

Centrum Kształcenia Zawodowego i Ustawicznego nr 2

Projekt programistyczny
AccessPointMap

Krzysztof Zoń

Racibórz 2020

Spis treści

Rozdział 1. Wprowadzenie.....	2
1.1 Istniejące podobne projekty.....	2
1.2 Cel projektu.....	2
Rozdział 2. Omówienie i specyfikacja techniczna.....	2
2.1 Aplikacja mobilna.....	3
2.1.1 Zadania aplikacji mobilnej.....	3
2.1.2 System Android i technologia Xamarin.....	4
2.1.3 Komunikacja z serwerem.....	5
2.1.4 Graficzny interfejs użytkownika.....	6
2.2 Architektura serwera.....	8
2.2.1 Architektura mikro-serwisów.....	8
2.2.2 Konteneryzacja przy użyciu Dockera.....	9
2.3 Frontend.....	9
2.3.1 Zastosowane technologie.....	9
2.3.2 Generowanie widoków.....	9
2.3.3 Graficzny interfejs użytkownika.....	10
2.4 Backend i baza danych.....	13
2.4.1 Zastosowane technologie.....	13
2.4.2 Schemat działania.....	13
2.4.3 Uwierzytelnianie przy pomocy JSON Web Token.....	16
2.4.4 Struktura bazy danych.....	17
Rozdział 3. Podsumowanie.....	19
3.1 Wnioski.....	19
3.2 Dalszy rozwój projektu.....	19

1. Wprowadzenie

Ludzie na całym świecie korzystają z bezprzewodowych sieci w celu korzystania z lokalnym zasobów oraz globalnej sieci Internet. Jedni decydują się na bezprzewodową łączność ze względu na wygodę, jedni ze względu na to, że infrastruktura sieci pozwala tylko na taką łączność, z kolei użytkownicy telefonów komórkowych mogą tylko w taki sposób połączyć się z danym punktem dostępu.

1.1 Istniejące podobne projekty

Przeszukując zasoby Internetu można trafić na wiele projektów poświęconych prowadzeniu statystyk dotyczących występowaniu access point'ów, czyli punktów dostępu do bezprzewodowych sieci. Projekty takie jak WiGLE, dzięki zaangażowaniu ludzi z całego świata, posiadają wiele informacji na temat access point'ów na całym świecie. Dane są przedstawiane za pomocą odpowiednich kolorów naniesionych na mapę, dzięki czemu wiemy gdzie punktów dostępu jest więcej a gdzie jest ich mniej.

1.2 Cel projektu

W mojej głowie zrodziło się pytanie, czy da się prowadzenie takich statystyk zrealizować na mniejszą skalę (obszar gminy lub powiatu), lecz zbierając o wiele więcej szczegółów, takich jak na przykład konkretne adresy MAC, typy zabezpieczeń, nazwy SSID i częstotliwości. Takie dane mogą być przydatne podczas tworzenia otwartych punktów dostępu dla mieszkańców danej okolicy oraz dla lokalnych dostawców Internetu. Projekt ma też na celu zwrócić uwagę jak wiele ludzi korzysta z szyfrowania WEP, które w dzisiejszych czasach jest bardzo łatwe do złamania i zagraża prywatności oraz bezpieczeństwu naszych poufnych danych, takich jak hasła do kont bankowych lub innych portali, z których korzystamy za pośrednictwem naszej sieci domowej.

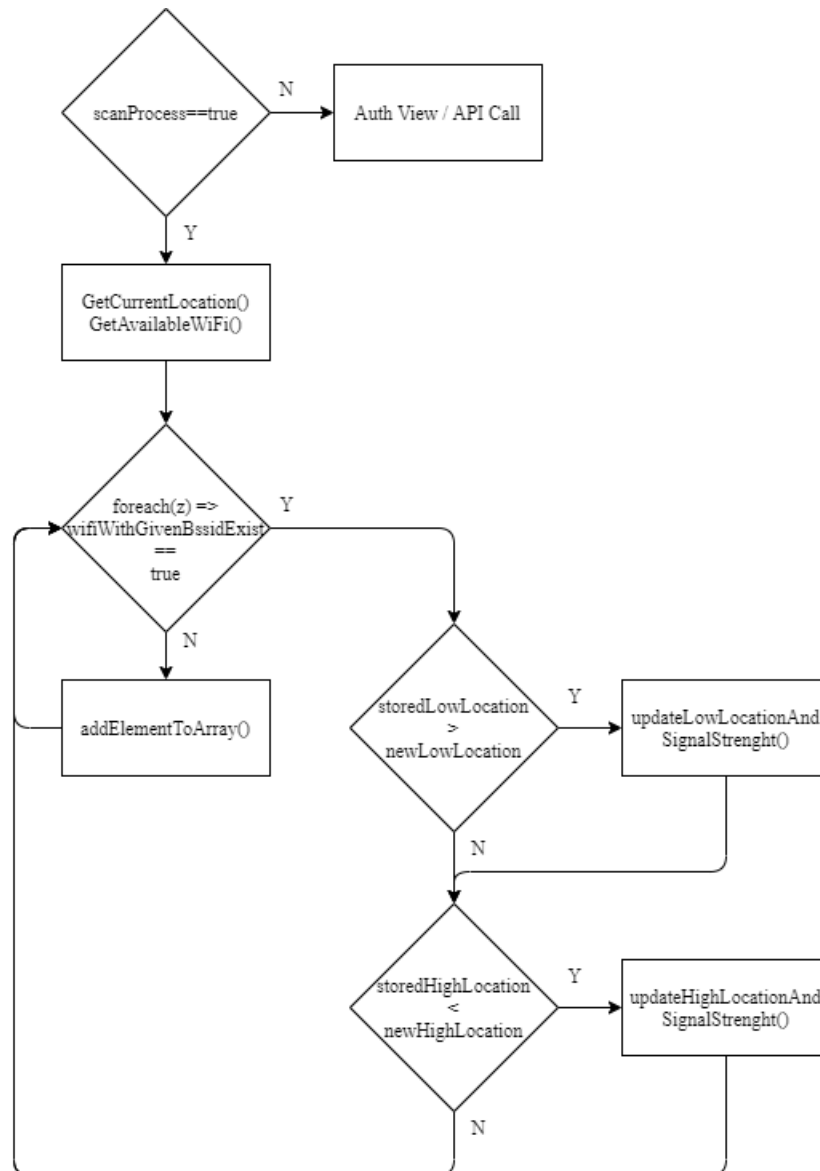
2. Omówienie i specyfikacja techniczna

Z racji tego, że projekt wymagał stworzenia różnego rodzaju oprogramowania, od aplikacji mobilnej zaczynając i na oprogramowaniu serwerowym kończąc, zmuszony byłem do wykorzystania różnych technologii, języków programowania oraz narzędzi programistycznych. Do wykorzystywanych narzędzi należy IDE Microsoft Visual Studio 2019, edytor tekstowy Microsoft Visual Studio Code, system zarządzania bazą danych Microsoft SQL Server Management Studio oraz program służący do badania serwerów http Postman.

2.1 Aplikacja mobilna

2.1.1 Zadania aplikacji mobilnej

Dane trzeba w jakiś sposób pozyskać. Potrzebne jest nam więc urządzenie, które ma dostęp do usług GPS oraz musi posiadać możliwość podłączania się i wyszukiwania sieci Wifi. W dzisiejszych czasach każdy ma takie urządzenie przy sobie, jest nim smartphone lub tablet. Aplikacja cyklicznie wykonuje skanowanie obszaru w celu zebrania informacji na temat punktów dostępu oraz wykonuje zapytania do usługi GPS. W pamięci jest zapisywany nowy obiekt, który posiada parametry poszczególnej sieci oraz aktualnej geolokalizacji. Zadanie to jest wykonywane cyklicznie. Jeśli dana sieć już jest zapisana w pamięci urządzenia a wykryliśmy ją z większą siłą sygnału, to wartość jest nadpisywana a niższa wartość też jest zapisywana w pamięci. Z przechowywania najsilniejszego oraz najsłabszego sygnału płynie taka zaleta, że dzięki temu można obliczyć odcinek od punktu najsilniejszego sygnału do najsłabszego punktu sygnału. Program przyjmuje, że pole które wyznaczymy dzięki temu promieniowi jest obszarem, na którym mamy dostęp do tej sieci. Promień jest obliczany według „Haversine formula”. Skanowanie może być przeprowadzane na różne sposoby, największą dokładność gwarantuje jednak wolne przemieszczanie się podczas procesu skanowania, lecz przemieszczanie się nieco szybsze na przykład samochodem, również gwarantuje zadowalające efekty. Zebrane przez aplikację dane możemy przesłać do serwera za pomocą zapytania POST do API, lecz w przypadku braku dostępu do Internetu możemy plik z danymi lokalnie zapisać na dysku urządzenia i w późniejszym czasie wysłać dane do serwera.



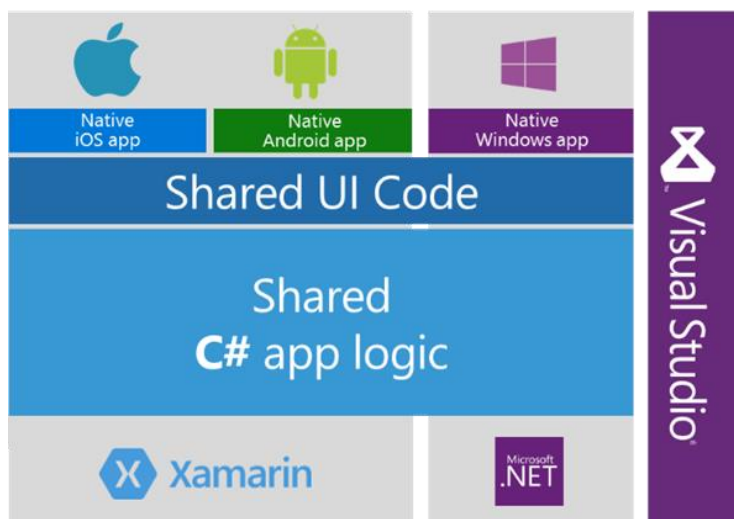
Rysunek 1 Schemat blokowy algorytmu skanowania.

2.1.2 System Android i technologia Xamarin

Aplikacja jest stworzona pod system operacyjny Android. Android to system operacyjny z jądrem Linux dla urządzeń mobilnych takich jak telefony komórkowe, smartfony, tablety (tablety PC) i netbooki. W 2013 roku był najpopularniejszym systemem mobilnym na świecie. Wspomniane jądro oraz niektóre inne komponenty, które zaadaptowano do Androida opublikowane są na licencji GNU GPL. Android nie zawiera natomiast kodu pochodzącego z projektu GNU. Cecha ta odróżnia Androida od wielu innych istniejących obecnie dystrybucji Linuksa. Początkowo był rozwijany przez firmę Android Inc. (kupioną później przez Google), następnie przeszedł pod skrzydła Open Handset Alliance. Aplikacje na system Android można

tworzyć korzystając z takich technologii i języków programowania jak Java lub Kotlin, ja jednak zdecydowałem się na zastosowanie technologii Xamarin, która daje możliwość użycia dobrze mi znanych bibliotek .NET oraz języka C#. Xamarin – amerykańskie przedsiębiorstwo zajmujące się wytwarzaniem oprogramowania, będące jednostką zależną od Microsoftu.

Przedsiębiorstwo zostało założone przez twórców projektów Mono, Mono for Android i MonoTouch, będących wieloplatformowymi rozwiązaniami dla środowisk Common Language Infrastructure oraz .NET Framework. Przedsiębiorstwo udostępnia narzędzia dla środowiska Microsoft Visual Studio umożliwiające tworzenie natywnych aplikacji na urządzenia mobilne (z takimi systemami, jak Android, iOS, czy Windows Phone) w języku C# za pośrednictwem platformy o tej samej nazwie.

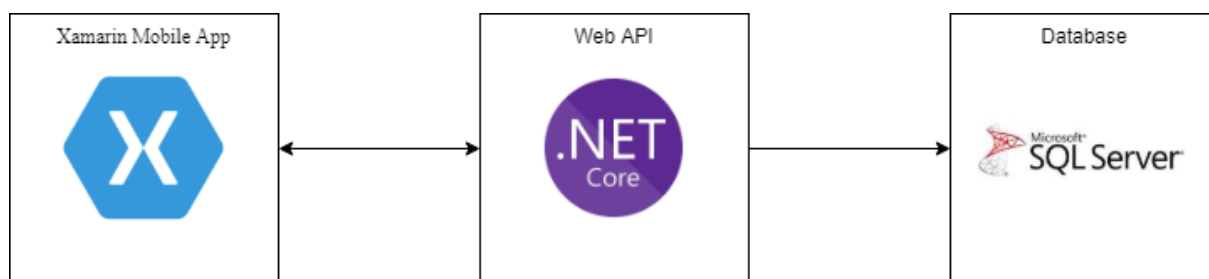


Rysunek 2 Schemat działania aplikacji Xamarin.

2.1.3 Komunikacja z serwerem

Technologia komunikacji z serwerem jest zrealizowana za pomocą Web API. WebAPI (z ang. Web Application Programming Interfaces) to rodzaj sieciowego interfejsu programowania aplikacji, w którym wykorzystuje się architekturę i protokoły sieci Web (w szczególności protokół HTTP) do komunikacji między aplikacjami znajdującymi się na oddzielnych urządzeniach w sieci. API webowe może przyjmować różne formy, takie jak wywoływanie zdalnych procedur (wraz ze zwrotnym przesyłaniem wyników ich pracy), transfer bieżącego stanu zasobów reprezentowanych w sieci i inne. Cechą odróżniającą API webowe od tradycyjnej interakcji w sieci WWW jest to, że użytkownikiem interfejsów nie jest bezpośrednio korzystający w przeglądarki internetowej człowiek, a aplikacja. Zbiór usług Web

API pozwala tworzyć bardziej złożone programy, w których zasób zdalny (obecny na serwerze) jest dostępny z podobną łatwością jak zasób lokalny. Początkową koncepcją była bezpośrednia komunikacja aplikacji z bazą danych, lecz jest to rozwiązanie, które ogranicza funkcjonalność całego projektu i może prowadzić nawet do niebezpiecznych konsekwencji, ponieważ przez inżynierię wsteczną osoba niepowołana mogłaby wejść w posiadanie danych wymaganych do uwierzytelniania i autoryzacji z bazą danych. Aplikacja wysyła dane do API, które następnie są umieszczane w bazie danych po ich odpowiedniej walidacji. Inne aplikacje w celu wysłania lub odebrania danych też łączą się tylko z API a nie z bazą danych, dzięki czemu z bazą danych komunikuje się tylko API i cały ruch danych jest bezpieczniejszy i łatwiejszy do zarządzania.



Rysunek 3 Uproszczony schemat komunikacji z Web API.

2.1.4 Graficzny interfejs użytkownika

Dzięki językowi znacznikowemu XML jesteśmy w stanie zbudować interfejs graficzny dla naszej aplikacji. Proces tworzenia widoków jest dosyć podobny do tworzenia widoków stron internetowych w języku HTML. Składnia budowy widoków w Xamarinie jest taka sama jak w przypadku pisania widoków dla aplikacji mobilnych opartych o języki takie jak Java lub Kotlin, dzięki czemu można łatwo znaleźć wiele dokumentacji i poradników.

APM

Data store method:

☒ Local file

☐ Api call

Geolocation accuracy:

☐ Best (not recommended)

☒ High

☐ Medium

SCAN

STORE DATA

Rysunek 4 Główny ekran aplikacji

APM

Login:

Login

Password

Password

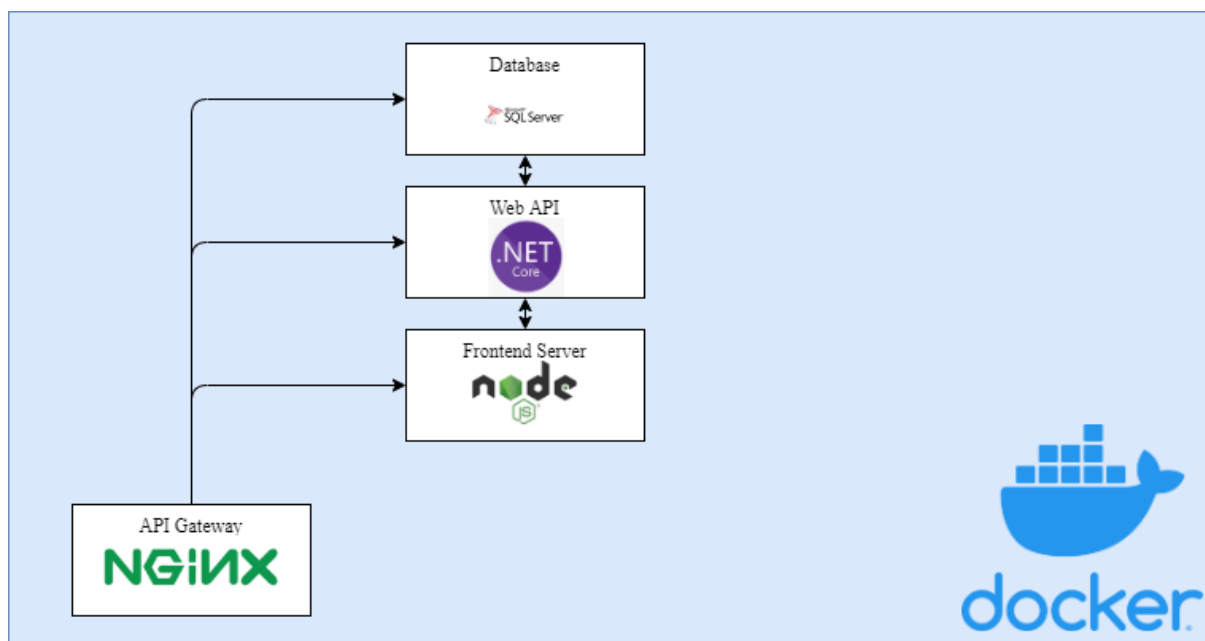
LOG IN AND UPLOAD

Rysunek 5 Panel logowania aplikacji

2.2 Architektura serwera

2.2.1 Architektura mikro-serwisów

Oprogramowanie związane z tym projektem składa się z 3 komponentów: bazy danych, WebAPI oraz serwera, który generuje widoki, czyli to co widzi użytkownik gdy wchodzi na stronę internetową projektu accesspointmap. Serwer, na którym pracują te komponenty jest platformą, na której udostępniam wszystkie moje projekty w formie niezależnych od siebie aplikacji. Taka architektura serwera jest nazywana architekturą mikroservisów, która daje możliwość skalowalności i swobodnych zmian w aplikacji A bez wpływu na działanie i uptime aplikacji B, konteneryzacja poszczególnych serwisów jest realizowana dzięki technologii Docker.



Rysunek 6 Schemat serwisów projektu AccessPointMap

Mikroserwisy to termin określający architekturę aplikacji i sposób ich pisania. W odróżnieniu od monolitycznych rozwiązań, których zasada działania opiera się na rozmieszczeniu poszczególnych części aplikacji w jej wnętrzu (z wykorzystaniem relacyjnego modelu danych), mikroserwisy dzielą je na mniejsze, niezależne od siebie komponenty. Mikroserwisy są zatem oddzielnymi częściami tej samej aplikacji – komponentami lub procesami. Umożliwiają programistom pracę nad elementami aplikacji bez ingerencji w jej całą architekturę. Przyspieszają proces tworzenia oprogramowania, czynią aplikację lżejszą. Zapobiegają awarii całego systemu (jeśli pojawi się problem, to tylko w jednym komponencie).

2.2.2 Konteneryzacja przy użyciu Dockera

Docker to otwarte oprogramowanie służące do realizacji wirtualizacji na poziomie systemu operacyjnego (tzw. „konteneryzacji”), działające jako „platforma dla programistów i administratorów do tworzenia, wdrażania i uruchamiania aplikacji rozproszonych”. Docker jest określany jako narzędzie, które pozwala umieścić program oraz jego zależności (biblioteki, pliki konfiguracyjne, lokalne bazy danych itp.) w lekkim, przenośnym, wirtualnym kontenerze, który można uruchomić na prawie każdym serwerze z systemem Linux. Kontenery wraz z zawartością działają niezależnie od siebie i nie wiedzą o swoim istnieniu. Mogą się jednak ze sobą komunikować w ramach ściśle zdefiniowanych kanałów wymiany informacji. Dzięki uruchamianiu na jednym wspólnym systemie operacyjnym, konteneryzacja jest lżejszym (mniej zasobochłonnym) sposobem wirtualizacji niż pełna wirtualizacja lub parawirtualizacja za pomocą wirtualnych systemów operacyjnych.

2.3 Frontend

2.3.1 Zastosowane technologie

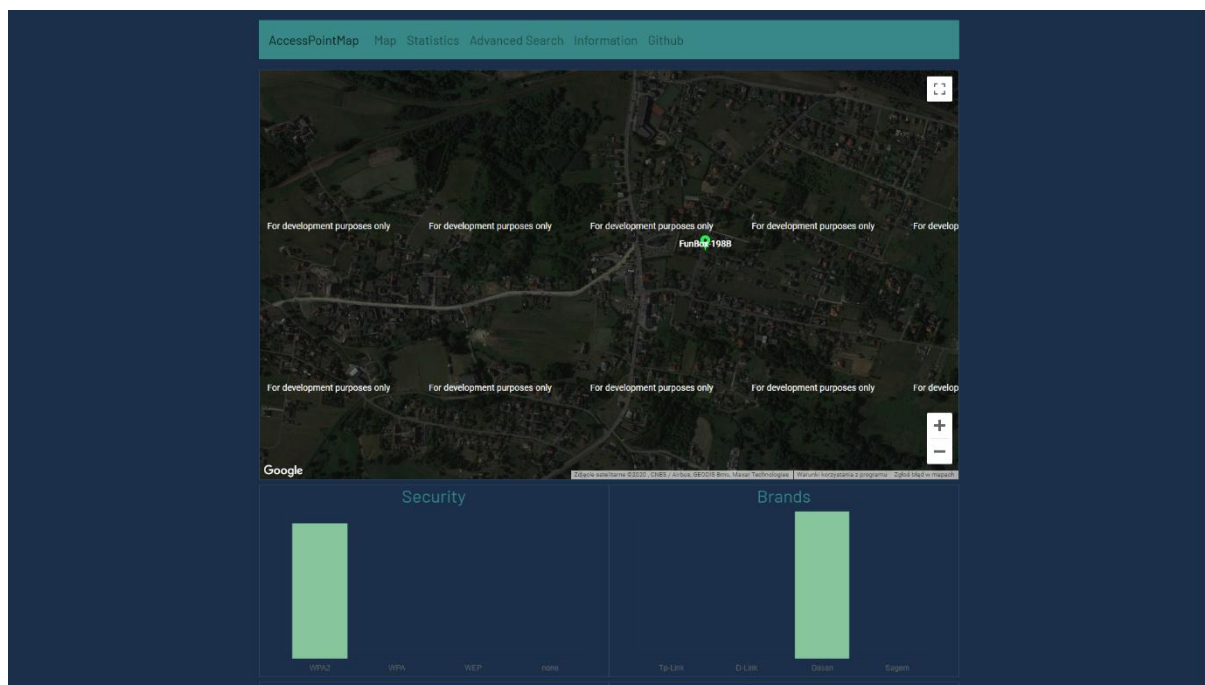
Wszystko, co widzi użytkownik podczas wejścia na stronę projektu AccessPointMap, jest generowane przez serwis zwany „Frontend Serwerem”, przyjmuje on odpowiednia żądania HTTP i na ich podstawie generuje widoki czyli dobrze znane wszystkim strony wykorzystujące HTML, CSS i JavaScript. Frontend serwer jest napisany w języku NodeJS i wykorzystuje ExpressJS jako serwer HTTP.

2.3.2 Generowanie widoków

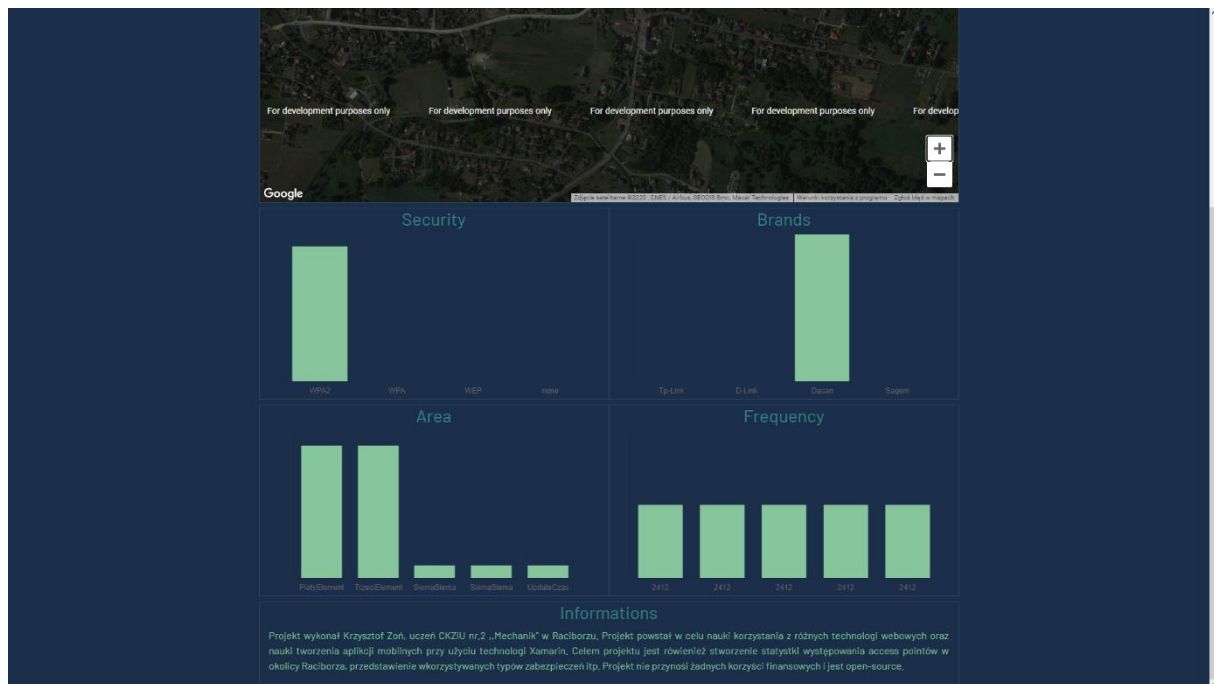
Silnikiem widoków jest EJS, który pozwala na przekazanie obiektów z danymi do widoków w celu przedstawienia użytkownikowi odpowiednich danych. Widoki wykorzystują frontend’owy framework Bootstrap, który pozwala dzięki systemowi siatki na łatwe i szybkie budowanie responsywnych interfejsów. Dzięki responsywności strona jest dostosowana pod duże ekrany telewizorów, średniej wielkości monitory oraz małe ekrany telefonów komórkowych. Gdy użytkownik wchodzi na daną stronę Frontend serwer wykonuje zapytanie do Web API, które zwraca mu odpowiednie dane, które następnie zostają odpowiednio formatowane i wstawione do widoku. Niektóre dane, które rzadziej się zmieniają są przechowywane na lokalnej maszynie gościa w celu odciążenia serwera. Użytkownik zamiast pobierać na nowo te same dane będzie pobiera je z pamięci lokalnej przeglądarki, oczywiście aktualność tych danych jest regularnie sprawdzana i aktualizowana.

2.3.3 Graficzny interfejs użytkownika

Na głównej stronie znajduje się mapa przedstawiająca zebrane w bazie danych access pointy, poniżej znajdują się wykresy dotyczące rodzajów zastosowanych zabezpieczeń, najczęściej występujących częstotliwości sieci, najczęściej występujących producentów sprzętu sieciowego, którzy są źródłem danej sieci oraz punktów dostępu o największej powierzchni zasięgu. Po kliknięciu w znacznik na mapie jesteśmy przenoszona na podstronę, która zawiera informacje o konkretnym punkcie dostępu. Na stronie znajduje się tabela z informacjami oraz mapa, na której okręgiem jest zaznaczony obszar zasięgu danego punktu dostępu. Na głównej stronie na pasku nawigacyjnym znajduje się również zakładka „Advanced search”, prowadzi on do systemu wyszukiwania, gdzie możemy sprecyzować różne parametry w celu wyszukania konkretnych access pointów. W systemie wyszukiwania możemy wyszukiwać punkty dostępu na podstawie ich nazwy, co ciekawe algorytm jest dostosowany do różnego formatowania tekstu, przez co „domyśla się” jakich punktów dostępu na podstawie nazwy szuka użytkownik. Poza nazwą jako parametru szukania access pointów możemy również użyć typu zabezpieczeń, producenta danego sprzętu sieciowego oraz częstotliwości danej sieci.



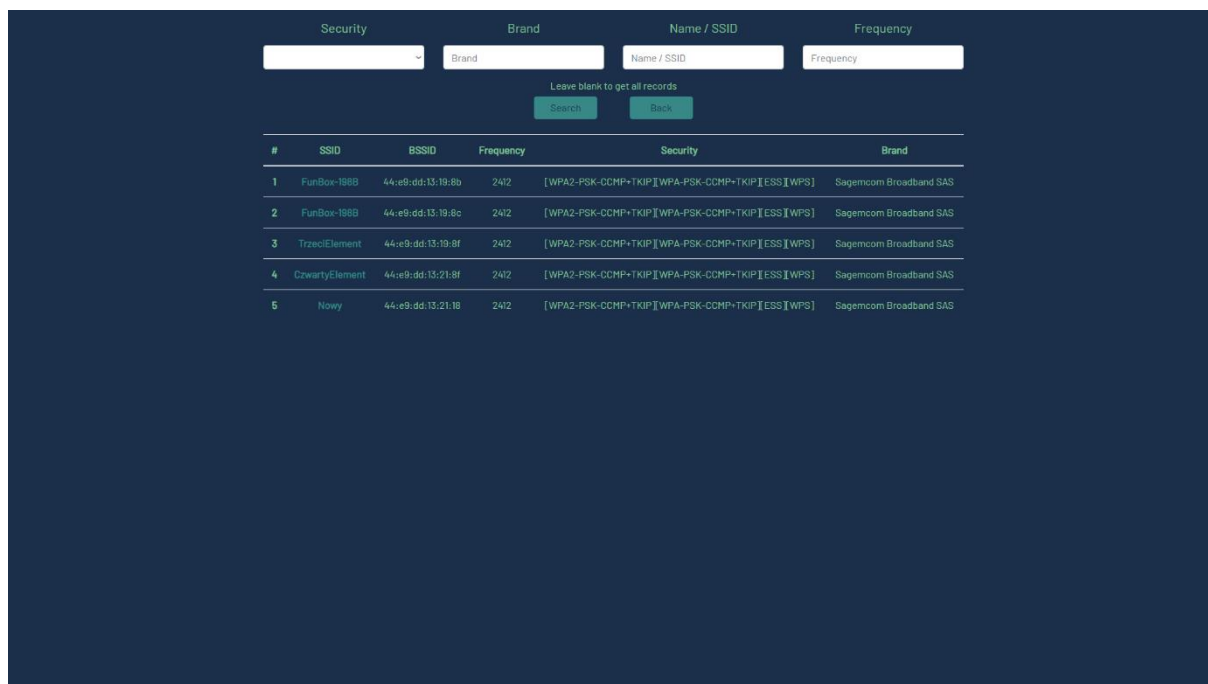
Rysunek 7 Strona główna – mapa



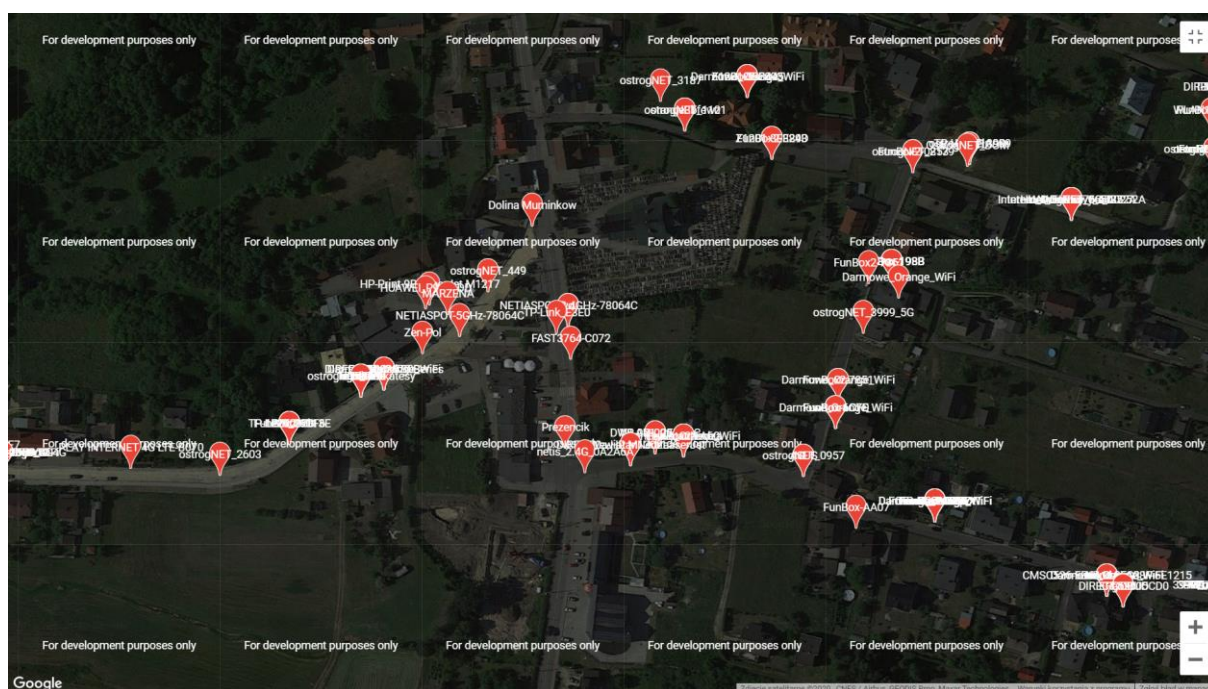
Rysunek 8 Strona główna - wykresy



Rysunek 9 Podstrona /accesspoint/id



Rysunek 10 System wyszukiwania



Rysunek 11 Przykładowe punkty dostępu w miejscowości Nędza w powiecie Raciborskim

2.4 Backend i baza danych

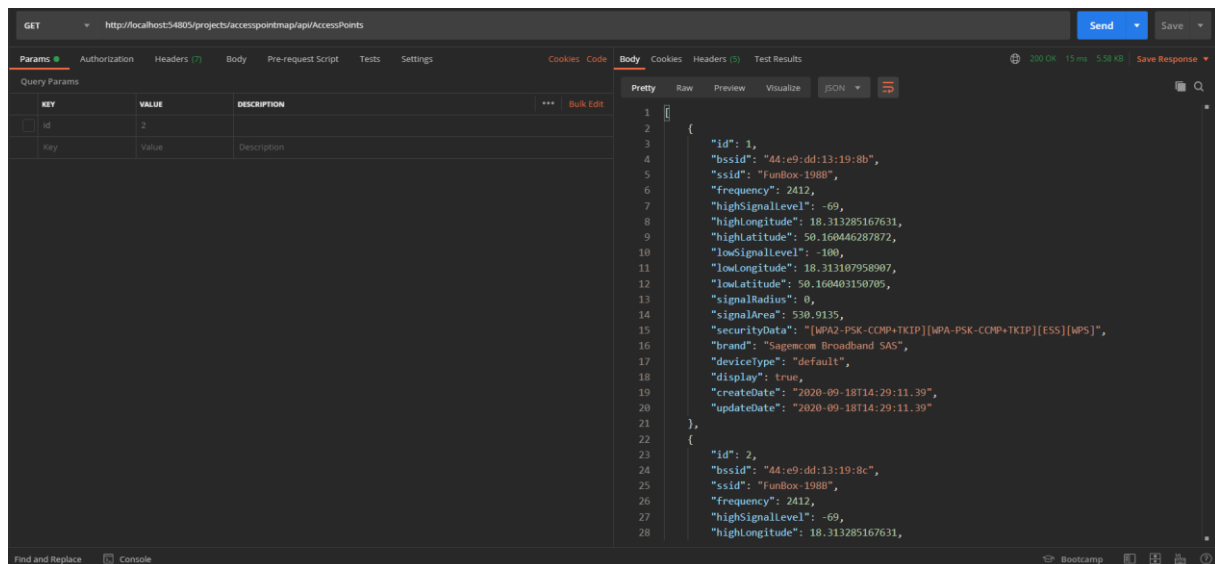
Backend czyli najbardziej istotny element tego projektu, który jest pośrednikiem bazy danych między źródłem danych czyli aplikacją mobilną i frontendem serwer, który jest odpowiedzialny za przedstawienie danych przechodził wiele modernizacji na przestrzeni czasu. Początkowo był bardzo prostym Web API napisanym w języku PHP. Następnie gdy przystąpiono do konteneryzacji został przepisany na język NodeJS z wykorzystaniem ExpressJS. Na ten moment Web API Access Point Map działa na technologii .NET Core.

2.4.1 Zastosowane technologie

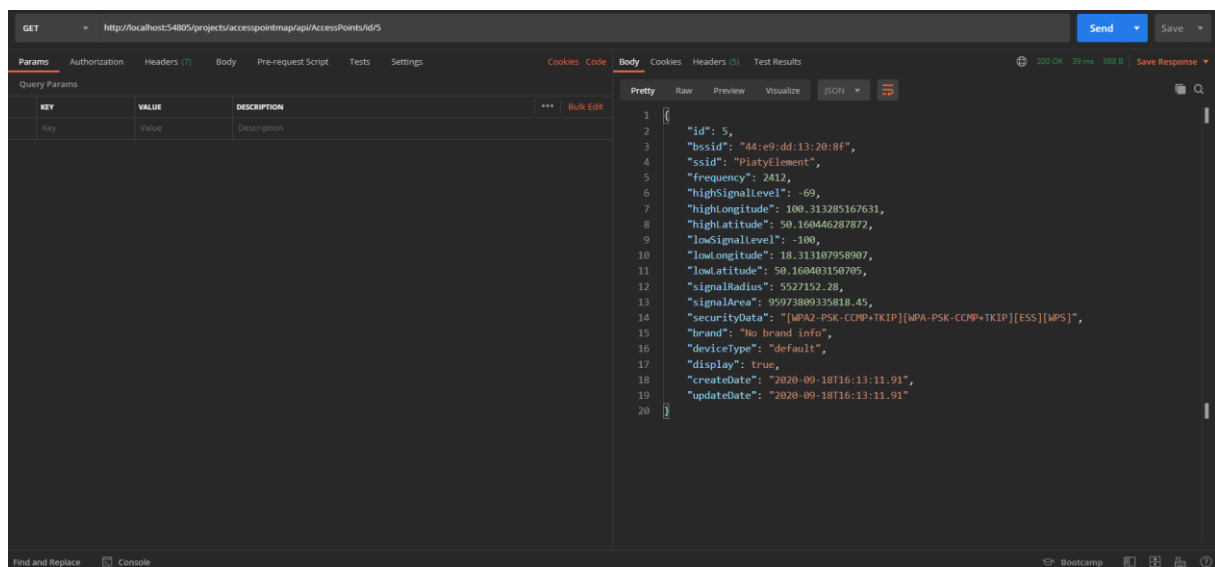
.NET Core to wolne i otwarte oprogramowanie pozwalające tworzyć i uruchamiać wydajne aplikacje na platformach Windows, Linux, macOS. Framework ten umożliwia programowanie aplikacji przeznaczonych dla chmury obliczeniowej oraz IoT, a także backendu aplikacji internetowych z użyciem wzorca MVC. Programy na .NET Core mogą być pisane przy pomocy języków C#, F# oraz Visual Basic. Wykorzystywana jest również technologia Entity Framework, która automatycznie migruje schemat bazy danych na klasy i obiekty, dzięki którym możemy operować na danych na poziomie abstrakcji samego języka, a dzięki technologii LINQ możemy na kolekcjach danych takich jak `List<T>` wykonywać kwerendy podobne do zapytań stosowanych w systemach baz danych takich jak np. SQL Server lub MySQL. Language INtegrated Query (LINQ) to część technologii Microsoft .NET, której podstawy teoretyczne zostały opracowane przez Erika Meijera. Technologia LINQ umożliwia zadawanie pytań na obiektach. Składnia języka LINQ jest prosta i przypomina SQL. Entity Framework (EF) to platforma do mapowania obiektowo-relacyjnego (ORM) typu open source dla ADO.NET.

2.4.2 Schemat działania

Po odwołaniu się do konkretnego endpoint'a w kontrolerze zostają wykonywane operacje na kolekcji obiektów wewnątrz kontekstu bazy danych, następnie kontekst bazy danych może być zsynchronizowany z bazą danych znajdującą się w osobnym kontenerze. Na poniższych grafikach przedstawione są odwołania do wszystkich endpoint'ów Web API.

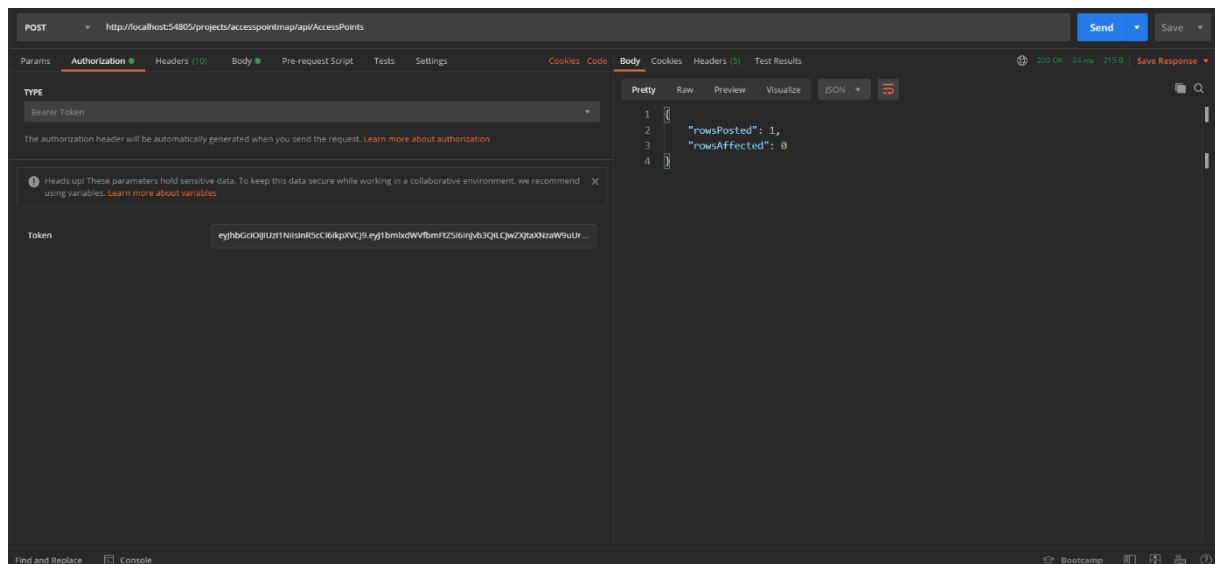


Rysunek 12 Pobranie wszystkich punktów dostępu.

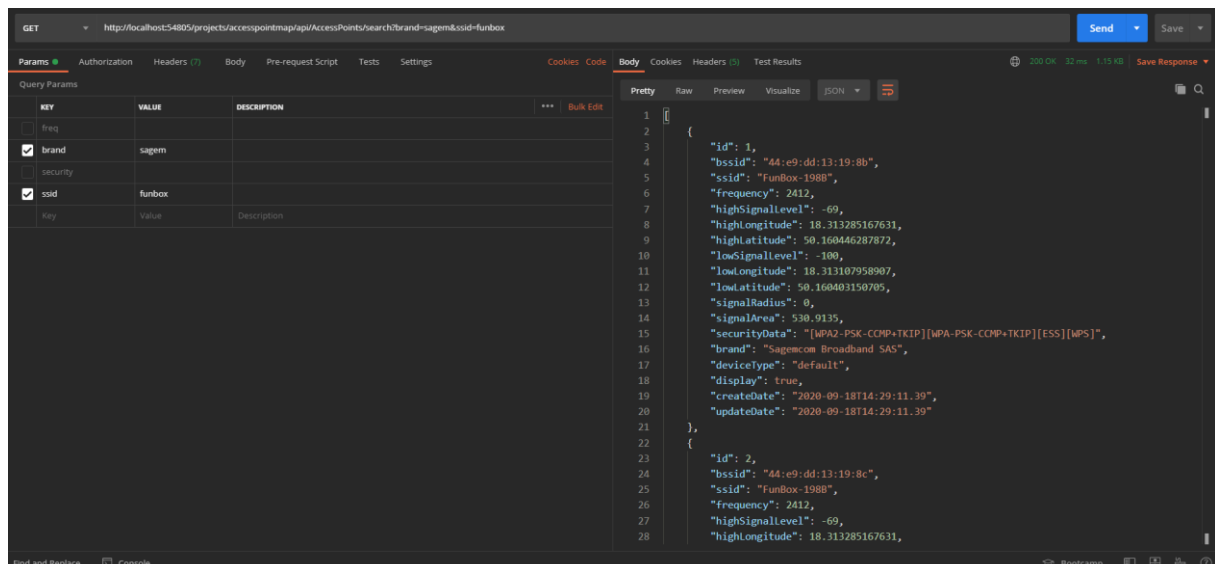


Rysunek 13 Pobranie jednego punktu dostępu na podstawie jego id.





Rysunek 16 Udana próba dodania/aktualizacji punktu dostępu (token został użyty).



Rysunek 17 Szukanie elementów na podstawie podanych parametrów.

2.4.3 Uwierzytelnianie przy pomocy JSON Web Token

Dostęp do informacji o access pointach ma każda osoba wchodząca na stronę projektu, więc nie ma na ten moment takiej potrzeby aby zabezpieczać endpoint służący do wydobywania danych. Istotną kwestią jest to, że nie chcemy aby niepowołane osoby dodawały dane do bazy danych, w tym celu zabezpieczyć trzeba endpoint POST służący do dodawania danych do bazy. Gdy przykładowo będziemy chcieli dodać coś do bazy wykorzystując ten endpoint, serwer zwróci nam informacje, że nie mamy dostępu do wykonania tej czynności. W pierwszej kolejności musimy skorzystać z endpoint'a `/auth` wysyłając tam login i hasło do naszego konta,

w celu uwierzytelnienia. Jeśli login i hasło jest poprawne jako odpowiedź otrzymamy JsonWebToken, który jest ciągiem znaków z zakodowanymi informacjami i sumą kontrolną.

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE


```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

Rysunek 18 Przykład JWT.


Gdy chcemy dodać lub zaktualizować dane przez Web API trzeba w nagłówku żądania umieścić otrzymany ciąg znaków jako „Bearer token”. Jeśli nasz klucz zostanie zweryfikowany i proces autoryzacji zakończy się pomyślnie nasze dane zostaną przetworzone w API i potencjalnie dodane do bazy lub zaktualizują istniejące rekordy. Sam klucz traci ważność po określonym czasie, aktualna konfiguracja serwera Access Point Map definiuje ważność JWT na 5min.

2.4.4 Struktura bazy danych

W celu magazynowania danych wykorzystywana jest relacyjna baza danych z systemem zarządzania Microsoft SQL Server. Microsoft SQL Server (MS SQL) to system zarządzania bazą danych, wspierany i rozpowszechniany przez korporację Microsoft. Jest to główny produkt bazodanowy tej firmy, który charakteryzuje się tym, iż jako język zapytań używany jest przede wszystkim Transact-SQL, który stanowi rozwinięcie standardu ANSI/ISO. Baza danych składa się z dwóch tabel. Pierwsza tabela o nazwie Access Points służy do przechowywania poszczególnych punktów dostępu, druga tabela natomiast przechowuje użytkowników, którzy mogą wprowadzać swoje dane poprzez Web API, a jej nazwa to Users.

	Column Name	Data Type	Allow Nulls
	id	int	<input type="checkbox"/>
	bssid	varchar(25)	<input type="checkbox"/>
	ssid	varchar(50)	<input type="checkbox"/>
	frequency	int	<input type="checkbox"/>
	highSignalLevel	int	<input type="checkbox"/>
	highLongitude	float	<input type="checkbox"/>
	highLatitude	float	<input type="checkbox"/>
	lowSignalLevel	int	<input type="checkbox"/>
	lowLongitude	float	<input type="checkbox"/>
	lowLatitude	float	<input type="checkbox"/>
	signalRadius	float	<input type="checkbox"/>
	signalArea	float	<input type="checkbox"/>
	securityData	varchar(65)	<input type="checkbox"/>
	brand	varchar(60)	<input checked="" type="checkbox"/>
	deviceType	varchar(50)	<input type="checkbox"/>
	display	bit	<input checked="" type="checkbox"/>
	createDate	datetime2(6)	<input checked="" type="checkbox"/>
	updateDate	datetime2(6)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Rysunek 19 Struktura tabeli „Accesspoints”.

	Column Name	Data Type	Allow Nulls
	id	int	<input type="checkbox"/>
	login	varchar(50)	<input type="checkbox"/>
	password	varchar(60)	<input type="checkbox"/>
	tokenExpiration	int	<input type="checkbox"/>
	writePermission	bit	<input type="checkbox"/>
	readPermission	bit	<input type="checkbox"/>
	createDate	datetime2(6)	<input checked="" type="checkbox"/>
	lastLoginDate	datetime2(6)	<input checked="" type="checkbox"/>
	lastLoginIp	varchar(45)	<input type="checkbox"/>
	active	bit	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Rysunek 20 Struktura tabeli „Users”.

3. Podsumowanie

3.1 Wnioski

Projekt przez to, że łączył ze sobą pracowanie z różnymi technologiami, działającymi zarówno po stronie klienta jak i po stronie serwera nauczyła mnie wielu nowych technik i zagadnień związanych z programowaniem. Zagadnienia, o których w tym miejscu warto wspomnieć to z pewnością zarządzanie Runtime Permissions w Androidzie wersji Nougat+. By aplikacja miała możliwość działania trzeba jej zezwolić na dostęp do wewnętrznego API lokalizacji, oraz wifi np.: ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION, ACCESS_NETWORKING itp. Poznałem wiele praktyk utrzymania baz danych oraz utrzymania stron internetowych, reguły REST, korzystanie z ORM (Entity Framework Core), stosowanie Dependency Injection.

3.2 Dalszy rozwój projektu

Plany rozwoju projektu przewidują przeniesienie frontendu na framework opracowany przez Google, Angular. Sama aplikacja służąca do zbierania danych zostanie udostępniona dla wszystkich chętnych do współpracy. Każdy będzie mógł zbierać dane o punktach dostępu i udostępniać te dane na stronie projektu, ponadto będzie możliwość stworzenia własnej prywatnej instancji strony w celu dostosowania przedstawionych danych i innych informacji pod swoje potrzeby. Aktualnie prowadzone też są testy algorytmu sprawdzającego jaki jest typ danego urządzenie (access point, telefon z opcją hot-spot, drukarka sieciowa, telewizor z opcją wifi-direct).

4. Źródła

Ovais Mehboob, Ahmed Khan (2018). *C#7 i .NET Core 2.0 Programowanie wielowątkowych i współbieżnych aplikacji*. Helion

Daniel Jacobson, Dan Woods, Greg Brail (2018). *Interfejs API: Strategia programisty*. Helion

<https://docs.microsoft.com/pl-pl/dotnet/> (18.10.2020)

<https://devopedia.org/xamarin> (18.10.2020)

<https://oktawave.com/pl/blog/co-to-sa-mikroserwisy-i-do-czego-sluza#> (18.10.2020)

<https://jwt.io> (18.10.2020)

Android-x86 – Porting Android to x86 (ang.). (2011-06-28)

Mono Mailpost:earlystory (ang.). mono-project.com. (2018-09-26)

Scott Guthrie: Announcing SQL Server on Linux (ang.). blogs.microsoft.com. (2016-09-11).