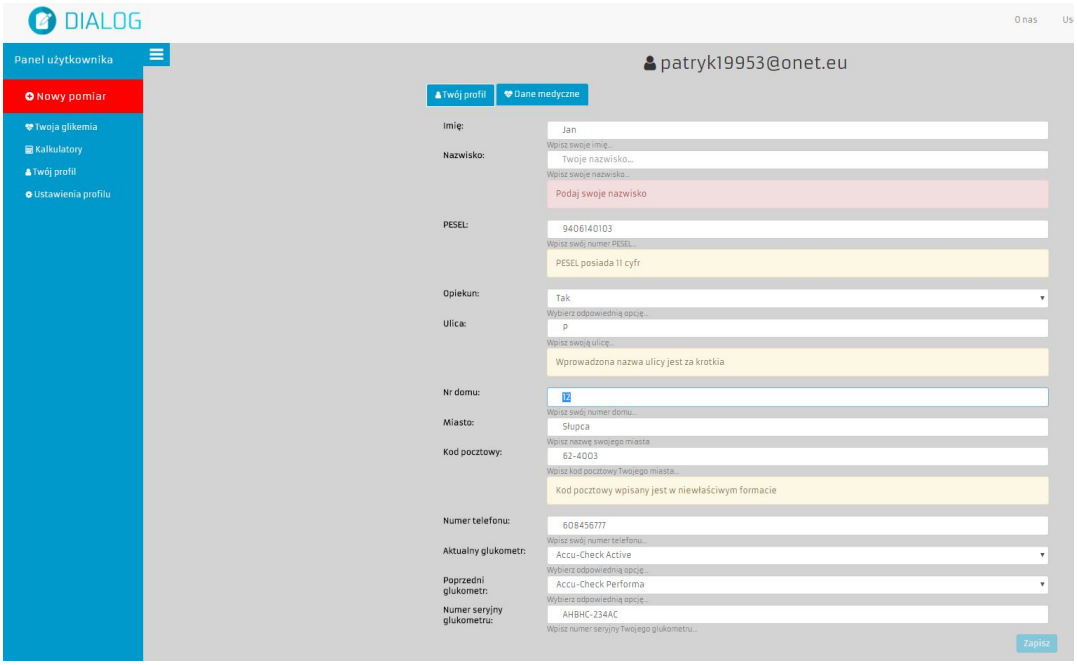
W przypadku wprowadzenia błędnego identyfikatora użytkownika w adresie URL metody GET służącej do pobrania jego danych medycznych uzyskano odpowiedź z serwera z pustą tablicą danych.

Przy próbie wprowadzenia poprawnego identyfikatora w adresie URL tej samej metody serwer zwrócił żądane dane medyczne użytkownika.



Rysunek 105: Próba pobrania danych medycznych za pomocą oprogramowania *Postman* użytkownika o poprawnym identyfikatorze

Kolejną czynnością było sprawdzenie poprawności walidacji danych w formularzach aplikacji. Reguły walidacji odpowiednich pól określone zostały przy użyciu wyrażeń regularnych. Zastosowano odpowiedni przycisk powiązany bezpośrednio z formularzem, służący do wysyłania żądania z uzupełnionymi danymi. Zablokowano go do momentu kiedy użytkownik nie wprowadzi poprawnych danych. Takie podejście zabezpiecza program przed błędami związanymi z niezgodnością typów w bazie danych. Pozwala również uniknąć zapisywania informacji (takich jak kod pocztowy) w niewłaściwym formacie (rys. 106) [Ang].



Rysunek 106: Widok formularza z błędnie wypełnionymi polami i zablokowanym przyciskiem do zapisu ich wartości

Rozdział 11

Zakończenie

Ustosunkowując się do założeń podanych we wstępie pracy, dołożono wszelkich starań, aby oferowane przez aplikacje funkcjonalności działały zgodnie z ich przeznaczeniem oraz zaprezentowane były za pomocą prostego i przejrzystego interfejsu. Aplikacja ma być jedynie jednym z czynników przyczyniających się do postępów w leczeniu cukrzycy.

Poprzez oferowanie czytelnych wykresów podsumowujących wszystkie, dotychczasowe pomiary wprowadzone przez użytkownika oraz tabeli zestawień grupującej wartości pomiarów osoba korzystająca z aplikacji wyciągać ma poglądowe wnioski na temat tego jak przebiega jej choroba od momentu założenia konta w systemie. Ponadto, dzięki dostępnym kalkulatorom BMI (*Body Mass Index*), wymienników węglowodanowych czy wymienników białkowo-tłuszczowych opartych na stale rozwijanej przez użytkowników bazie produktów i danych wprowadzanych przez korzystającą z nich osobę możliwe jest określenie jaki stopień współczynnika masy ciała posiada dany użytkownik czy też jakie produkty mogą być spożyte w danej chwili tak, aby nie zakłócić porządku dziennej diety i nie pogłębić stopnia zaawansowania cukrzycy.

Jeżeli chodzi o dalsze etapy rozwoju aplikacji – autor chciałby rozszerzyć jej funkcjonalność o moduł oparty na zdobywaniu punktów przez użytkowników w zamian za postępy w leczeniu cukrzycy. Funkcjonalność ta mogłaby mieć znaczący wpływ na motywację i sukcesywne korzystanie z aplikacji w zamian za systematyczne prowadzenie pomiarów. Dodatkowo, jest w planach dodanie modułu skanowania kodów kreskowych produktów przez użytkowników za pomocą kamerki internetowej, bądź kamery telefonu i automatycznego dodawania ich do bazy danych systemu. Ułatwiłoby to znacznąco proces wprowadzania nowego produktu.

Literatura

[Ang] Angular - Reactive Forms guide. [on-line 11.11.2017] `https://angular.io/guide/reactive-forms`.

[Ans] Angular - Setup for local development guide. [on-line 13.08.2017] `https://angular.io/guide/setup`.

[Aut] Auth0 documentation. [on-line 01.10.2017] `https://auth0.com/docs/quickstart/spa/angular2`.

[Fra15] Ben Frain. *Responsive Web Design with HTML5 and CSS3 — Second Edition.* Packt Publishing Ltd., Birmingham, 35 Livery Street B3 2PB, UK, 2015.

[Vsc] Visual Studio Code documentation. [on-line 08.09.2017] `https://code.visualstudio.com/docs`.

Dodatek A

Załączniki

Płyta CD z następującą zawartością:

- tekst pracy w formacie PDF,
- projekt z plikami źródłowymi.