

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 ANALYSIS 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
Bezpieczeństwo systemów informatycznych			
Specjalność:			

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
Student:			Figiel

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 promotorata:	Instytut Automatyki i Inżynierii Informatycznej

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

podpis dyrektora/kierownika jednostki organizacyjnej promotorata

PRODZIEKAN
Wydziału Elektrycznego
Politechniki Poznańskiej
A. Tomczewski
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 posłużyły się dokumentacjami technicznymi poszczególnych zestawów narzędzi. Przeanalizowano mechanizmy bezpieczeństwa oferowane przez framework. 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 and conclusions were summarized and discussed.

Spis treści

1 Wstęp	1
1.1 Wprowadzenie	1
1.2 Cel projektu	1
1.3 Struktura pracy	1
2 Technologie	3
2.1 Wprowadzenie	3
2.2 Technologie	3
2.2.1 Framework Angular	3
2.2.2 TypeScript	3
2.2.3 Lodash	4
2.2.4 HTML5	4
2.2.5 CSS3	4
2.2.6 Bootstrap 4	4
2.2.7 Node.js	5
2.2.8 Express.js	5
2.2.9 Karma	5
2.2.10 Jasmine	5
2.2.11 Apache Cordova	5
2.2.12 LaTeX	6
2.2.13 PBKDF2	6
2.2.14 Argon2	6
2.3 Narzędzia	6
2.3.1 Visual Studio Code 1.28	6
2.3.2 github.com	6
2.3.3 Auth0	7
2.3.4 TexStudio	7
3 Node.js	9
3.1 Wprowadzenie	9
3.2 Implementacja aplikacji serwerowej	10
3.2.1 Inicjalizacja serwera	10
3.2.2 Implementacja modułu rejestracji nowego użytkownika	11
3.2.3 Implementacja modułu logowania użytkownika	13
3.2.4 Implementacja modułu wylogowywania użytkownika	14
3.2.5 Implementacja mechanizmów podtrzymywania sesji użytkownika	14
4 Baza danych	15
4.1 Opis tabeli bazy danych	15
4.1.1 Person	15
4.1.2 Medical Data	15
4.1.3 Glucometer	16
4.1.4 Medical Data has Tablets	16
4.1.5 Tablets	16
4.1.6 Medical Data has Insulin	16
4.1.7 Diabetes Type	16
4.1.8 Insulin	17
4.1.9 Measurement	17

4.1.10 Measurement has Insulin	17
4.1.11 Glycemia Ranges	17
4.1.12 Products	17
4.2 Model bazy danych	18
5 Panel administratora	21
5.1 Logowanie do panelu administratora	21
5.2 Użytkowanie panelu administratora	22
6 Panel użytkownika zarejestrowanego	25
6.1 Logowanie do panelu użytkownika	25
6.2 Użytkowanie panelu użytkownika zarejestrowanego	25
6.2.1 Twoja glikemia	25
6.2.2 Dodaj pomiar	29
6.2.3 Kalkulatory	29
6.2.4 Twój profil	32
6.2.5 Ustawienia profilu	34
7 Panel użytkownika niezarejestrowanego oraz niezalogowanego	35
7.1 Okno rejestracji użytkownika	35
7.1.1 Rejestracja przy użyciu konta <i>Google</i>	35
7.1.2 Rejestracja tradycyjna – podanie adresu e-mail oraz hasła	36
8 Bezpieczeństwo aplikacji	39
9 Testy	41
9.1 Debugowanie aplikacji i dostęp do informacji o jej działaniu	41
9.2 Testy funkcjonalne aplikacji	43
10 Zakończenie	45
Literatura	47
A Załączniki	49

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 ciągły rozwój technologii informatycznych (zwłaszcza tych internetowych) powodują stale 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 użytkowych bibliotek i rozwiązań, które w znacznym stopniu ułatwiają implementację najważniejszych mechanizmów komunikacji poszczególnych warstw aplikacji z serwerem, dlatego też postanowilem 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 Struktura pracy

Praca składa się z siedmiu rozdziałów. Pierwszy z nich zawiera wprowadzenie, przedstawia cel projektu oraz ogólną strukturę pracy. W drugim opisane są technologie wykorzystane do tworzenia testowej aplikacji i analizy wybranych zagadnień. Trzeci rozdział opisuje środowisko backendowe *Node.js*. Przedstawia on sposób implementacji i uruchomienia serwera z wykorzystaniem bezpiecznego połączenia - HTTPS. Kolejny, czwarty, opisuje mechanizmy bezpieczeństwa aplikacji internetowych opartych na języku *JavaScript*. W rozdziale piątym przeanalizowano i zestawiono ze sobą wybrane technologie implementacji warstwy prezentacji. Rozdział szósty zawiera analizę hybrydowego frameworka *Ionic* i opisuje wybrane mechanizmy bezpieczeństwa, oferowane przez funkcje natywne biblioteki. Ostatni, siódmy rozdział jest podsumowaniem całej pracy. 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

Angular jest to frontendowy zestaw bibliotek, który służy do implementacji stron typu SPA (*Single Page Application*). Cały DOM (*Document Object Model*) strony ładowany jest wówczas jednorazowo w trakcie jej uruchomienia. Dzięki takiemu rozwiązaniu można w wygodny sposób korzystać ze strony internetowej bez jej ciągłego przeładowywania. *Angular* oferuje bardzo proste mechanizmy manipulacji DOMem. Dzięki instrukcjom sterującym, takim jak *ngIf*, *ngFor* itp. możemy w dowolny sposób sterować treścią wyświetlaną na stronie. Kolejnym udogodnieniem oferowanym przez framework *Angular* jest mechanizm *Data binding*, dzięki któremu dowolna zmiana zmiennej z poziomu szablonu powoduje automatyczne odświeżanie jej wartości w przeglądarce internetowej. Cała struktura *Angulara* wymusza u programistów stosowanie dobrych praktyk pisania kodu i pomaga usystematyzować na pozór skomplikowaną architekturę projektu. Kod pisany jest w języku *TypeScript*, który kompilowany (transpilowany) jest do wynikowego kodu *JavaScript*. Pomimo tego, że mechanizm kompilacji spowalnia nieco działanie aplikacji, to dostarcza on szereg udogodnień oferowanych przez *ECMAScript*, takich jak na przykład typowanie. Typowanie pozwala w bardzo prosty sposób zapanować nad kodem i znacznie sprawniej lokalizować ewentualne błędy. Kolejnym z udogodnień jest biblioteka *RxJS* (*Reactvie Extensions for JavaScript*), która oferuje zestawy narzędzi do programowania asynchronicznego czy opartego na zdarzeniach. Ogromną zaletą *Angulara* jest jego wieloplatformowość - pozwala on bowiem na tworzenie aplikacji webowych, aplikacji PWA (*Progressive Web App*), aplikacji mobilnych, korzystających z funkcji natywnych urządzeń (framework *Cordova*) czy też aplikacji desktopowych z wykorzystaniem frameworka *Electron*. [Fre18]

2.2.2 TypeScript

TypeScript to zorientowany obiektowo, oparty na klasach nadzbiór języka *Javascript* stworzony przez firmę *Microsoft*. Oznacza to, że kod pisany w *TypeScriptie* kompilowany (transpilowany) jest do wynikowego kodu *JavaScript*, którego wersję ustalić może sam użytkownik. Kod pisany w tym języku jest znacznie przejrzysty i łatwiejszy do przeanalizowania. Współpracuje on z różnymi mechanizmami podpowiedzi, takimi jak *IntelliSense*, przez co programowanie staje się znacznie łatwiejsze. *TypeScript* obsługuje pliki nagłówkowe, dodające informacje o typie do istniejących już bibliotek *JavaScript*. Dzięki takiemu rozwiązaniu takie biblioteki jak *jQuery* czy *Node.js* można używać bez żadnych przeszkód. Język ten oferuje również dodatkowe mechanizmy programistyczne, takie jak klasy, interfejsy, moduły, zmienne typowanie, argumenty i funkcje czy też opcjonalne parametry funkcji. *TypeScript*, poprzez zastosowanie m.in. silnego typowania znacząco przyspiesza pracę programistów i eliminuje pojawiające się błędy, przez co z roku na rok nabiera on coraz większą rzeszę zwolenników. [Fen18]

2.2.3 Lodash

2.2.4 HTML5

Język HTML5 (*HyperText Markup Language*) jest najnowszą wersją popularnego standardu opisującego język HTML. Wprowadza on nowe elementy, atrybuty, zachowania i znacznie większy zestaw nowoczesnych technologii, umożliwiających budowanie bardziej zróżnicowanych i zaawansowanych technologicznie stron internetowych. Według powszechniej opinii jest to jedna z najlepszych rzeczy jakie spotkały dzisiaj Internet. Jedną z najbardziej spektakularnych zmian w najnowszej wersji języka HTML jest natywna obsługa audio (znacznik `<audio>`) oraz wideo (znacznik `<video>`) bez użycia modułu *Flash*. Ponadto, oferuje on dużo prostsze tworzenie i wyświetlanie grafiki przy użyciu znaczników `<canvas>`. Kolejnym udogodnieniem jest możliwość otwierania stron internetowych bez aktywnego połączenia z Internetem. Ponadto, została dodana opcja edycji dokumentów online (stosowana m.in. przez firmę *Google*) czy też mechanizm fizycznego przenoszenia plików do okna przeglądarki internetowej zwany także jako *Drag and Drop*. HTML5 wprowadza znaczniki (`<header>`, `<section>`, `<footer>`) ułatwiające ustrukturyzowanie elementów strony internetowej. Współpracuje on z wieloma rodzajami przeglądarek, jednakże nie wszystkie z nich mogą zapewnić stu procentową kompatybilność z tym językiem. Poprzez współpracę i interakcję z wieloma powszechnie znymi technologiami tworzenia stron internetowych HTML5 często jest zwany jako "HTML5 i inne powiązane standardy". Sama składania języka nie jest zbyt skomplikowana. Kod można tworzyć w najprostszym edytorze tekstu. Warto także wspomnieć o tym, że HTML5 obsługuje tzw. *Web-Workers*, które umożliwiają wielowątkową obsługę przepływu danych. [Car17]

2.2.5 CSS3

CSS (*Cascading Style Sheets*) jest to framework służący do zwiększenia funkcjonalności, wydajności i poprawy wizualnej witryn WWW. Nie jest to *strict* język programistyczny - wymaga od swoich użytkowników myślenia typowo abstrakcyjnego oraz kreatywności. Pozwala on na tworzenie interfejsu GUI (*Graphical User Interface*) użytkownika przy zachowaniu wysokiej szybkości działania aplikacji i jednocześnie lekkości pisanej kodu. CSS używany jest w połączeniu z językiem HTML, który nadaje stronie określona strukturę. Kaskadowe arkusze stylów w wersji trzeciej wprowadzają szereg udogodnień do nadawania wizualnej szaty stronom internetowym. Są nimi m.in. pozycjonowanie zawartości strony, ramki dla elementów, nowe właściwości dla tła, gradienty, cienie, transformacje 2D oraz 3D, lekkie i wydajne animacje, wielokolumnowy *layout*, rewolucyjny *Flex Box* czy przydatne przy tworzeniu responsywnych stron internetowych *Media Queries*, pozwalające dopasowywać zawartości witryn do różnych rozdzielczości ekranów. Jest to najbardziej powszechny i znany na świecie język tworzenia UI (*User Interface*) stron WWW. Bariera wejście w świat CSS jest zaskakująco niska. Należy jednak pamiętać o tym, że złożoność jego kodu rośnie wprost proporcjonalnie do wielkości projektu, w którym jest stosowany. Istnieje wiele preprocesorów języka CSS, które w znacznym stopniu ułatwiają jego stosowanie poprzez narzucanie określonej struktury i wprowadzanie możliwości tworzenia zmiennych, domieszek, operacji czy funkcji. Dwa z nich, najczęściej używane przez programistów to LESS (*Leaner CSS*) oraz SASS (*Syntactically Awesome Style Sheets*). [Gra18]

2.2.6 Bootstrap 4

Bootstrap to otwarte narzędzie (framework) do tworzenia stron WWW, przy użyciu języka HTML, CSS i *JavaScript*, stworzone przez programistów popularnego serwisu *Twitter*. Używany jest do tworzenia responsywnych witryn internetowych w podejściu *Mobile First* kładącym duży nacisk na dostosowywanie widoku strony do urządzeń mobilnych. Umożliwia projektowanie stron WWW opartych na wierszowo-kolumnowym systemie siatek (twz. *Grid*) mającym na celu usystematyzowanie położenia elementów witryny. Jest całkowicie darmowy do pobrania i codziennego użytku. W wersji czwartej frameworka dołożone zostały nowe komponenty. Przypieszono również znacząco arkusze stylów i polożono w niej jeszcze większy nacisk na responsywność. *Bootstrap 4* jest wspierany przez wszystkie najnowsze wersje przeglądarek internetowych. W porównaniu z poprzednią, trzecią wersją *Bootstrap 4* wprowadza m.in. prostą możliwość zmiany wielkości nagłówka H1, tzw. *Flexbox* czyli nowy sposób układania elementów na stronie, nowy wygląd przycisków, karty, łatwiejszy sposób zmiany tła paska nawigacji czy łatwiejsze manipulowanie marginesami elementów. Jest to jednak wierzchołek góry lodowej jeżeli chodzi o zaistniale zmiany. Możliwość

edytowania i bezpośredniego wpływu na każdy z elementów biblioteki czyni ją niezastąpionym narzędziem w jeszcze szybszym i bardziej kreatywnym tworzeniu stron WWW. [BJ16]

2.2.7 Node.js

Node.js jest w pełni darmowym, otwartym, opartym na języku *JavaScript* uruchomieniowym środowiskiem dla aplikacji serwerowych. Działa na wielu platformach, takich jak *Windows*, *Linux* czy *MacOS*. Bazuje na silniku *Google V8 JavaScript engine* napisanym w języku C++, pracującym w połączeniu z mechanizmem jednowątkowej pętli zdarzeń i niskopoziomowym interfejsem wejść/wyjść API (*Application Programming Interface*). Silnik V8 stworzony przez firmę *Google* pozwala na komplikowanie kodu *JavaScript* do niskopoziomowego języka maszynowego. *Node.js* wprowadza możliwość obsługi dużej ilości zapytań co znacząco wpływa na skalowalność aplikacji, w których jest on wykorzystywany. Środowisko oparte jest na opracowanym przez firmę *Ecma International* standardzie *ECMAScript*, na którym bazuje język *JavaScript*. *Node.js* jest mocno ustandaryzowany. Jego popularność wynika z tego, że język, na którym jest oparty jest w dzisiejszych czasach powszechnie stosowany. Dodatkowo, współpracuje z wieloma dobrze dopracowanymi frameworkami, takimi jak *Express.js* co czyni go jeszcze bardziej niezastąpionym. Umożliwia on sprawne zarządzanie instalowanymi paczkami wraz z ich zależnościami dzięki dostarczanemu, dedykowanemu oprogramowaniu NPM (*Node Package Manager*). Dzięki takiemu rozwiązaniu można w bardzo szybki i prosty sposób znaleźć odpowiadającą nam bibliotekę, przeszukując powszechnie dostępne, internetowe repozytorium NPM. Aplikacje *Node.js* mogą być uruchamiane zarówno na urządzeniach IoT (*Internet of Things*), jak i na powszechnie wykorzystywanych urządzeniach mobilnych, pracujących na procesorach architektury ARM (*Advanced RICS Machine*). W dzisiejszych czasach środowisko to jest używane przez wielu gigantów biznesu internetowego, takich jak *Google*, *Microsoft*, *Netflix*, *PayPal* czy *LinkedIn*. [Dog18]

2.2.8 Express.js

Express.js jest darmowym, otwartym frameworkiem przeznaczonym do współpracy ze środowiskiem serwerowym *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 frameworku *Node.js*. *Express.js* pozwala definiować tabele routingu w celu wykonywania różnego typu akcji w oparciu o metodę HTTP. Umożliwia ponadto dynamiczne renderowanie stron napisanych w języku HTML w oparciu o przekazywanie argumentów do szablonów. [Aug15]

2.2.9 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*.

2.2.10 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*).

2.2.11 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*.

2.2.12 LaTeX

Oprogramowanie służące do tworzenia przejrzystych dokumentów tekstowych takich jak na przykład książka czy artykuł. 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.

2.2.13 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 używania go przy implementacji nowych projektów.

2.2.14 Argon2

Algorytm *Argon2* został zwycięzcą *Password Hashing Competition* i jako następca *BCrypt* oraz *Scrypt* jest obecnie zalecany do zabezpieczania hasel. *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 biblioteki, które znaczco ułatwiają używanie *Argon2* w implementacjach. Algorytm występuje w dwóch bazowych wersjach - *Argon2i* oraz *Argon2d* chroniącym przed atakami typu *GPU cracking*. Istnieje również hybrydowe połączenie dwóch rodzajów funkcji zwane jako *Argon2id*. *Argon2* współpracuje z 64-bitowymi architekturami procesorów i powinien być komplikowany na systemach takich jak *Linux*, *OS X* czy *Windows*.

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, *VSC* 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 podłożę w języku *Erlang* z wykorzystaniem frame-

worka *Ruby on Rails*. Github umożliwia darmowy hosting plików oraz płatne, prywatne repozytoria. Platforma oferuje szereg statystyk powiązanych z implementowanym kodem źródłowym, mechanizm typu *bugtracker*, możliwość 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 bezpośredni kontroli nad tworzonym kodem oprogramowania i dostęp do historii pracy co znaczco ułatwia zarządzanie projektem programistycznym.

2.3.3 Auth0

Serwis *Auth0* umożliwia połaczenie 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 późniejszego 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

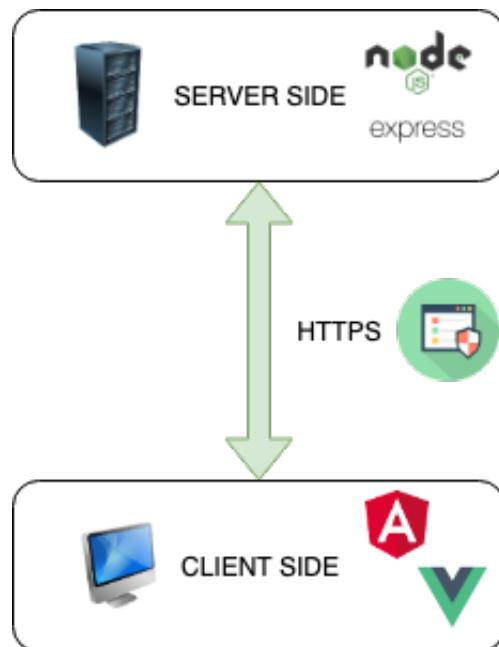
Node.js

3.1 Wprowadzenie

ToDo:

- Edycja rysunku i zmiana używanego protokołu z HTTP na HTTPS, dodanie kłódki, jakiś grafik pokazujących że połączenie jest szyfrowane,
- Wstawienie kodu jako obrazki

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 3.2.2.



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 do celów testowych klucza i certyfikatu auto-podpisanej SSL (*Secure Socket Layer*), który używany jest przez protokół HTTPS (*Hyper-text Transfer Protocol Secure*) w celu zwiększenia bezpieczeństwa danych. Kolejnym krokiem jest zdefiniowanie portu, na który będą kierowane żądania klienta i rozpoczęcia nasłuchiwanego. W tym celu posłużyono się frameworkm *Express.js* i zdefiniowano lokalny serwer HTTPS pracujący na porcie numer 9000.

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

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

Wygenerowany klucz RSA zawiera 2048 bitów. Klucze mniejsze niż 2048 bitów nie są obecnie uważane za bezpieczne. Wersje 2048 bitowe mają wystarczającą liczbę unikalnych kodów szyfrowania.

Sama, wstępna implementacja serwera HTTPS nasłuchującego na porcie o numerze 9000 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 kolejnym 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* w oknie terminala.

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

Po uruchomieniu serwera z poziomu konsoli otrzymano komunikat widoczny na rysunku 32. Oznacza on, że serwer został poprawnie uruchomiony i rozpoczęto nasłuchiwanie:

```
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/krzysztofigiel/Desktop/Praca Magisterska/Angular/Angular-Security/angular-security
> ts-node ./server/server.ts --secure
```

Rysunek 32: Zrzut ekranu konsoli systemu *MacOS* ukazujący komunikaty o poprawnym uruchomieniu procesu nasłuchiwanego 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:

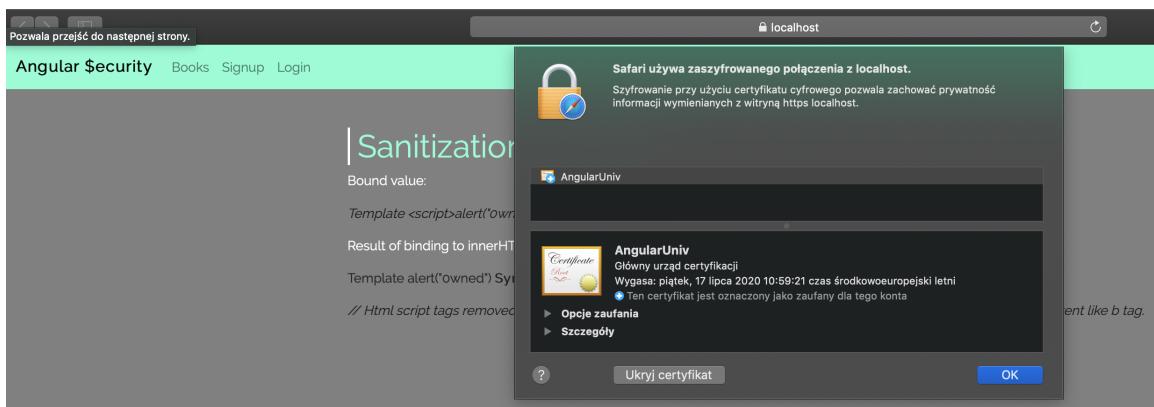
- `/api/books`
- `/api/signup`
- `/api/user`
- `/api/logout`
- `/api/login`

Dane zawarte w tabelach routingu umożliwiają

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

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

Po poprawnym uruchomieniu aplikacji działającej na serwerze, włączonej na porcie lokalnym `https://localhost:4200` otrzymano informację o tym, że przeglądarka używa zaszyfrowanego połączenia z `localhost`:



Rysunek 33: Okno przeglądarki internetowej *Safari* ukazujące informację o poprawnym zaszyfrowaniu lokalnego połączenia aplikacji na porcie 4200 przy użyciu certyfikatu autopodpisanej.

3.2.2 Implementacja modułu rejestracji nowego użytkownika

W kolejnym 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 przekazaniu odpowiednich statusów metody HTTP do klienta (`res`) w zależności czy weryfikacja nowego użytkownika przebiegła pomyślnie czy też nie.

Do walidacji haseł użytkownika posłużono się validatorem z repozytorium NPM (*Node Package Manager*) - *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:

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

```

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

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

```

W przypadku podania hasła niezgodnego z przyjętymi regułami tworzenia nowych haseł użytkownika serwer odpowiada komunikatem z kodem odpowiedzi HTTP o numerze 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 z kodem odpowiedzi HTTP o numerze 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 the same email address: ' + email;
  console.error(message);
  throw new Error(message);
}

```

Jeżeli użytkownik istnieje w bazie danych wyświetlany jest stosowny komunikat. Serwer wysyła również do klienta odpowiedź z kodem statusu HTTP o numerze 500. Cały mechanizm zapobiega tworzeniu duplikatów użytkowników w bazie danych i uniemożliwia podszywanie się pod inne osoby widniejące w systemie. W przypadku braku podobnego konta i pomyślnej weryfikacji tworzony i zwracany jest nowy obiekt `user` a licznik użytkowników jest inkrementowany:

```

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

Po zaimplementowaniu modułu rejestracji użytkownika stworzono moduł logowania. W tym celu, na samym początku dodano wpis do tabeli routingu z następującą ścieżką:

```

app.route('/api/login')
.post(login);

```

W kolejnym kroku zaimplementowano funkcję `login(req: Request, res: Response)` odpowiedzialną za procedurę logowania. W środku funkcji następuje sprawdzanie, czy podane w parametrze dane użytkownika występują w bazie danych czy też nie (metoda `db.findUserByEmail`). W przypadku, gdy osoby nie ma w bazie zwracany jest kod HTTP o numerze 403 ze stosowną informacją zwrotną.

Jeżeli użytkownik widnieje w systemie wywoływana jest metoda `loginAndBuildResponse`, zwracająca w przypadku poprawnego logowania odpowiedni kod odpowiedzi HTTP o numerze 200. Wówczas następuje dodanie tokena sesji o nazwie *SESSIONID* do zawartości *cookie* przeglądarki internetowej (Rys. 35):

Application	Name	Value	Domain	Path	Exp...	Size	HTTP	Secure
Manifest	SESSIONID	fdea4a46e3d7d05cd83763ead6650f7a...	localhost	/	196...	73	✓	✓
Service Workers	_stripe_mid	ea93000d-b766-4e67-8034-a31aad1fe...	.fontawesome.c...	/	202...	48		
Clear storage	_utmx	226539914.758GtUaVSj2DQs9XNDlst...	.fontawesome.c...	/	202...	42		
Storage	_utmxx	226539914.758GtUaVSj2DQs9XNDlst...	.fontawesome.c...	/	202...	60		
Local Storage	_ga	GA1.2.486858816.1543438593	.bootstrapcdn.c...	/	202...	29		
Session Storage	_ga	GA1.2.1992104818.1543438622	.fontawesome.c...	/	202...	30		
IndexedDB								
Web SQL								
Cookies								
https://localhost:4200								

Rysunek 34: Okno narzędzi deweloperskich (zakładka *Application/Cookies*) przeglądarki internetowej *Google Chrome* z dodanym tokenem sesyjnym *SESSIONID* użytkownika

Sama implementacja tworzenia tokena sesji użytkownika i zapisywania go do ciasteczka przeglądarki wygląda następująco:

```

try {
  const sessionToken = await attemptLogin(credentials, user);
  console.log("Login successfull");
  res.cookie("SESSIONID", sessionToken, { httpOnly: true, secure: true });
  res.status(200).json({ id: user.id, email: user.email });
} catch (error) {
  console.log("Login failed");
  res.sendStatus(403);
}

```

Dzięki temu połączenie jest dodatkowo uwierzytelniane, a użytkownik ma wyłączność na korzystanie z własnej sesji danej witryny internetowej. Dodatkowo, token zapewnia podtrzymanie sesji użytkownika na pewien odstęp czasu (*timestamp*). Więcej informacji na temat działania tokenu sesji zawarto w rozdziale czwartym dotyczącym mechanizmów bezpieczeństwa aplikacji internetowych opartych na języku *JavaScript*.

3.2.4 Implementacja modułu wylogowywania użytkownika

Moduł wylogowywania również posiada własną ścieżkę w tabeli routingu. Jej implementacja wygląda następująco:

```

app.route('/api/logout')
.post(logout);

```

Jego działanie zaimplementowano w prosty sposób - całość opiera się na usunięciu tabeli ciasteczek sesyjnych (*cookie*) w przeglądarce. Wiąże się to z usunięciem informacji o sesji użytkownika. Przy odpowiednim obsłużeniu wyniku działania tego modułu z poziomu warstwy aplikacji można w bardzo prosty sposób zablokować dostęp do treści strony, po ówczesnym wylogowaniu. Środowisko *Node.js* umożliwia szybkie wyczyszczenie tabeli *cookie* za pomocą metody *clearCookie()*:

```
res.clearCookie('SESSIONID');
```

Czyszczenie zawartości tabeli ciastek sesyjnych przeglądarki jest konieczne z punktu widzenia bezpieczeństwa. Zapobiega to przechwyceniu informacji przez osoby trzecie, np. za pomocą ataku XSS (*Cross-site scripting*). Mimo wszystko, nawet po uzyskaniu zawartości ciasteczka sesyjnego osoba atakująca napotka problem rozszыfrowania zawartych tam danych, które przechowywane są w postaci funkcji skrótu *Argon2*.

Name	Value
SESSIONID	fdea4a46e3d7d05cd83763ead6650f...

Rysunek 35: Okno narzędzi deweloperskich (zakładka *Application/Cookies*) przeglądarki internetowej *Google Chrome* z dodanym tokenem sesyjnym *SESSIONID* użytkownika przechowywanym za pomocą funkcji skrótu *Argon2*

3.2.5 Implementacja mechanizmów podtrzymywania sesji użytkownika

Rozdział 4

Baza danych

4.1 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.

4.1.1 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 posiadanej 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.

4.1.2 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.

4.1.3 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.

4.1.4 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.

4.1.5 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.

4.1.6 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.

4.1.7 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.

4.1.8 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.

4.1.9 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.

4.1.10 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.

4.1.11 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.

4.1.12 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 tłuszczy, 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.

4.2 Model bazy danych

Rysunek 41 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 obydwiu 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 konwencji *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  
}
```

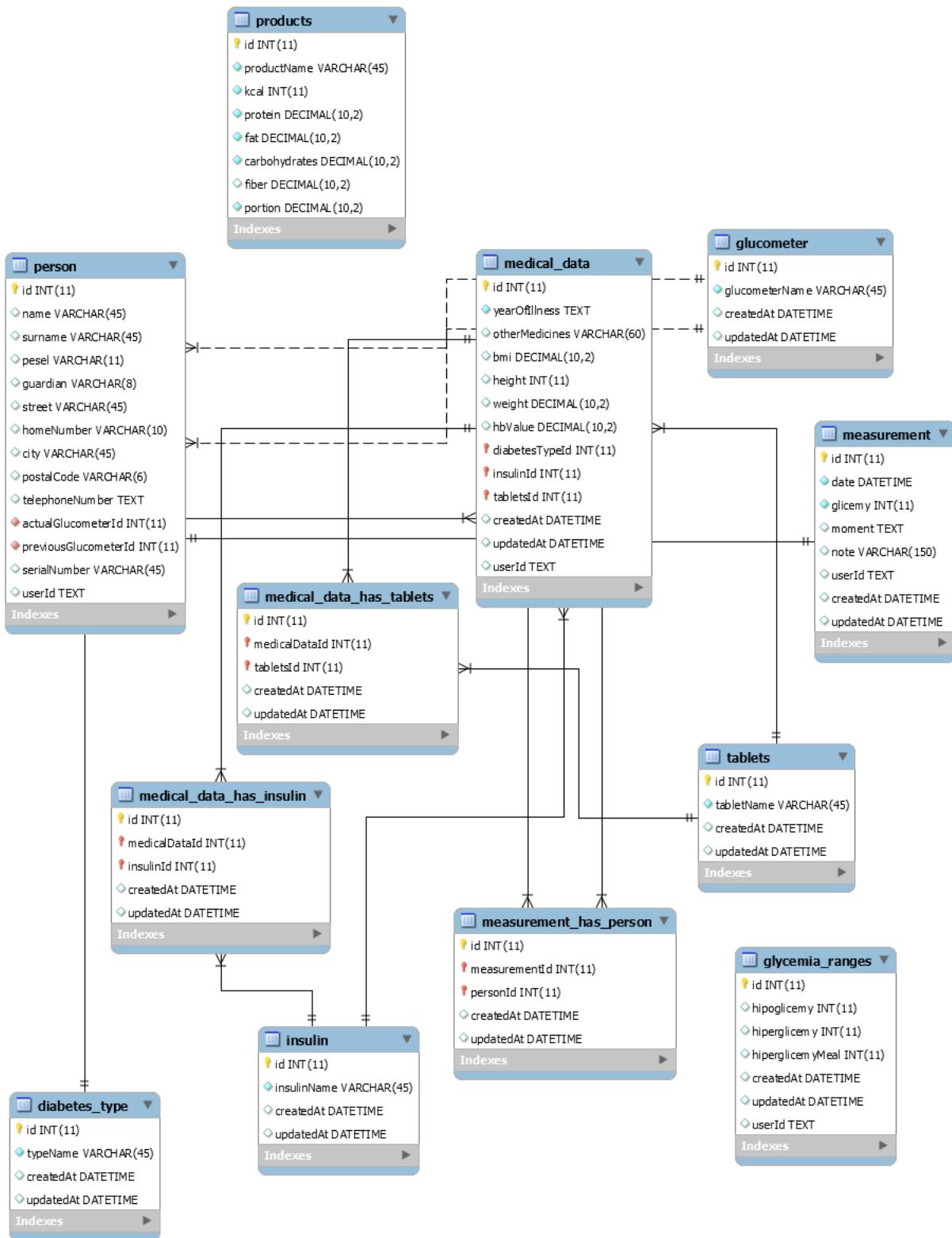
Ograniczenia jakie zostały użyte do zdefiniowania klucza głównego to:

```
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.



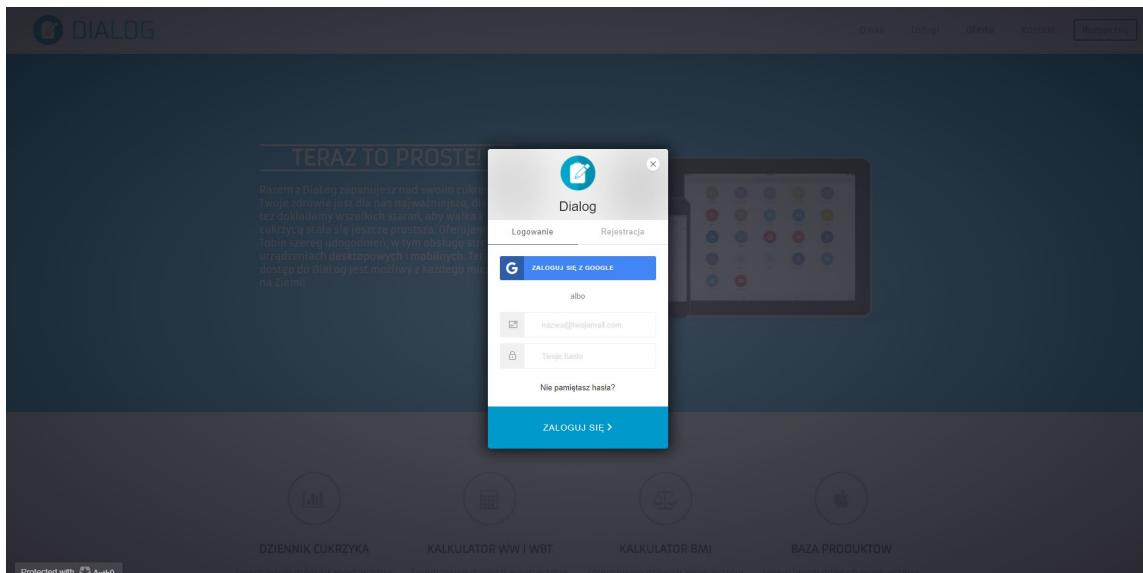
Rysunek 41: Diagram EER bazy danych aplikacji

Rozdział 5

Panel administratora

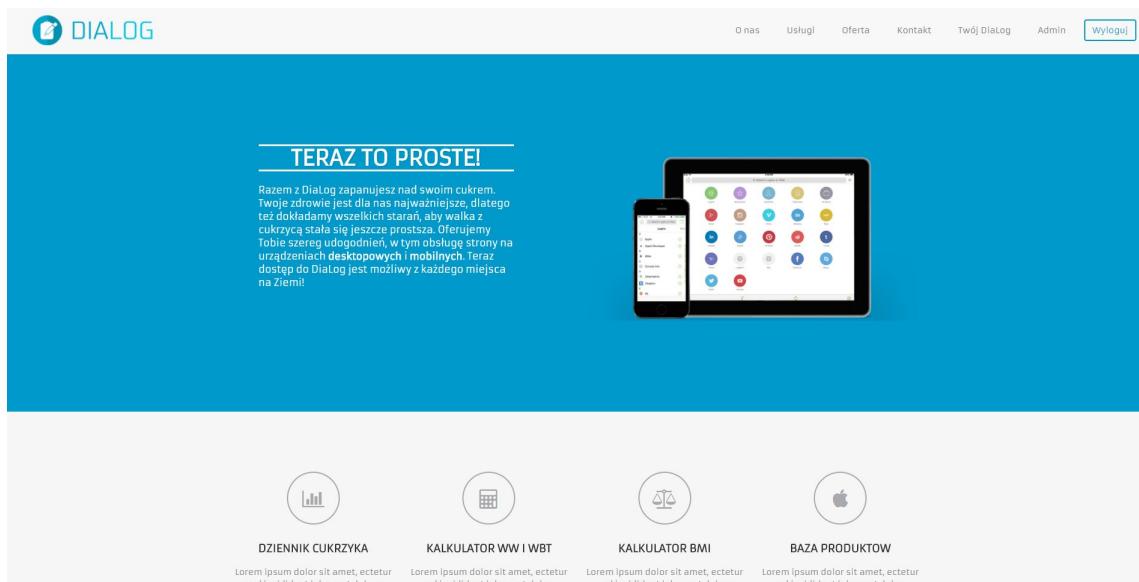
5.1 Logowanie do panelu administratora

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 51 przedstawia okno logowania administratora.



Rysunek 51: 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 52.



Rysunek 52: Okno główne aplikacji z dostępną opcją panelu administratora

5.2 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 53. 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.

The screenshot shows the administrator panel. At the top left is the logo 'DIALOG'. The top right contains a navigation bar with links: 'O nas', 'Usługi', 'Oferta', 'Kontakt', 'Twój DiaLog', 'Admin' (which is highlighted in blue), and 'Wyloguj'. Below the navigation bar is a table titled 'Tabela produktów' (Product Table). The table has columns: 'Nazwa produktu', 'Kalorie [kcal]', 'Białko [g]', 'Tłuszcze [g]', 'Węglowodany [g]', 'Błonnik [g]', 'Porcja [g]', and 'Czynność'. The table lists various products with their nutritional values and a 'Usuń' (Delete) button in each row. At the bottom of the table are pagination controls: '1 2 Next >'

Nazwa produktu	Kalorie [kcal]	Białko [g]	Tłuszcze [g]	Węglowodany [g]	Błonnik [g]	Porcja [g]	Czynność
Snickers	250	20.00	15.00	60.00	10.00	100.00	<button>Usuń</button>
Morenda	50	2.00	3.00	30.00	10.00	100.00	<button>Usuń</button>
Czekolada mleczna Milka	530	6.00	29.00	58.00	4.00	100.00	<button>Usuń</button>
Batonik MilkyWay	454	3.80	17.00	71.60	2.10	100.00	<button>Usuń</button>
Czekolada mleczna Goplana	530	8.00	30.00	56.00	4.00	100.00	<button>Usuń</button>
Czekolada mleczna Wedel	530	6.00	30.00	58.00	2.50	100.00	<button>Usuń</button>
Czekolada gorzka Wedel	502	8.40	32.00	39.00	13.00	100.00	<button>Usuń</button>
Czekolada gorzka Goplana	516	7.90	32.00	42.00	3.20	100.00	<button>Usuń</button>
Bułka pszenna	277	8.10	1.50	57.70	1.80	100.00	<button>Usuń</button>
Baton Mars	452	3.70	17.00	70.30	0.10	100.00	<button>Usuń</button>

Rysunek 53: Panel administratora aplikacji

Tabela podzielona jest na osiem kolumn:

1. *Nazwa produktu,*
2. *Kalorie [kcal],*
3. *Białko [g],*
4. *Tłuszcze [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.

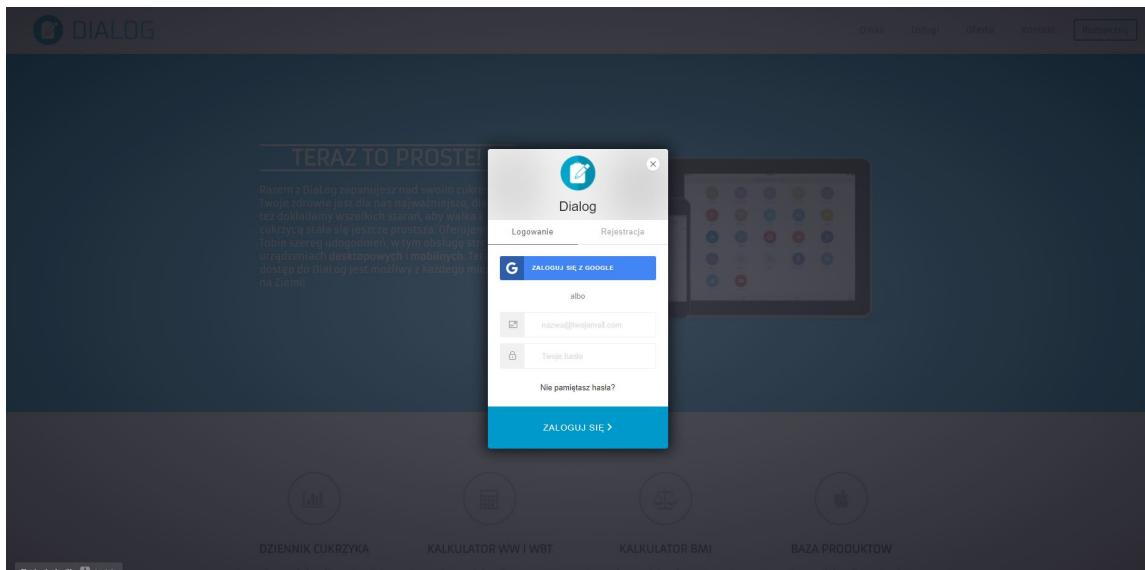
Administrator, tak jak każdy zalogowany użytkownik, ma również możliwość korzystania ze wszystkich funkcjonalności systemu przeznaczonych dla tej grupy użytkowników – zakładka *Twój DiaLog*.

Rozdział 6

Panel użytkownika zarejestrowanego

6.1 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 61.



Rysunek 61: Okno logowania użytkownika

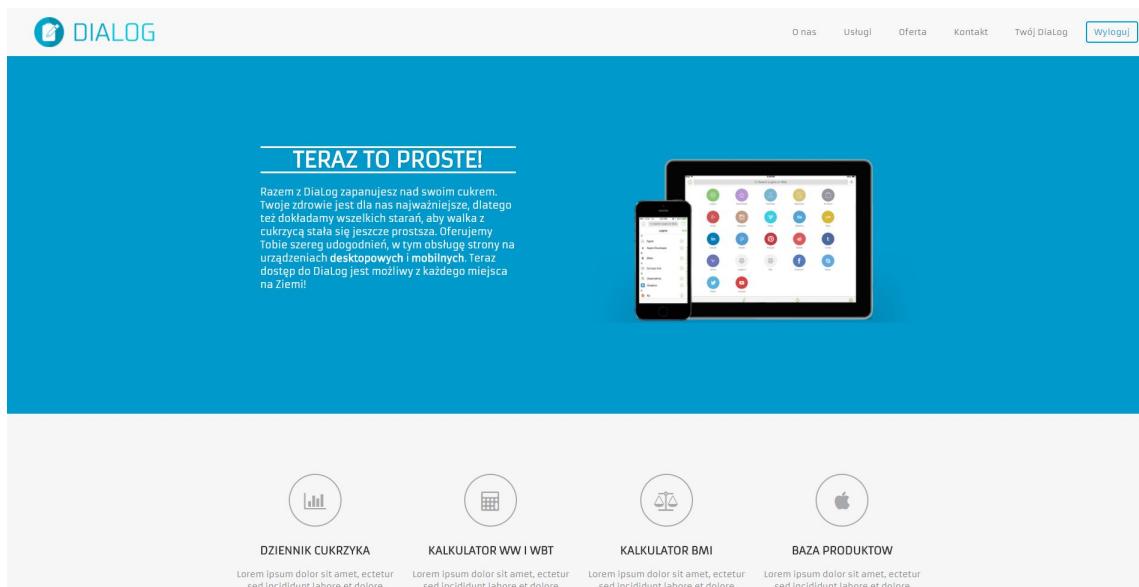
Po wpisaniu danych służących do logowania użytkownik uzyskuje dostęp do nowej pozycji w menu górnym – *Twój DiaLog*. Menu górne dostępne po zalogowaniu użytkownika zostało ukazane na rysunku 62.

6.2 Użytkowanie panelu użytkownika zarejestrowanego

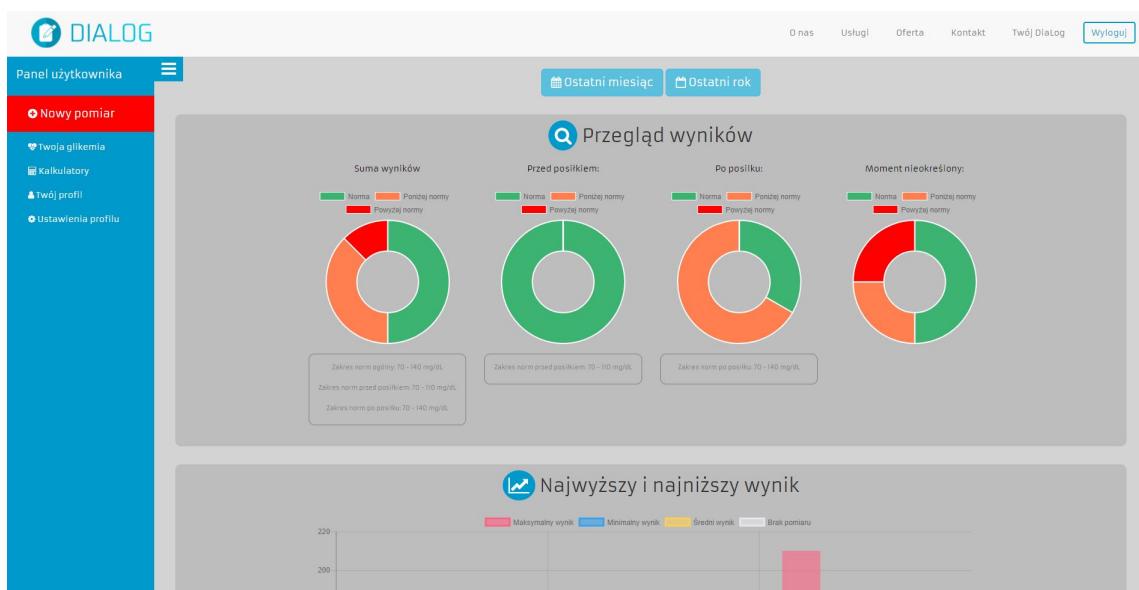
Po uzyskaniu dostępu do zakładki *Twój DiaLog* i kliknięciu jej, użytkownik uzyskuje dostęp do strony głównej panelu użytkownika zarejestrowanego będącej jednocześnie komponentem podsumowującym pomiary wprowadzone przez użytkownika. Początkowo, po rejestracji wykresy i tabela zestawień są puste, gdyż nie zawierają żadnych danych. Strona główna po kliknięciu pozycji górnego menu *Twój DiaLog* została ukazana na rysunku 63.

6.2.1 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 62: Menu górne rozszerzone o dodatkową pozycję – *Twój DiaLog*, dostępne po zalogowaniu użytkownika



Rysunek 63: 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:

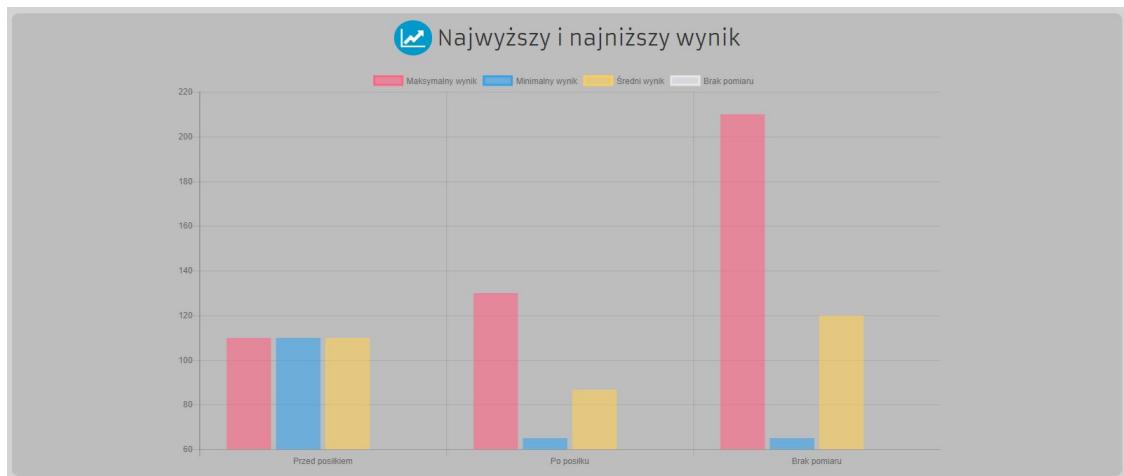
- Norma ogólna: 70 - 140 mg/dL.
- Po posiłku,
 - Zakres norm przed posiłkiem: 70 - 110 mg/dL.
 - Moment nieokreślony.
 - Zakres norm po posiłku: 70 - 140 mg/dL.

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 kursem na daną część wykresu użytkownikowi wyświetla się dymek z podpowiedzią ile pomiarów znajduje się w danym zakresie norm.



Rysunek 64: 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 65: 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 66: Tabela zestawień pomiarów glikemii dostępna z poziomu zakładki *Twoja glikemia*

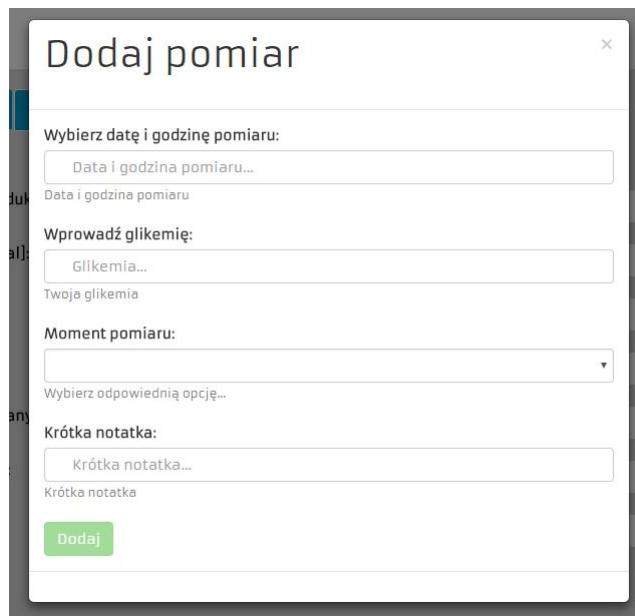
Użytkownik ma możliwość wybrania przedziału czasu, na podstawie którego wyświetlane mają być dane. Dostępne są dwie opcje – *Ostatni miesiąc* oraz *Ostatni rok*. Wyboru dokonuje się poprzez kliknięcie jednego z dwóch dostępnych przycisków na szczytce podstrony.

6.2.2 Dodaj pomiar

Zakładka *Dodaj pomiar* otwiera okno modalne dodawania nowego pomiaru (rys. 67). 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 nowy pomiar.



Rysunek 67: Okno modalne dodawania nowego pomiaru

6.2.3 Kalkulatory

Zakładka *Kalkulatory* pozwala użytkownikowi na uzyskanie dostępu do kalkulatora *BMI (Body Mass Index)* oraz do kalkulatora wymienników węglowodanowych oraz białkowo -tłuszczykowych, 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 kolejnym wzrost (rys. 68). 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. 69):

- 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.

The screenshot shows the 'Kalkulator BMI' (BMI Calculator) interface. It has two input fields: 'Waga [kg]' (Weight [kg]) with placeholder 'Wprowadź swoją wagę...' (Enter your weight...) and 'Wzrost [cm]' (Height [cm]) with placeholder 'Wprowadź swój wzrost [m]...'. Below these are two smaller text boxes: 'Wprowadź swoją wagę [kg]...' and 'Wprowadź swój wzrost [m]...'. A blue 'Oblicz' (Calculate) button is located at the bottom right.

Rysunek 68: Interfejs kalkulatora BMI

The screenshot shows the same BMI calculator interface as in Rysunek 68, but with values entered: 'Waga [kg]' is 85 and 'Wzrost [cm]' is 187. A green horizontal bar at the bottom displays the result: 'Wynik: Twój BMI: 24.31 co oznacza, że masz prawidłową masę ciała' (Result: Your BMI: 24.31 which means you have a healthy body mass).

Rysunek 69: Interfejs kalkulatora BMI po uzyskaniu wyniku

Kolejną funkcjonalnością dostępną z poziomu zakładki *Kalkulatory* menu bocznego jest kalkulator wymienników węglowodanowych i wymienników białkowo-tłuszczywych na podstawie tabeli makroskładników produktów dostępnych z poziomu tabeli *products*, która może być stale rozszerzana przez zalogowanych użytkowników aplikacji (rys. 610). Wynik przedstawiany jest w postaci kolumn dołączonych do tabeli makroskładników danego produktu. Aby obliczyć wartość wymienników węglodowanych i wymienników białkowo-tłuszczywych 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. 611).

Kalkulator WW i WBT

Wybierz produkt:	<input type="text" value="Znajdź produkt"/>							
Wyszukaj i wybierz produkt z dostępnej bazy produktów								
Wpisz wagę produktu [g]:	<input type="text" value="100"/>							
Wpisz wagę wybranego produktu [g]								
Tabela wartości:								
Nazwa	Kcal	Białko [g]	Tłuszcze [g]	Węglowodany [g]	Błonnik [g]	Porcja [g]	WW	WBT
< >								

Rysunek 610: Interfejs kalkulatora wymienników węglowodanowych i białkowo-tłuszczywych

Kalkulator WW i WBT

Wybierz produkt:	<input type="text" value="Bułka pszenna"/>								
Bułka pszenna									
Wpisz wagę produktu [g]:	<input type="text" value="100"/>								
Wpisz wagę wybranego produktu [g]									
Tabela wartości:									
Nazwa	Kcal	Białko [g]	Tłuszcze [g]	Węglowodany [g]	Błonnik [g]	Porcja [g]	WW	WBT	Uwagi
Bułka pszenna	277	8.10	1.50	57.7	1.80	100	5.77	0.444	Ten produkt może być szkodliwy dla Twojego zdrowia!
< >									

Rysunek 611: Interfejs kalkulatora wymienników węglowodanowych i białkowo-tłuszczywych 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. 612).

Po wypełnieniu wszystkich pól:

- Nazwa produktu,
- Kalorie [kcal],
- Białko [g],
- Tłuszcze [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-tłuszczywych.

Dodaj produkt

Nazwa produktu:	Wprowadź nazwę produktu...
Kalorie [kcal]:	Wprowadź ilość kalorii...
Białko [g]:	Wprowadź ilość białka...
Tłuszcze [g]:	Wprowadź ilość tłuszczy...
Węglowodany [g]:	Wprowadź ilość węglowodanów...
Błonnik [g]:	Wprowadź ilość błonników...
Porcja [g]:	Wprowadź wagę posiłku...

Zapisz

Rysunek 612: Formularz dodawania nowego produktu

6.2.4 Twój profil

Z poziomu zakładki *Twój profil* użytkownik ma możliwość dodania informacji zarówno o profilu (rys. 613) jak i danych medycznych (rys. 614). Czynność ta odbywa się przy pomocy dwóch formularzy z odpowiednimi polami:

- 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.

Twój profil	Dane medyczne
Imię:	Patryk
Nazwisko:	Dzwoniarski
PESEL:	94061401032
Opiekun:	Tak
Ulica:	Poznańska
Nr domu:	12
Miasto:	Słupca
Kod pocztowy:	62-400
Numer telefonu:	608456777
Aktualny glukometr:	Accu-Chek Active
Poprzedni glukometr:	Accu-Chek Performa
Numer seryjny glukometru:	AHBHC-234AC

Rysunek 613: Formularz dodawania i aktualizacji informacji o profilu

Twój profil **Dane medyczne**

Typ cukrzycy: Typ 2
Wybierz odpowiednią opcję...

Rok zachorowania: 2009
Wybierz odpowiednią opcję...

Rodzaj insuliny: NovoMix 30 Penfill
Wybierz rodzaj insuliny...

Przyjmowany lek: Adeksa,tabl.,100 mg
Wybierz główny lek, który przyjmujesz...

Inne leki i insuliny: Ibuprofen
Wybierz inne leki, które przyjmujesz...

BMI: 28.00
Wpisz swój BMI

Wzrost: 187
Wpisz swój wzrost [cm]...

Waga: 79.00
Wpisz swoją wagę...

HbA1c: 114.00
Wpisz swój poziom HbA1c...

Zapisz

Rysunek 614: Formularz dodawania i aktualizacji danych medycznych użytkownika profilu

6.2.5 Ustawienia profilu

W zakładce *Ustawienia profilu* użytkownik ma możliwość dodania, bądź zaktualizowania bieżących zakresów glikemii (rys. 615). Komponent składa się z trzech pól formularza:

- Hipoglikemia,
- Hiperglikemia,
- Hiperglikemia po posiłku.

Ustawienia profilu - zakresy glikemii

Hipoglikemia*: 101
Twoja hipoglikemia...

Hiperglikemia*: 202
Twoja hiperglikemia...

Hiperglikemia po posiłku*: 303
Twoja hiperglikemia po posiłku...

Zapisz

Rysunek 615: Formularz dodawania i aktualizacji danych dotyczących zakresów glikemii użytkownika

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

Rozdział 7

Panel użytkownika niezarejestrowanego oraz niezalogowanego

7.1 Okno rejestracji użytkownika

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

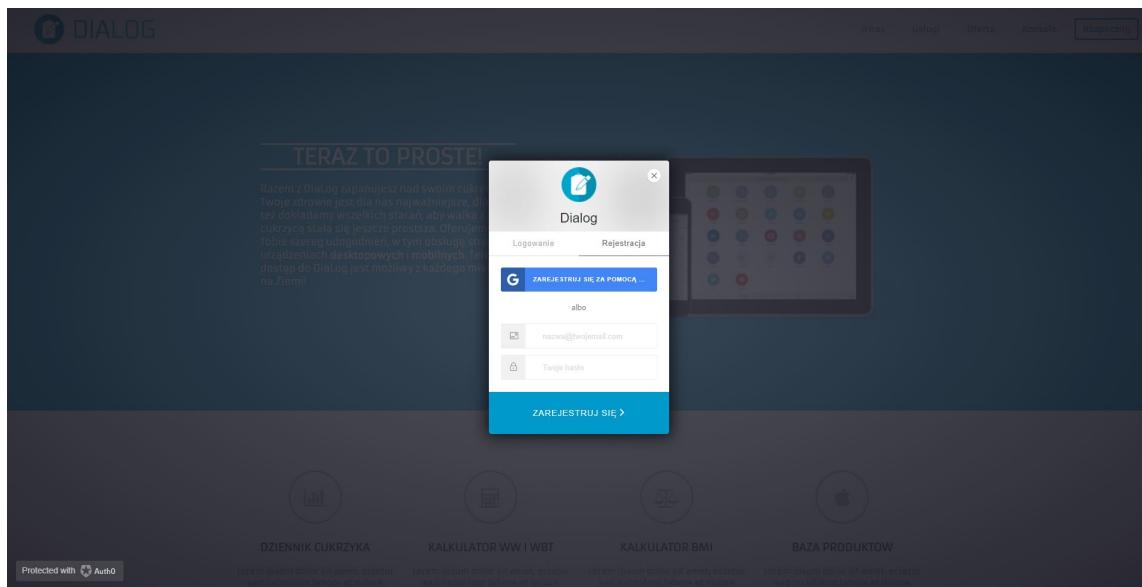
- Rejestracja przy użyciu konta *Google*,
- Rejestracja tradycyjna – podanie adresu e-mail oraz hasła.

7.1.1 Rejestracja przy użyciu konta *Google*

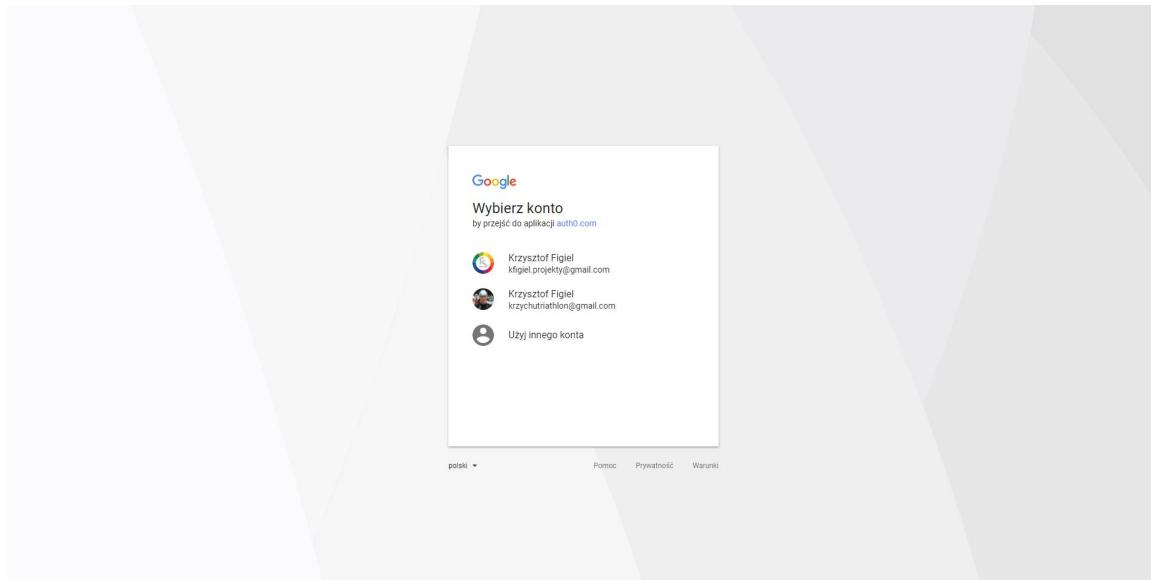
Opcja rejestracji i logowania za pomocą konta *Google* znacznie ułatwia i przyspiesza proces dodawania nowego użytkownika. Pozwala ona zwiększyć odsetek nowych użytkowników systemu. Kolejnym powodem, który przemawia za tym, aby stosować logowanie przy użyciu danych pochodzących z portali społecznościowych 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 społeczeństwa 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 klikając przycisk *Zaloguj się za pomocą konta Google* (rys. 71) i wpisuje dane logowania do konta *Google*, bądź w przypadku kiedy jest zalogowany globalnie, wybiera odpowiednie konto (rys. 72),
- 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.



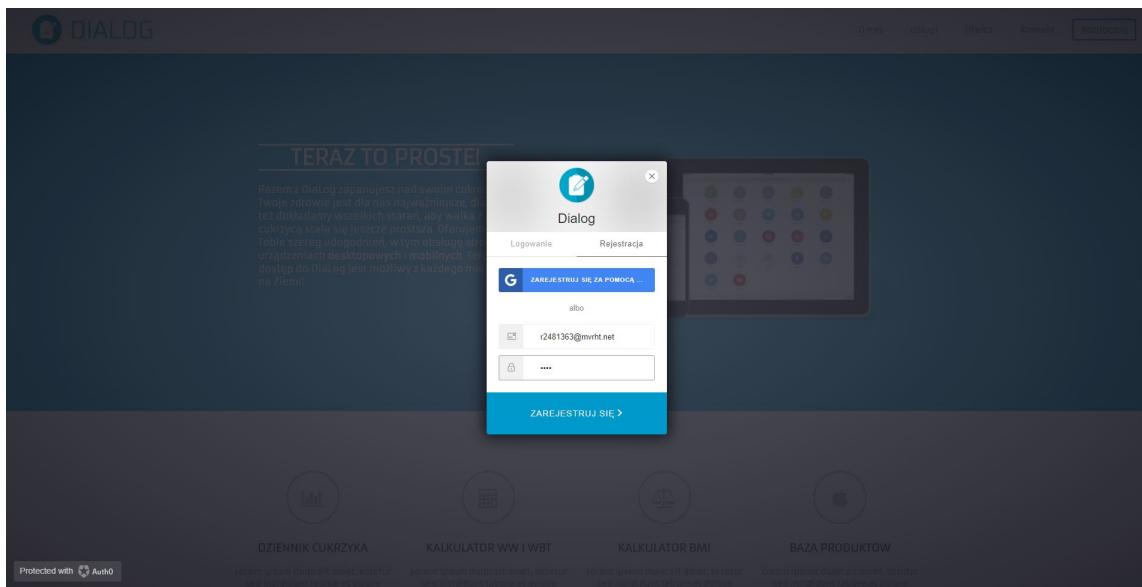
Rysunek 71: Okno rejestracji użytkownika do systemu



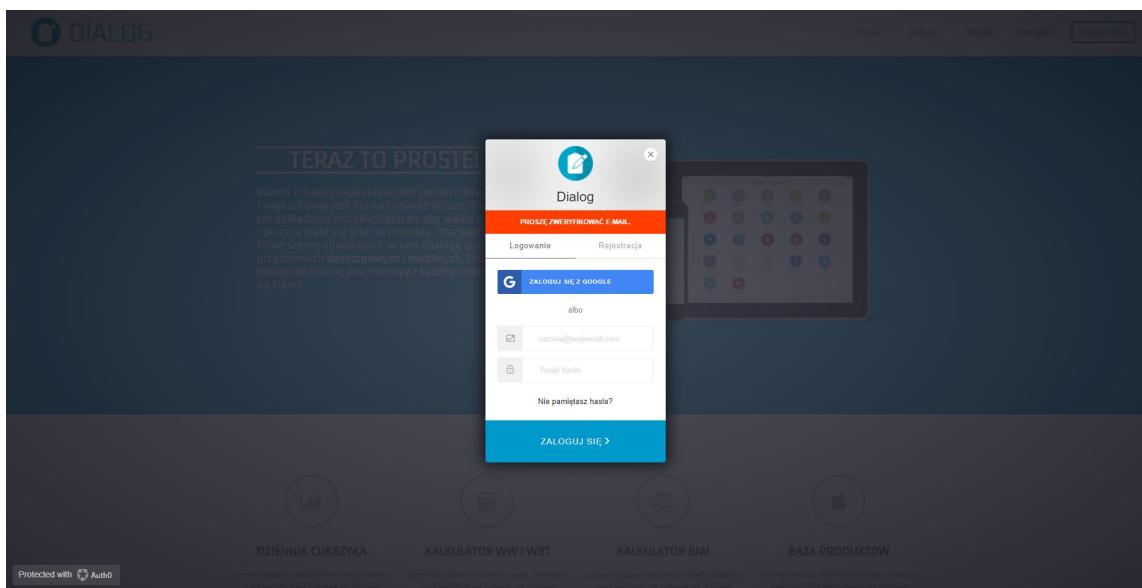
Rysunek 72: Okno wyboru konta Google

7.1.2 Rejestracja tradycyjna – podanie adresu 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. 73) użytkownik zostaje powiadomiony o konieczności potwierdzenia autentyczności konta poprzez kliknięcie w link aktywacyjny wysłany automatycznie w wiadomości e-mail przez serwis *Auth0* (rys. 74). Po kliknięciu w link aktywacyjny możliwe jest już zalogowanie się.



Rysunek 73: Okno rejestracji tradycyjnej



Rysunek 74: 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. 75):

- Imię,
- Nazwisko,
- Temat,
- Wiadomość

The screenshot shows a web-based application interface for 'DIALOG'. At the top, there's a blue header bar with the 'DIALOG' logo and navigation links: 'O nas', 'Usługi', 'Oferta', 'Kontakt', 'Twój DiaLog', and 'Wyloguj'. Below the header is a map of a street area with several buildings labeled, including 'Szuskiego 66', 'VALDI', and 'Blajan Piekarnia Blaszkiewicz'. A red dot marks the location of the building at Szuskiego 66. To the right of the map is a contact form. It includes input fields for 'Imię' (Jan), 'Nazwisko' (Nowak), and a question 'Jak dodawać pomiary?'. Below these is a text area containing 'Witam,' followed by a question 'jak dodaje się pomiary w Państwnej aplikacji?' and a message 'Pozdrawiam, Jan Nowak'. At the bottom right of the form is a green 'Wyślij' button.

Rysunek 75: Formularz do wysyłania maili do administracji

Rozdział 8

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 uwierzytelnić, a nie dowolnych danych binarnych. *BCrypt* wymaga stosowania soli, co wyróżnia ją od innych funkcji skrótu. Sól w algorytmie *BCrypt* jest złożona z następujących 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 weryfikacją 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 *Odnowa 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. 81).



Rysunek 81: 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. 82).



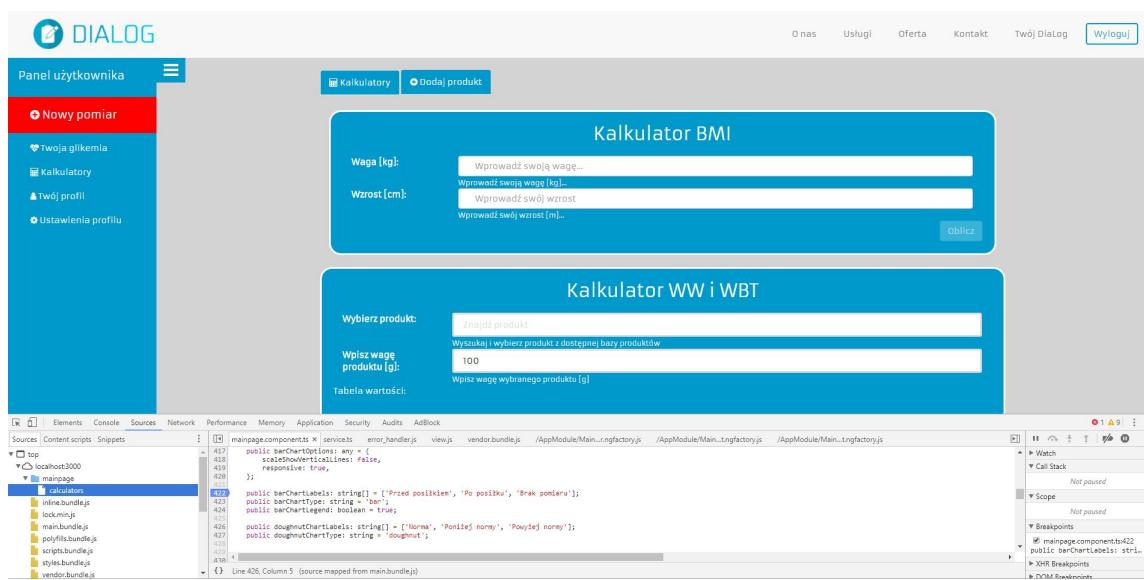
Rysunek 82: Błąd przy próbie uzyskania dostępu do panelu administratora poprzez link URL

Rozdział 9

Testy

9.1 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 91. 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 innych.



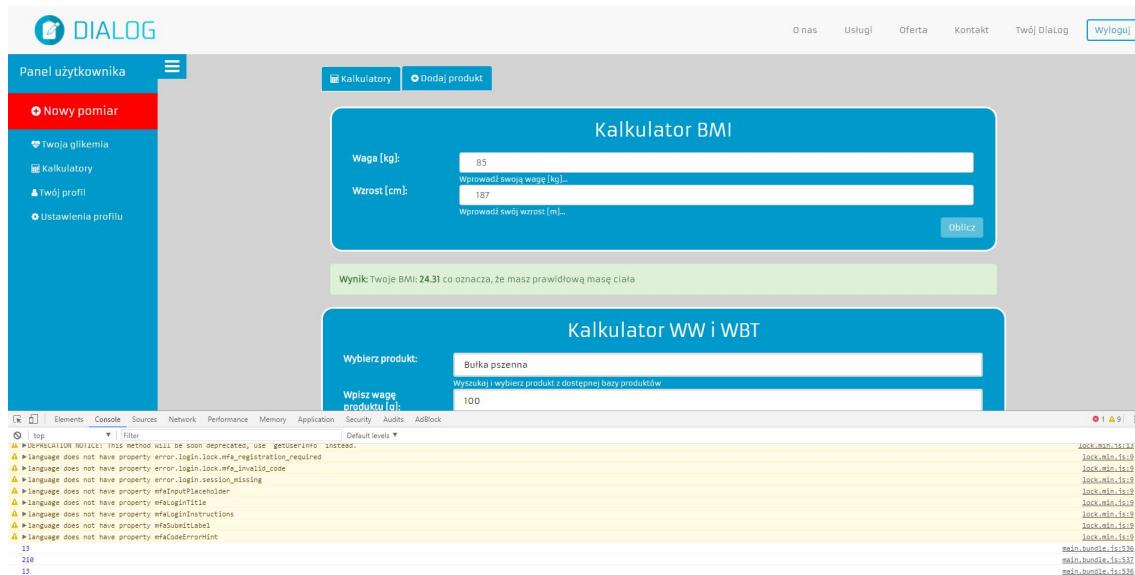
Rysunek 91: Opcje debugowania i podglądu kodu dostępne z poziomu zakładki *Sources* narzędzi deweloperskich przeglądarki *Google Chrome*

Do podglądu poprawności wysyłanych żądań za pomocą metody 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. 92).



Rysunek 92: Opcje podglądu przesyłanych żądań dostępne z poziomu zakładki *Network* narzędzi deweloperskich przeglądarki *Google Chrome*

W celu identyfikacji rodzaju błędu posłużyono się oknem konsoli dostępnym z poziomu zakładki *Console*. Oferuje ona dokładny opis zaistniałego błędu wraz ze ścieżką lokalizacji do pliku z błędym 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 specyfikacją 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. 93).



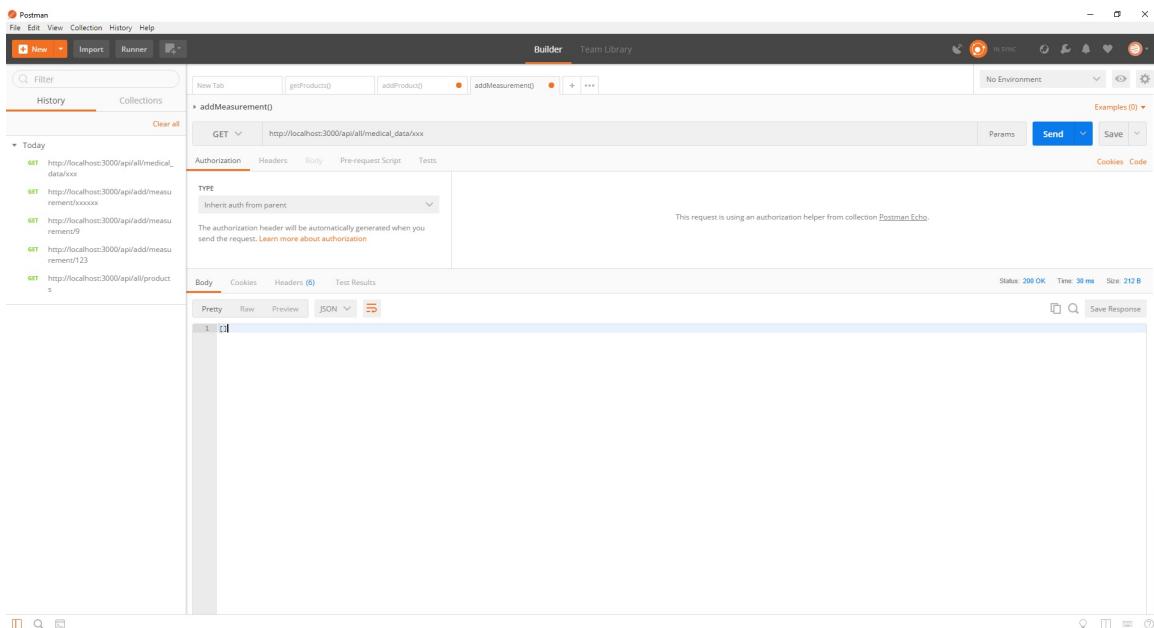
Rysunek 93: Opcje podglądu błędów, ostrzeżeń i wartości zmiennych dostępne z poziomu zakładki *Console* narzędzi deweloperskich przeglądarki *Google Chrome*

9.2 Testy funkcjonalne aplikacji

Aplikację skonstruowano pod kątem UX (*User Experience*), a kod napisano w oparciu o podstawowe zasady budowania struktury projektu tworzonego z wykorzystaniem framework'u *Angular 2* [Ans]. Dołożono wszelkich starań, aby ilość danych umieszczanych w ramach jednego widoku nie była zbyt duża, ponieważ w przypadku *Angulara 2* znaczco 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*). Posłużyło się oprogramowaniem *Postman*.

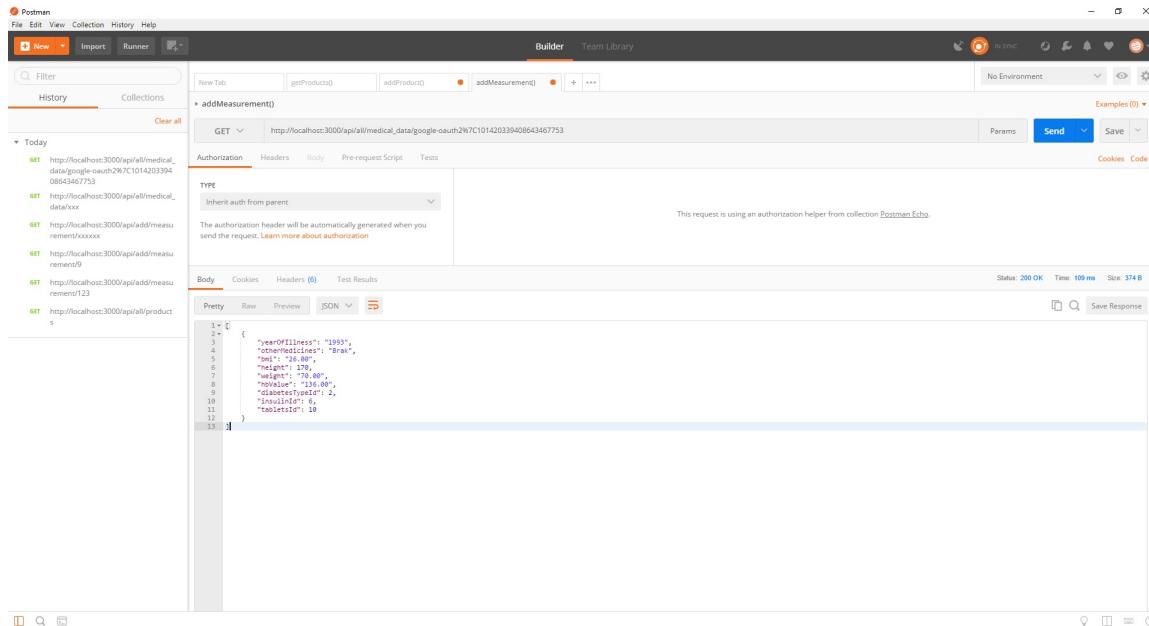
Każda z metod HTTP (*Hypertext Transfer Protocol*) służących do komunikacji z serwerem w celu pobrania lub wysłania danych została sprawdzona poprzez wprowadzenie najpierw błędnych, a następnie poprawnych parametrów w nagłówku, bądź w adresie URL (*Uniform Resource Locator*) metody. Przykładowe wyniki testu przedstawiono na rysunkach 94 oraz 95.



Rysunek 94: Próba pobrania danych medycznych za pomocą oprogramowania *Postman* użytkownika o błędny identyfikatorze

W przypadku wprowadzenia błędego 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 95: 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. 96) [Ang].

The screenshot shows a web-based form titled "Panel użytkownika" (User Panel) for editing a medical profile. The form includes fields for personal information (Imię, Nazwisko, PESEL), address (Oplekun, Ulica, Nr domu, Miejsce), and contact information (Numer telefonu, Aktywny glucometr). A sidebar on the left lists navigation options: Nowy pomiar, Twój glikemia, Kalkulatory, Twój profil, and Ustawienia profilu. The main form has several fields highlighted in red, indicating validation errors:

- Imię: "Jan" (Error message: Wpisz swoje imię...)
- Nazwisko: "Twoje nazwisko..." (Error message: Wpisz swoje nazwisko...)
- PESEL: "9406140103" (Error message: Wpisz swój numer PESEL. PESEL posiada 11 cyfr)
- Oplekun: "Tak" (Error message: Wybierz odpowiednią opcję.)
- Ulica: "Wpisz swoją ulicę..." (Error message: Wprowadzona nazwa ulicy jest za krótka)
- Nr domu: "1" (Error message: Wpisz swój numer domu...)
- Miejsce: "Słupca" (Error message: Wpisz nazwę Twoego miasta)
- Kod pocztowy: "62-4003" (Error message: Wpisz kod pocztowy Twojego miasta. Kod pocztowy wpisany jest w niewłaściwym formacie)
- Numer telefonu: "608456777" (Error message: Wpisz swój numer telefonu.)
- Aktywny glucometr: "Accu-Chek Active" (Error message: Wybierz odpowiednią opcję.)
- Poprzedni glucometr: "Accu-Chek Performa" (Error message: Wybierz odpowiednią opcję.)
- Numer serwisy glucometru: "ABHC-234AC" (Error message: Wpisz numer serwisy Twojego glucometru...)

Rysunek 96: Widok formularza z błędnie wypełnionymi polami i za blokowanym przyciskiem do zapisu ich wartości

Rozdział 10

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-tłuszczych 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 znacząco 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>.
- [Aug15] Ben Augarten. *Express.js Blueprints*. Packt Publishing Ltd., Birmingham, B3 2PB, UK, 2015.
- [Aut] Auth0 documentation. [on-line 01.10.2017]
<https://auth0.com/docs/quickstart/spa/angular2>.
- [BJ16] Jason Marah Benjamin Jakobus. *Mastering Bootstrap 4*. Packt Publishing Ltd., Birmingham, 35 Livery Street, UK, 2016.
- [Car17] Patrick Carey. *New Perspectives on HTML5 and CSS3, 7th Edition*. Cengage Learning., Boston MA 02210, 20 Channel Street B3 2PB, USA, 2017.
- [Dog18] Fernando Doglio. *REST API Development with Node.js, 2nd Edition*. Apress Media LLC, La Paz, Canelones, Uruguay, 2018.
- [Fen18] Steve Fenton. *Pro TypeScript, 2nd Edition*. Apress Media LLC, Basingstoke, UK, 2018.
- [Fre18] Adam Freeman. *Pro Angular 6, 2nd Edition*. Apress Media LLC, London, UK, 2018.
- [Gra18] Keith J. Grant. *CSS in depth*. Manning Publications Co., Shelter Island, 20 Baldwin Road, NY, 2018.
- [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.