

**KOLEGIUM INFORMATYKI STOSOWANEJ**

**Kierunek: INFORMATYKA**

**Specjalność: Programowanie**

Yurii-Volodymyr Shchehliuk

Nr albumu studenta: 58913

***Aplikacja na Xamarin do obsługi klientów restauracji***

Promotor: Dr Marek Jaszuk

**PRACA DYPLOMOWA INŻYNIERSKA**

**Rzeszów 2022**

Ja niżej podpisany/a oświadczam, że składana przeze mnie praca dyplomowa pt. „...........................” została przygotowana samodzielnie.

Oświadczam również, że praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna ze złożoną wersją elektroniczną.

.........................................................

data czytelny podpis autora

*Uwaga - w przypadku pracy dyplomowej zespołowej (o której mowa w punkcie II. Załącznika nr 2 do Zarządzenia) treść oświadczenia brzmi:*

Ja niżej podpisany/a oświadczam, że składana przeze mnie praca dyplomowa pt. „...........................” została przygotowana wspólnie z …………… (wymienić imiona, nazwiska i numery albumów współautorów).

Oświadczam również, że praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna ze złożoną wersją elektroniczną.

.........................................................

data czytelny podpis autora

1. Czytelnie podpisane oświadczenie promotora pracy o treści:

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

................................................................

data czytelny podpis promotora

**Spis treści**

[Wstęp 5](#_Toc94178958)

[1. Wprowadzenie do problemu 6](#_Toc94178959)

[1.1. Mobilne systemy operacyjne 6](#_Toc94178960)

[2. Część teoretyczna 7](#_Toc94178961)

[2.1. Porównywanie narzędzi i technologii mobilnych 7](#_Toc94178962)

[2.2. Platforma Xamarin 7](#_Toc94178963)

[2.3. API i jego rodzaje 11](#_Toc94178964)

[2.4. JWT 15](#_Toc94178965)

[2.5. Postman 15](#_Toc94178966)

[2.6. MSSQL Server 15](#_Toc94178967)

[2.7. C# 16](#_Toc94178968)

[2.8. .NET Core 16](#_Toc94178969)

[2.9. Entity Framework Core 17](#_Toc94178970)

[2.10. SignalR 17](#_Toc94178971)

[2.11. Wzorce projektowe 17](#_Toc94178972)

[2.12. Wzorce architektoniczne 17](#_Toc94178973)

[2.13. Opis wykorzystywanych/implementowanych algorytmów 18](#_Toc94178974)

[2.14. Analiza rynku 18](#_Toc94178975)

[3. Część praktyczna 18](#_Toc94178976)

[3.1. Specyfikacja wymagań 18](#_Toc94178977)

[3.2. Analiza wymagań 19](#_Toc94178978)

[3.3. Diagram przypadków użycia 19](#_Toc94178979)

[3.4. Projektowanie systemu 19](#_Toc94178980)

[3.1. Implementacja 19](#_Toc94178981)

[3.2. Prezentacja produktu końcowego 19](#_Toc94178982)

[3.3. Testy 19](#_Toc94178983)

[3.4. Wnioski 19](#_Toc94178984)

[Podsumowanie 19](#_Toc94178985)

[Literatura 19](#_Toc94178986)

[Spis rysunków 19](#_Toc94178987)

[Streszczenie 19](#_Toc94178988)

## Wstęp

Świat chyba zawsze będzie współistnieć z technologiami internetowymi. W czasach teraźniejszych otrzymać dostęp do Internetu można za pomocą różnych narzędzi, zaczynając od klasycznych komputerów aż do lodówek, ale łatwy i szybki dostęp z największymi funkcjonalnościami zapewnia telefon.

Technologie mobilne rozwijają się nie tak długo, jak komputerowe, ale w różnych sferach życia są zastosowane coraz częściej i od dawna są używanie nie tylko do rozmów, i też do transmisji danych, opłaty rachunków, dokonania zakupów, operacji obliczeniowych itp. Zgodnie ze statystyką ponad połowa wszystkich zakupów online jest dokonana z telefonu.

W pracy zostanie przedstawiony projekt i implementacja aplikacji wieloplatformowej dla systemów Android oraz iOS oparte o Xamarin.Forms oraz .NET Core API – są to narzędzia, za pomocą których została stworzona aplikacja przeznaczone na kilka różnych platform jednocześnie (Android, iOS, UWP, Windows Phone), co umożliwia biblioteka Xamarin.Forms i Backend, który będzie wykorzystany do tej aplikacji też można wykorzystać do aplikacji webowej, dzięki czemu rozwój i wspieranie będzie zajmowało dużo mniej kosztów i czasu.

Cel tworzenia takiego programu z wykorzystaniem różnych instrumentów będzie służył do przeniesienia niektórych funkcyjności z rzeczywistości do aplikacji mobilnej, by zautomatyzować procesy komunikacji z miejscem publicznym, na przykład restauracją oraz nauczenia się szczegółom implementacji serwisów wieloplatformowych. Zostaną wykorzystane i omówione wzorce architektoniczne oraz projektowe, dzięki którym aplikacja będzie łatwo wspierana, oraz rozbudowana.

W części teoretycznej zostaną omówione narzędzia oraz ich zastosowanie, wady i zalety w porównaniu do innych instrumentów. Zostanie przeanalizowany rynek i popularność aplikacji takiego typu. W części praktycznej została opisana interakcja użytkownika z aplikacją…

## Wprowadzenie do problemu

Aplikacje różnego typu rozwijają się każdego roku, w tym aplikacje służące do komunikacji online, rezerwacji lub wynajęcia jakichś rzeczy, lub obliczeniowe dla różnych potrzeb.

Praca z aplikacją takiego typu powinna być z intuicyjnym interfejsem dla użytkownika, by łatwo się przyzwyczaić, szybko działająca na różnych wersjach i systemach operacyjnych oraz łatwa we wsparciu i rozwoju ze strony deweloperów.

### **Mobilne systemy operacyjne**

Najpopularniejszym systemem operacyjnym na świecie, który pojawił się jako projekt open-source, licencji Apache 2.0 jest Android. W tej chwili ilość zarejestrowanych użytkowników korzystających z tego systemu jest większa w porównaniu z systemem Windows o 7% a od iOS popularniejszy 2 razy,czyli Android stanowi 39.75%, 32.44% Windows oraz 16.7% iOS i inne. Ten system był tworzony przez Android Inc., który następnie został kupiony przez Google w 2005, we wrześniu 2008, była przedstawiona pierwsza wersja systemu. W 2009 roku została wydana aktualizacja pod nazwą Cupcake i wszystkie kolejne nazwy systemu, aż do wersji Android 10, który był wydany w 2019 roku. Wersje zaczynają się kolejną literą łacińskiego alfabetu nazwą jakiegoś deseru, czyli: 1.6 Donut, 2.0/2.1 Eclair i tak do wersji 9.0 Pie. W czasie ewolucji pokonał konkurentów jak Windows Phone, BlackBerry oraz Symbian. Wśród mobilnych systemów operacyjnych obecnie jedynym konkurentem tego systemu Linuxowego jest produkt firmy Apple, czyli iOS.

Co do historii iPhone OS można powiedzieć, że w swoich czasach to była naprawdę rewolucja w świecie technologii. Na pierwszy rzut oka te smartfony podbijają niezrównanym designem zarówno samego telefonu, jak i systemu operacyjnego. Oprócz wyglądu i interfejsu jest możliwość dokonania transakcji bezdotykowych, FaceID, Siri i inne nowoczesne technologie. Owszem, że takie same możliwości są na telefonach z systemem operacyjnym Android, takich jak Samsung, Xiaomi, Google Pixel i inne, które oparte o Androida, ale Apple się udało dopracować te technologie do perfekcji, mimo to, że, jak i w każdym innym telefonie są jak wady tak i zalety.

Obecność systemu operacyjnego jest główną cechą wyróżniającą smartfon od zwykłego telefonu komórkowego. Przy wyborze konkretnego systemu operacyjnego jest często czynnikiem decydującym o modelu telefonu lub innego urządzenia. Najpopularniejsze systemy operacyjne dla smartfonów to:

1. Android - system operacyjny dla smartfonów, tabletów, e-booków, odtwarzaczy cyfrowych, zegarków i netbooków oparty na jądrze Linux. Pierwotnie opracowany przez Android Inc. i późnij kupiony przez Google. Następnie Google utworzyło organizację Open Handset Alliance (OHA), która obecnie wspiera i rozwija platformę. Android pozwala na tworzenie aplikacji opartych na Javie, które sterują urządzeniem za pomocą bibliotek opracowanych przez Google. Android Native Development Kit pozwala systemowi na korzystanie z bibliotek napisanych w C i innych językach;
2. iOS (iPhone OS do 24 czerwca 2010) to mobilny system operacyjny, opracowany i produkowany przez amerykańską firmę Apple. Został wydany w 2007 roku; początkowo dla iPhone'a i iPoda Touch a później dla urządzeń takich jak iPad i Apple TV. W przeciwieństwie do Windows Phone i Google Android, jest on dostępny tylko dla urządzeń firmy Apple;

## Część teoretyczna

Zostaną opisane wybrane technologie w porównaniu do innych możliwych rozwiązań oraz podejście architektoniczne i projektowe realizowanego programu. Będzie opisana analiza rynku pod kątem celów wykorzystania aplikacji dla danej branży.

### Porównywanie narzędzi i technologii mobilnych

1. Kotlin – natywny język programowania tylko dla systemu Android, dlatego jest wydajniejszy w porównaniu do aplikacji cross-platformowych. Natywne aplikacje pozwalają na uzyskanie bezpośredniego dostępu do podzespołów urządzenia, na przykład aparatu, GPS czy mikrofonu w bardziej wydajny sposób. Jest to istotne ze względu na to, że wydajność i szybkość działania ma istotny wpływ na ogólną jakość aplikacji.
2. Swift – zasadniczo służy do tworzenia aplikacji na iOS i inne produkty Apple, ale od wersji 5.3 pozwala na napisanie aplikacji na Windows, a od wersji 2.2 na Linux. Aktualną wersją tego języka jest wersja 5.5, września 2020 roku.

Używa kodowania UTF-8, aby zoptymalizować wydajność dla szerokiej gamy przypadków użycia. Pamięć jest zarządzana automatycznie przy użyciu ścisłego, deterministycznego zliczania referencji, utrzymując zużycie pamięci na minimalnym poziomie bez kosztów ogólnych zbierania śmieci.

1. React native - Łączy elementy natywnego programowanie z React i biblioteką JavaScript do budowania interfejsów użytkownika. Może dlatego po 7 latach od premiery, jesteśmy jeszcze na wersji 0.66, oznacza to, że framework nie jest stabilny i wciąż nie wiadomo, kiedy to nastąpi. Nawet mimo to, że jest jeszcze rozwijany, deweloperzy na całym świecie (Facebook w pierwszy w kolejce) używaj ją, że w dużej skali aplikacji.
2. Xamarin – ta technologia pozwala na tworzenie aplikacji cross-platformowych. Zaletą tego podejścia jest to, że jeśli budżet pracy i czasu realizacji ograniczony, wybór platformy takiego typu jest idealnym rozwiązaniem. Można zaoszczędzić około 30%-40% finansów, na przykład tylko na napisaniu kodu, ponieważ pierwsze wersje projektu będą wydane na Androida i iOS w dość krótkim okresie. W 2016 roku Microsoft kupiła Xamarin, czyli projekt Mono za $400 mln i zrobiła go darmowym i open-source.
3. Flatter - według badań przeprowadzonych w 2021 roku przez niemiecką firmę Statista, Flutter okazał się najpopularniejszym hybrydowym frameworkiem mobilnym preferowanym przez deweloperów na całym świecie, z oszałamiającym udziałem w rynku wynoszącym 42%. Pozwala tworzyć na różne platformy przy użyciu tej samej bazy kodu i zachować natywną wydajność. Mnóstwo wbudowanych widżetów i hot reload.
4. MAUI - open-source'owy, cross-platformowy framework do tworzenia natywnych aplikacji mobilnych i desktopowych za pomocą XAML i C# docelowo na .NET 6. Pierwszy release odbył się w listopadzie 2021, twierdząc, że jest to ewolucja ich wcześniej znanego hybrydowego frameworka aplikacji mobilnych o nazwie Xamarin. .NET MAUI dodał do tego zestawu wsparcie dla rozwoju aplikacji desktopowych. Zestaw narzędzi .NET MAUI zastąpił tradycyjne zestawy narzędzi Xamarin.

### Platforma Xamarin

#### Składnia Xamarin

Kluczowe elementy, które pozwalają na tworzenie aplikacji dla kilku systemów:

* Język C# - umożliwia znajomą składnię oraz zaawansowane funkcje, takie jak uniwersalne szablony, LINQ oraz bibliotekę zadań asynchronicznych.
* Mono Framework - zapewnia multiplatformową implementację zaawansowanych funkcji Microsoft .NET Framework.
* Kompilator - W zależności od platformy, tworzy aplikację podobną do natywnej, na przykład na iOS lub zintegrowaną aplikację .NET i runtime dla Androida. Kompilator wykonuje również wiele optymalizacji dla wdrożeń mobilnych, takich jak łączenie nieużywanego kodu.
* Narzędzia IDE - Visual Studio na Mac i Windows pozwala na tworzenie, budowanie i wdrażanie projektów Xamarin.

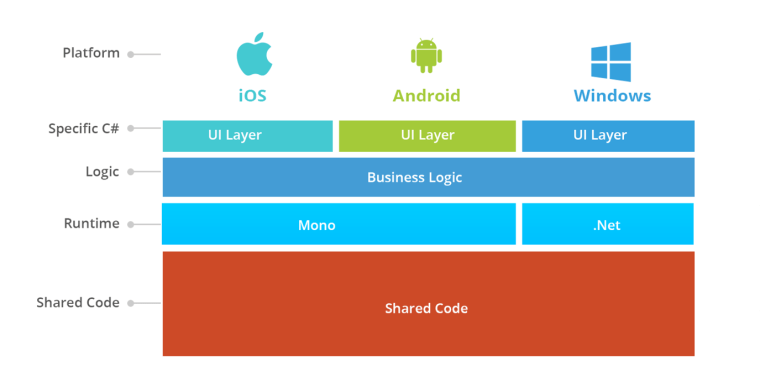
Chociaż Xamarin pozwala na tworzenie aplikacji w C# i współdzielenie tego samego kodu na wielu platformach, jednak konfiguracja różnych ustawień, takich jak tytuł wyświetlanej nazwa aplikacji, ikona, numery wersji i wszelkie kody, które są potrzebne dla iOS lub Android i inne specyficzne rzeczy, które nie są dostępne z cross-platformowego wdrożenia, są różne na każdym systemie.

#### Wady i zalety

Istnieje kilka powodów, dla których Xamarin jest używany przez wiele firm, w tym Pinterest, Slack lub Simens.

1. Jeden stos technologiczny dla rozwoju na wszystkich platformach

Xamarin wykorzystuje język C# oraz .NET Framework do tworzenia aplikacji na dowolną platformę mobilną. Można ponownie wykorzystać około 30% swojego kodu źródłowego, przyspieszając tym samym proces tworzenia oprogramowania. Xamarin nie wymaga również przełączania się pomiędzy środowiskami programistycznymi: wszystkie aplikacje Xamarin mogą być tworzone przy użyciu narzędzi Visual Studio. Narzędzia do programowania wieloplatformowego są dostarczane jako wbudowana część IDE bez dodatkowych kosztów.



Rys. 1 Działanie aplikacji krosplatformowe na bazie .NET  
Źródło [<https://www.nexgendesign.com/xamarin-troubles>]

1. Wydajność jest zbliżona do natywnej

Aplikacje wieloplatformowe tworzone za pomocą Xamarin można zakwalifikować jako natywne, w przeciwieństwie do tradycyjnych hybrydowych rozwiązań webowych. Metryki wydajności są porównywalne z Javą dla Androida i Objective-C oraz Swift dla rozwoju aplikacji iOS. Co więcej, metryki wydajności Xamarin były stale poprawiane i udoskonalane, aby w pełni sprostać standardom natywnego rozwoju. Platforma Xamarin oferuje kompleksowe rozwiązanie do testowania i śledzenia wydajności aplikacji: Xamarin Test Cloud w połączeniu z narzędziem Xamarin Test Recorder pozwala na przeprowadzanie zautomatyzowanych testów UI i wyszukiwanie problemów z wydajnością jeszcze przed wydaniem aplikacji. Usługa ta jest jednak dostępna za dodatkową opłatą.

1. Native UI

Używając elementów UI zależnych od platformy, Xamarin pozwala na stworzenie interfejsu. Zalecane jest używanie Xamarin.iOS i Xamarin.Android osobno dla lepszego rozwoju UI aplikacji. Zapewnia to lepsze wyniki.

1. Kompatybilność sprzętu

Xamarin eliminuje problemy związane z kompatybilnością sprzętową poprzez wykorzystanie wtyczek i różnych API do obsługi wspólnych funkcji urządzeń na wszystkich platformach. Wraz z dostępem do API specyficznych dla danej platformy, Xamarin może być połączony z bibliotekami specyficznymi dla tej platformy. Pozwala to na lepszą konfigurację i wsparcie dla specyficznych dla danej platformy funkcji warstwy natywnej.

1. Łatwe wsparcie

Dzięki swojej międzyplatformowej naturze, Xamarin ułatwia właśnie wsparcie i rozwój oprogramowania. Można po prostu wprowadzić zmiany w jednym pliku źródłowym i będą one stosowane zarówno w aplikacjach na iOS, jak i na Androida. Dotyczy to jednak tylko aplikacji korzystających ze wspólnej logiki biznesowej i także wspólnego kodu dla aplikacji Xamarin.iOS i Xamarin.Android. Pomoże to zaoszczędzić czas i pieniądze podczas utrzymywania aplikacji w ruchu.

1. Pełny zestaw narzędzi programistycznych

Xamarin w jednym pakiecie zawiera pełen zestaw narzędzi deweloperskich: natywne IDE (Visual Studio), Xamarin SDK, testowanie (Xamarin Test Cloud), dystrybucję i analitykę (Hockeyapp i Xamarin.Insights). Nie ma więc potrzeby inwestowania w dodatkowe narzędzia lub integrowania innych aplikacji do tworzenia, testowania i wdrażania aplikacji Xamarin.

Mimo swoich zalet wady tworzenia na Xamarin jednak nie jest proste i oto główne wady:

1. Ograniczony dostęp do bibliotek typu open-source

Natywny rozwój szeroko wykorzystuje technologie open source. Z Xamarinem jest konieczne korzystanie z komponentów dostarczanych przez platformę i niektórych zasobów .NET, z których korzystają programiści i konsumenci. Wybór z pewnością nie jest tak duży, jak w przypadku tworzenia aplikacji mobilnych na iOS i Android osobno, ale komponenty Xamarin zapewniają tysiące różnych elementów interfejsu: diagramy i grafiki, motywy i inne przydatne funkcje, które można dodać do każdej aplikacji za pomocą kilku kliknięć. Platforma ta obejmuje wbudowane funkcje przetwarzania płatności (takie jak Stripe), obsługę sygnałów nawigacyjnych i urządzeń do noszenia, usługi powiadomień Push, rozwiązania do przechowywania w chmurze, możliwości multimedialne, streaming i wiele innych.

1. Opóźnienia w aktualizacjach platformy

To zależy wyłącznie od zespołu developerów Xamarina. Niemożliwe jest, aby narzędzia "firm trzecich" zapewniały natychmiastowe wsparcie dla wydania najnowszych aktualizacji iOS i Android: wprowadzenie nowych zmian, nowych wtyczek itp. zajmuje trochę czasu. Mimo to, że Xamarin twierdzi, że zapewnia wsparcie tego samego dnia, nadal może wystąpić opóźnienie o dzień lub dwa.

1. Ograniczenia ekosystemu

Choć platforma Xamarin wspierana przez Microsoft, jasne jest, że społeczność związane z Xamarin jest znacznie mniejsza niż od iOS czy Androida, więc znalezienie doświadczonego programisty może być trudne. Na podstawie informacji z różnych źródeł społeczność Xamarin stanowi 10% całej społeczności programistów mobilnych. Choć liczba inżynierów Xamarin nie jest porównywalna z liczbą specjalistów iOS czy Android, twórcy platformy zapewniają wsparcie dla swoich specjalistów. Na przykład, istnieje dedykowana instytucja edukacyjna Xamarin University, która zapewnia wiele zasobów i możliwości praktycznych szkoleń dla profesjonalistów z branży. Dzięki takiemu wsparciu, krzywa uczenia się jest minimalna dla doświadczonych programistów C#.

1. Wymagana podstawowa znajomość języka programowania

Kiedy używasz Xamarin.iOS i Xamarin.Android do tworzenia natywnych aplikacji mobilnych, musisz napisać warstwę kodu zależną od platformy. W związku z tym, wymagana jest przynajmniej podstawowa znajomość technologii natywnych (Java/Kotlin dla Androida oraz Objective-C/Swift dla iOS). Nie dotyczą one jednak Xamarin.Forms.

1. Xamarin nie nadaje się do aplikacji z wysokowydajną grafiką

Główną zaletą Xamarin jest możliwość wykorzystania kodu na różnych platformach. Chodzi tu jednak tylko o logikę, kod UI będzie w większości unikalny dla danej platformy. Pozwala na tworzenie gier w Xamarin, ale bogate UI lub złożone animacje z niewielką ilością kodu generycznego sprawiają, że Xamarin nie nadaje się do tego.

1. Większe zastosowania

W zależności od rodzaju i złożoności, aplikacje Xamarin są zazwyczaj większe od aplikacji natywnych, czasami nawet dwukrotnie większe. Na Androidzie, proste "Hello, World" może zająć do 16 MB, z czego większość jest wykorzystywana przez powiązane biblioteki, środowisko Mono i bibliotekę klas bazowych (*ang. Base Class Library*). W związku z tym aplikacja zazwyczaj wymaga dodatkowej optymalizacji, aby utrzymać jej rozmiar pliku na niskim poziomie.

1. Trudności z integracją

Prawdopodobnie wystąpią pewne problemy z integracją zasobów trzecich z Xamarinem. Podczas gdy większość narzędzi i bibliotek oferuje pełne wsparcie natywne, kompatybilność z Xamarin nie może być zagwarantowana przez sprzedawcę z innych firm. Podczas gdy Xamarin posiada swój własny arsenał komponentów, zawsze istnieje możliwość, że będziesz potrzebował specyficznej funkcji lub integracji w swojej aplikacji, która nie jest dostarczana przez platformę.

#### Kompilacja

Kod źródłowy C# trafia do natywnej aplikacji w różny sposób na każdej platformie:

* Android - aplikacja działa równolegle z Java/ART (Android runtime environment) i współdziała z natywnymi typami poprzez JNI. C# jest kompilowany do Intermediate Language i budowany z MonoVM + Just In Time. Uwidacznia interfejsy a biblioteki Android SDK od Google. Nieużywane klasy w platformie są usuwane podczas przetwarzania
* iOS - C# używa wstępnej kompilację (*ang. Ahead of Time Compilation*) do języka asemblera. Framework włącza się i nieużywane klasy są usuwane podczas przetwarzania w celu zmniejszenia rozmiaru aplikacji, czyli tak samo, jak na Androidzie. Mono uwidacznia struktury zestawu SDK CocoaTouch od Apple, do których można się odwoływać w Xamarin, ale niektóre funkcje Apple nie pozwala do wykorzystania w tym reflekcji lub dynamiczne generowanie kodu w czasie rzeczywistym w iOS.

### API i jego rodzaje

API (*ang. Application programming interface*) – opis zachowania się aplikacji i w jaki sposób ona może komunikować się z innym programem i operować danymi lub innymi serwisami. W miarę standaryzacji komunikacji sieciowej informacje były wymieniane cyfrowo przez linie telefoniczne i przewody sieciowe przy użyciu protokołów ogólnego przeznaczenia, takich jak Telnet, SMTP, FTP i HTTP. Na początku był stworzony i używany XML, który jest nadal szeroko stosowany. Jednak teraz w większości przypadków jest wykorzystywany JSON, oparty na tekście. Mniej popularne formaty binarne, na przykład Protocol Buffers i Thrift, które stały się „lingua franca” wymiany danych.

#### SOAP

Simple Object Access Protocol (SOAP) - jest protokołem wymiany informacji zakodowanej w XML (*ang. Extensible Markup Language*) pomiędzy klientem i procedurą lub usługą, która znajduje się w Internecie. Został wydany w świat w 1999 roku i jest publikowana przez W3C jako otwarty standard.

SOAP może być używany przez różne protokoły transportowe oprócz HTTP, na przykład FTP i SMTP. (Konwencjonalnym wzorcem jest użycie HTTP do synchronicznej wymiany danych i SMTP lub FTP do interakcji asynchronicznych).

W celu zapewnienia spójności podczas strukturyzacji danych SOAP wykorzystuje standardowy schemat XML (XSL) do kodowania XML. Dodatkowo programiści mogą tworzyć własne schematy XML, aby dodać niestandardowe elementy XML do wiadomości SOAP.

SOAP jest zwykle używany z językiem opisu usług sieciowych WSDL (*angl. Web Services Description Language*). Znaczenie użycia WSDL polega na tym, że programiści i maszyny mogą sprawdzić usługę sieciową, która obsługuje SOAP, aby odkryć specyfikę wymiany informacji przez sieć. Ponadto WSDL opisuje, w jaki sposób strukturyzować komunikaty żądania i odpowiedzi SOAP, które dana usługa obsługuje. Odkrywanie poprzez WSDL upraszcza programowanie usług sieciowych wykorzystujących SOAP.

Zalety

* SOAP pozwala na generowanie kodu opierając się na XML. XML jest dobrze znany i wszechobecny z dużą elastycznością (na przykład, przestrzenie nazw).
* Ponieważ jest oparty na tekście, debugowanie jest wygodne i nie ma ograniczeń dotyczących transportu. Ponieważ podejścia są opisane w WSDL, plik działa jako wiążący kontrakt dla podziwiania wiadomości i typów.
* Dodatkowa warstwa WS-Security zapewniana przez SOAP działa na poziomie wiadomości, aby upewnić się, że nie tylko treść wiadomości może być odczytana przez właściwy serwer, ale także właściwy proces na serwerze.

Wady:

* Działa tylko z XML, a XML jest bardzo dosłowny i pomiar wiadomości ma tendencję do wzrostu eksponencjalnego.
* Pliki są ciężkie ze względu na duży rozmiar XML tekstu.
* Wsparcie dla technologii kodu w obecnych językach jest ograniczone do niedużej ilości języków, na przykład to Java, Python i C#, podczas gdy SOAP z Go/Rust/PHP/Elixir to rozczarowujące rutyny szkoleniowe.

#### gRPC

gRPC Remote Procedure Call od Google jest rekurencyjnym akronimem dla metody zdefiniowanej wokół wszystkich zalet HTTP/2 podwojonych przez fakt, że używa binarnego formatu do transferu (skompilowane bufory protokołu), który jest niezwykle wydajny.

Celem gRPC jest umożliwienie wywoływania procedur a nie interakcji danych, co czyni go nieco podobnym do SOAP w koncepcji.

Zalety:

* Bardzo wydajny. Wykorzystuje wszystko co HTTP/2 ma do zaoferowania w tym, że można wysyłać rzeczy synchronicznie (czekając na odpowiedź), można strumieniować, można multipleksować, wszystko przez to samo połączenie. Dodajmy do tego fakt, że wiadomości binarne są cholernie małe, i otrzymujemy słodką ofertę na te czasy, kiedy wydajność jest krytyczna.
* Generowanie kodu ułatwia konfigurację i HTTP/2 wymaga TLS, więc jesteś niejako zmuszony do minimalnego bezpieczeństwa.

Wady:

* Trudny do debugowania, ponieważ komunikaty są binarne i nie są czytelne dla człowieka. Chociaż jest obsługiwany we wszystkich językach, niektóre mają ograniczenia (np. nie można mieć serwerów PHP gRPC, tylko klientów), i ze względu na naturę HTTP/2 nie ma wsparcia (obecnie) bezpośrednio w aplikacjach frontendowych.
* Ponieważ musi być TLS wszędzie, konieczne jest to, że trzeba uwzględnić też w wymaganiach konfiguracyjnych obecność TLS protokołu lub nabywając certyfikaty do użytku wewnętrznego od znanego CA (*ang. Certificate Authority*), lub tworzyć własny wewnętrzny CA do wydawania certyfikatów, ale musimy dodać go jako zaufany w całej infrastrukturze/kontenerach/itd.

#### GraphQL

GraphQL, produkt Facebooka, plasuje się gdzieś pomiędzy REST i gRPC. Jego celem jest ułatwienie żądania danych poprzez własny język zapytań, który daje kontrolę klientowi. Komunikuje się za pomocą protokołu HTTP i wykorzystuje format danych JSON.

Zalety

* Jego oparcie na JSON czyni go nieco podobnym do REST, ale z dodatkową korzyścią, że adaptacyjne zapytania oznaczają, że można poprosić o dane, które chcesz, jak chcesz nie ma dużo wielokrotnych zapytań, aby uzyskać wszystko, czego potrzebujesz.
* Przy debugowaniu dostajemy detaliczny opis problemu, czyli w tym dodatkowo referencję do połączenia i całą informację z handlera.
* Posiada walidację schematu i typowanie, więc w tym sensie jest nieco zbliżony do tego, jak gRPC definiuje rzeczy.

Wady

* Cała ta elastyczność odbija się na zdolności do buforowania i jest słaby wydajnościowo. Wynika to z tego jak detaliczną informację dostajemy prze naszych zapytaniach.
* Pomimo nazwy, nie jest to do końca interfejs grafowy. Nie można na przykład pobrać wszystkich przodków jednostki nadrzędnej.
* Nie ma dużo informacji o tworzeniu serwisów na podstawie tego API.

#### REST

REST jest akronimem od Representational State Transfer. REST jest stylem architektonicznym opracowanym przez Roya Fieldinga w jego pracy doktorskiej z 2000 roku. W REST wszystko odbywa się przy użyciu metod HTTP, takich jak:

* GET - odczyt;
* POST – tworzenie i odczyt;
* PUT - modyfikacja;
* DELETE - usuwanie
* PATCH - częściowe modyfikacje zasobu
* OPTIONS – przekazywanie metadanych
* HEAD – odczyt, ale bez odpowiedzi

Używanie tekstowych formatów danych stało się konwencją. JSON jest najbardziej popularnym formatem danych, chociaż można używać innych, takich jak XML, CSV, i nawet RSS. W taki sposób jak przedstawiono niżej wyglądają zapytania odpowiedź z i do serwera, którą otrzymuje klient aplikacji.

curl -X POST "http://localhost:49887/api/Accounts/Login" -H

"accept: application/json" -H

"Content-Type: application/json-patch+json"

-d

{

"Email":"client@email.com",

"Password":"Aacc&556"

}

HTTP/1.1 200 OK

ETag: "b8a7ef8b4b282a70d1b64ea5e79072df"

X-Runtime: 13

Cache-Control: private, max-age=0, must-revalidate

Content-Length: 449

Status: 200

Keep-Alive: timeout=2, max=100

Content-type: application/problem+json; charset=utf-8

{

"access\_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJjbGllbnRAZW1haWwuY29tIiwianRpIjoiZTdhZTg3NDQtOGI5OC00OWNiLTllYzgtNDUwMGQyZGEyNTE1IiwiZW1haWwiOiJjbGllbnRAZW1haWwuY29tIiwiaHR0cDovL3NjaGVtYXMueG1sc29hcC5vcmcvd3MvMjAwNS8wNS9pZGVudGl0eS9jbGFpbXMvbmFtZSI6ImNsaWVudEBlbWFpbC5jb20iLCJodHRwOi8vc2NoZW1hcy5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRlbnRpdHkvY2xhaW1zL3JvbGUiOiJVc2VyIiwibmJmIjoxNjQzMDY0NTcxLCJleHAiOjE2NDMxNTA5NzEsImlzcyI6ImxvY2FsaG9zdC5jb20iLCJhdWQiOiJsb2NhbGhvc3QuY29tIn0.U1UdWYMMna8yaWvnSYXNv1Qyu7PB\_JcHcp7NrBFcy9Y",

"token\_type": "bearer",

"user\_Id": 3,

"user\_name": "client",

"expires\_in": 86400,

"creation\_Time": 1643064571,

"expiration\_Time": 1643150971

}

Zalety:

* Bazowanie na istniejących standardach HTTP wiąże się zarówno z elastycznością jak i ograniczeniami. Większość problemów związanych z transportem jest obsługiwana przez istniejące standardy i ewoluuje wraz z nimi. Czasowniki są powiązane z operacjami (co ułatwia mapowanie operacji CRUD na przykład), encje nadają nazwy i znaczenie URI. Proste konwencje sprawiają, że interakcja z interfejsami API REST jest dość prosta.
* JSON sprawia, że REST jest szczególnie prosty w użyciu dla usług frontendowych, gdzie JS jest wszechobecny, tak że tłumaczenie danych na obiekty jest natychmiastowe.
* Ponadto, JSON jest dość czytelny dla człowieka i łatwy do debugowania.

Wady:

* W praktyce, usługi REST rzadko są tak przyjazne jak zamierzone, ponieważ nie ma wspólnej koncepcji tworzenia. Elastyczność często przekłada się na chaos implementacyjny i na dodatek nie ma wiążącego kontraktu na strukturę używaną w wiadomościach.
* HTTP - aplikacje RESTful są ograniczone do protokołu HTTP
* Gdy ludzie zwracają odpowiedź HTTP 200 w każdym przypadku i zamiast tego kodują status błędu w ciele odpowiedzi to tu leży pis pogrzebany. To jest to, co ludzie SOAP zwykli robić.

Właśnie w tym projekcie będzie wykorzystywany REST API napisany za pomocą .NET Core do komunikacji z aplikacją na Xamarinie za pomocą http zapytań z powodu łatwej implementacji i wydajności.

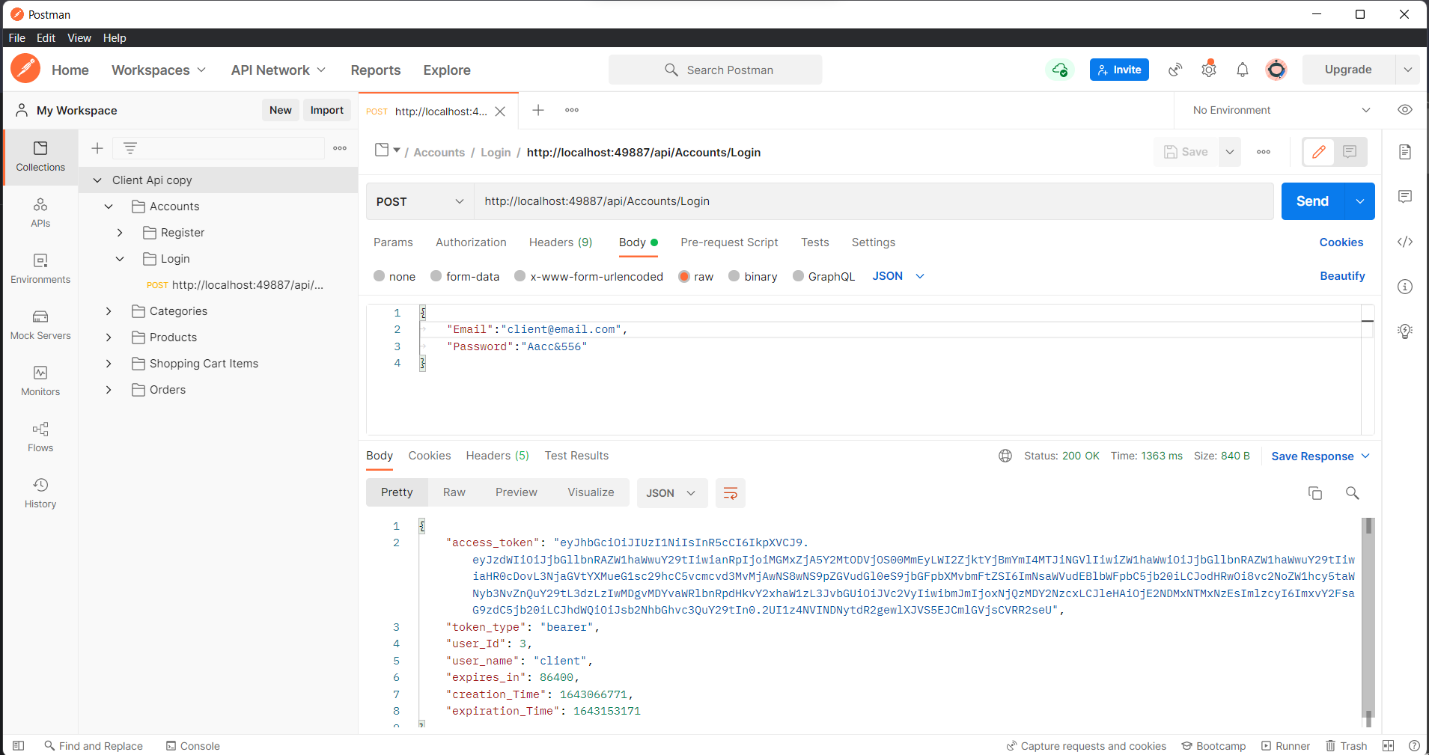
### JWT

Tak jak REST nie mają dobrze zdefiniowanego protokołu bezpieczeństwa, JSON Web Tokens (JWTs) są najbardziej powszechną metodą uwierzytelniania i autoryzacji żądań.

JWT tokeny (*ang JSON Web Token*) jest dobrym sposobem na bezpieczne przesyłanie informacji pomiędzy stronami, ponieważ mogą być podpisane, co oznacza, że możesz mieć pewność, że nadawcy są tymi, za których się podają. Dodatkowo, struktura JWT pozwala zweryfikować, czy treść nie została zmodyfikowana.

### Postman

Jest klient HTTP używany do tworzenia, testowania, udostępniania i dokumentowania interfejsów API wykorzystując graficzny interfejs użytkownika. Służy do testowania backendu, gdzie wpisujemy adres URL punktu końcowego, wysyła żądanie do serwera i odbiera odpowiedź z serwera.

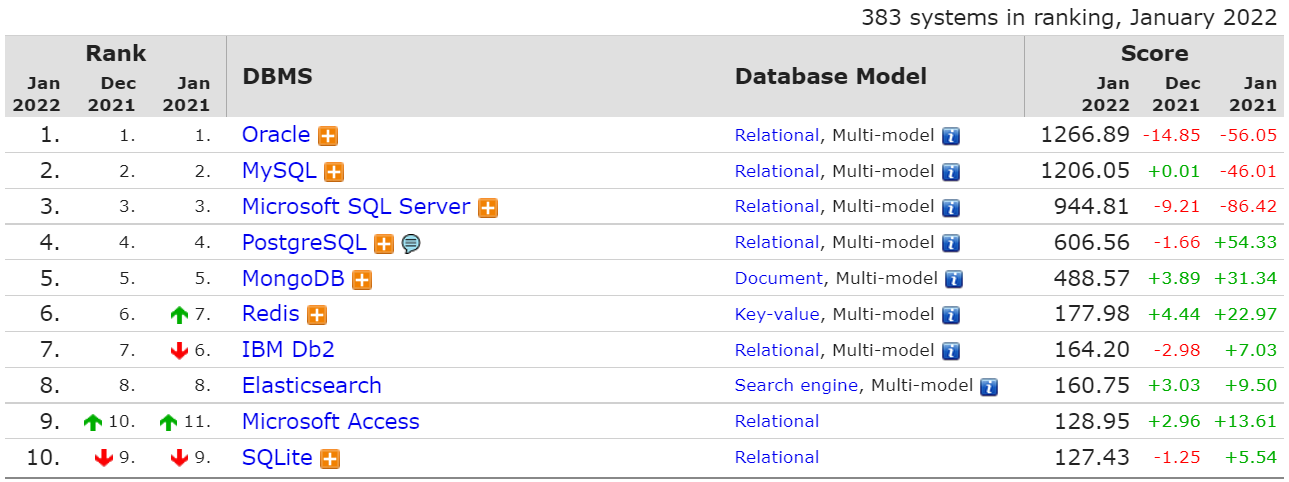


Rys. 2 Własny

W tej pracy jest wykorzystywany do testowania punktów końcowych napisanych w .net core, które się łączą z aplikacją.

### MSSQL Server

W danym przypadku, baza danych będzie lokalna, właśnie, gdzie będzie przechowywana informacja z aplikacji, taka jak: dane z tabel, zdjęcia itp. Wśród najpopularniejszych programów jako serwer może wystąpić Oracle, PostgreSQL, MySQL, MSSQL itp.



Rys. 3 Ranking najpopularniejszych baz danych  
Źródło: W https://db-e ngines.com/en/ranking, z dnia 25.01.2022

Zgodnie ze statystyką sierpnia 2022-ego roku, najpopularniejszą bazą jest Oracle, 1980 roku powstania od firmy „Oracle Corporation”. Porównując bazy danych, na przykład MySQL i MSSQL (Microsoft SQL Server), mamy jak różniące się między sobą cechy tak i wspólne. Właścicielem MySQL, jak i bazy danych Oracle jest firma „Oracle Corporation”. Obydwa programy mogą być zainstalowane na platformie Windows i Linux, ale tylko MySQL może być zainstalowany też na OS X, Solaris i FreeBSD. MySQL wspiera wszystkie języki programowania co i MSSQL i jeszcze kilku dodatkowo. Ale skrypty na MySQL są tworzone wyłącznie na SQL, a MSSQL wspiera też Transact-SQL, .NET języki, R oraz Java.

### C#

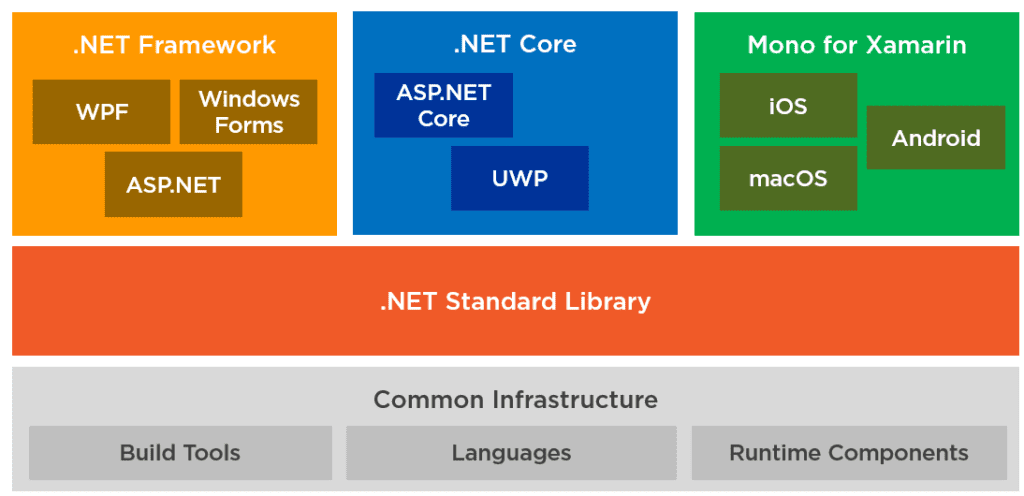
C# jest obiektowym językiem programowania, opracowanym przez firmę Microsoft w 2000 roku w celu zaadoptowania najlepszych cech języków programowania Java i C++. Jest używany z wielu powodów, ale jego popularność polega na wykorzystaniu go do tworzenia serwisów backendowych, desktopowych aplikacji, tworzenia aplikacji webowych, game developmentu i w mniejszym stopniu do machine learningu oraz tworzenia aplikacji mobilnych.

Środowisko uruchomieniowe CLR zawiera kompilator JIT z pośredniego języka IL do kodu platformy, na której zainstalowane jest środowisko CLR; Zawiera Grabage Collector i korzystne feauture cukru syntaktycznego, na przykład getery i setery, anonimowa inicjalizacja obiektów, lockowanie wątków asynchronicznych, i w wesji .NET SDK 6, C# 10 to globalne usingi oraz inne.

Kompilator C# tłumaczy kod źródłowy na kod IL; moduł kompatybilny z CLS może być używany w każdym języku programowania na platformie .NET, a w drugim kroku IL jest transformowany do kodu maszynowego. Jedno co można dodać, że JIT nie przetwarza od razu cały program, a po kolei wywoływane metody.

### .NET Core

.NET Core to runtime open-source platforma używana do tworzenia oprogramowania na macOS, Linux oraz Widnows. Zawiera w sobie Runtime, JIT, Base Class Library, Entity Framework, WPF, VB.NET, F# i wiele innych. Nowsze wersję są lżejsze I szybsze w porównaniu np. do .NET Framework lub .NET Core 2.1 który już nie jest wspierany. Ostatnia wersja to .NET 6 LTS release której odbył się w 2021 i będzie rozwijany do 2024 natomiast w tej aplikacji jest używany .NET Core 3.1 LTS z końcową datą suportu do końca tego roku. Choć nowsza wersja tego frameworku jest mniej rozbudowana, ale wsparcie mikroserwisowej architektury i kontenerów robią ją wydajniejszą i perspektywiczną.



Rys. 4 Ekosystem .NET  
Źródło <https://stackify.com/net-ecosystem-demystified/>

### Entity Framework Core

Nowsza, wydajniejsza i jeszcze raz wspomnę to słowo, krosplatformowa wersja ORM (*ang. Object relative mapping*), która zapewnia dostęp do bazy danych przed modele, czyli klasy i obiekty kontekstu reprezentujący bazę danych. Kontekst służy do zapisywania rekordów i napisania zapytań do bazy danych za pomocą LINQ.

Alternatywą EF jest ADO.NET lub NHibernate. W pierwszym przypadku są wykorzystywane zwykle zapytania SQL, dzięki czemu mamy większą kontrolę. Ale minusem jest ogromna ilość kodu oraz potencjalne ataki w postaci SQL iniekcji. Gdyż NHibernate zapewnia dostęp przez obiekt sesji i używa mapowanie do tabel za pomocą XML, dzięki czemu do zarządzania jest potrzebne znacznie mniej kodu.

### SignalR

Biblioteka przeznaczona do komunikacji w czasie rzeczywistym w zastosowaniu do web aplikacji za pomocą połączenia HTTP i używa web socketów do przesyłania danych. Używa szyfrowania i podpisu cyfrowego do ochrony tokena połączenia. Dla każdego żądania serwer waliduje zawartość tokenu, aby upewnić się, że żądanie pochodzi od określonego użytkownika. Hosting SignalR w usługach IIS jest prosty, polega na utworzeniu witryny za pomocą SignalR Hub, a następnie opublikowanie jej w witrynie w ramach usług IIS. Zaletą jest to, że on obsługuje zarządzania połączeniami automatycznie, wysyła wiadomości do grup i pojedynczych klientów jednocześnie, umożliwia zwiększenie trafiku.

### Wzorce architektoniczne

Realizacja tego projektu jest oparta architektonicznie na wzorzec MVC (ang. Model View Controller), który polega na oddzieleniu widoku, który jest reprezentowany przez modele tabel bazy danych, w których znajdują się dane walidowane i opracowane przez kontroler. Działanie wygląda w taki sposób, że kontrolery to endpointy z aplikacji backendowej, a widok to klient aplikacji Xamarin. Do komunikacji między warstwami wykorzystałem rozszerzenie Visual Studio pod nazwą „ConveyorbyKeyoti”, które tworzy wirtualny zdalny host zamiast publikacji aplikacji na lokalny IIS lub Azure.

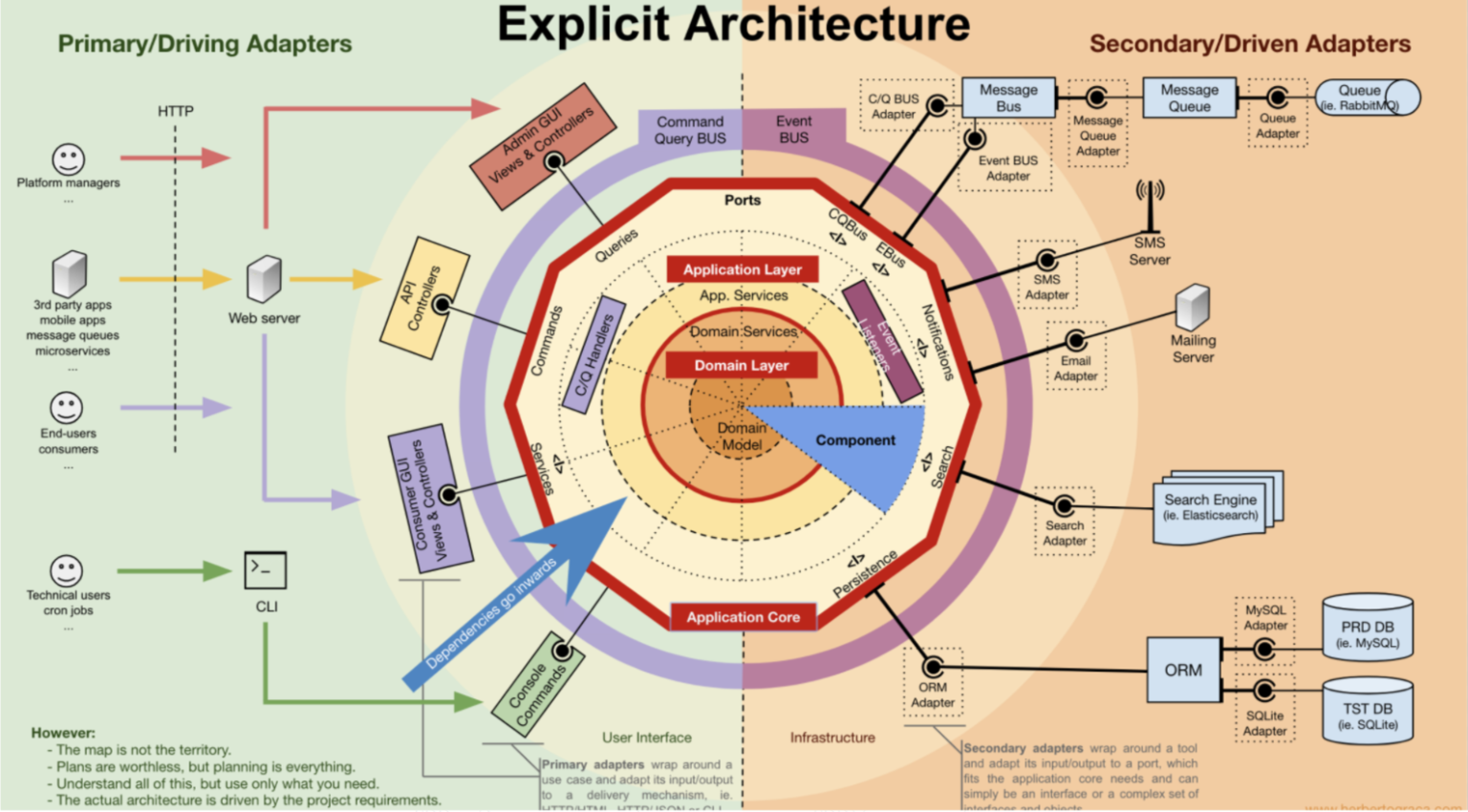
Realizacja aplikacji backendowej opiera się na DDD architekturę. Domain-driven design - zbiór zasad i schematów mających na celu tworzenie optymalnych systemów obiektów. Proces rozwoju sprowadza się do tworzenia abstrakcji oprogramowania zwanych modelami domenowymi. Modele te obejmują logikę biznesową, która łączy rzeczywiste warunki aplikacji produktu z kodem.

Co nam to daje w końcu:

* prawie wszyscy członkowie zespołu mogą odczytać kod projektu;
* zestawienie zadań staje się bardziej jednoznaczne;
* błędy logiki biznesowej stają się łatwiejsze do znalezienia;

Minusy:

* wymagane są wysokie kwalifikacje programistów, zwłaszcza na początku projektu;
* nie wszyscy klienci są gotowi na takie koszty, DDD musi być poznany przez wszystkich uczestników procesu rozwoju.



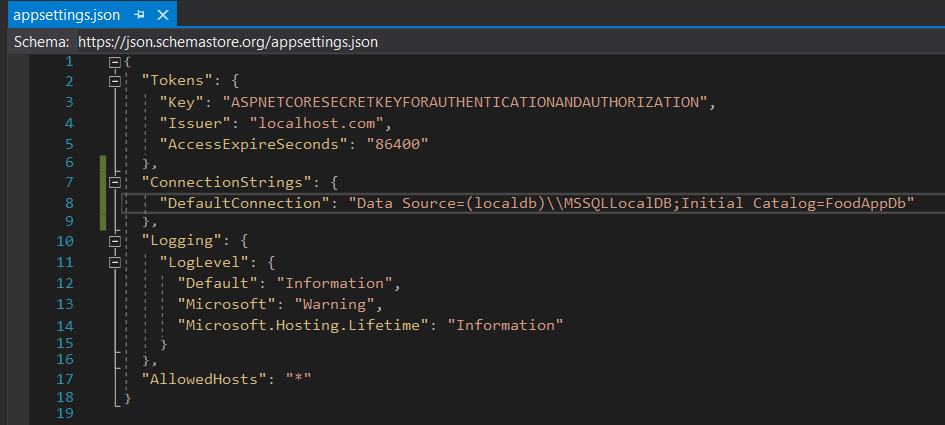
Rys. 5 <https://medium.com/the-software-architecture-chronicles/ddd-hexagonal-onion-clean-cqrs-how-i-put-it-all-together-f2590c0aa7f6>

### Wzorce projektowe

Repozytorium – został zaimplementowany w projekcie do komunikacji między warstwami Domain, Infrastructure oraz Application.

W Domain znajdują się wszystkie modele klas oraz interfejsy, które za pomocą Dependency Injection realizują dany wzorzec.

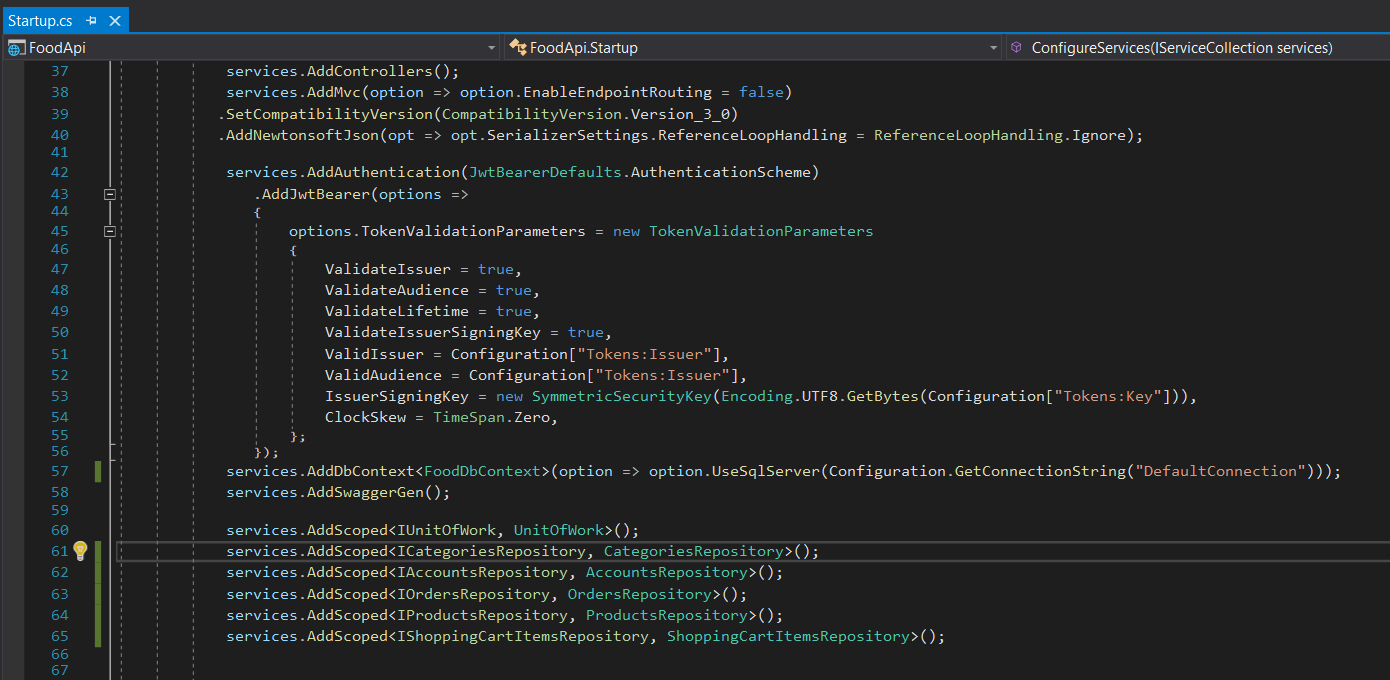
W infrastructure znajdują się migracje bazodanowe realizacja interfejsów oraz połączenie z bazą danych za pomocą kontekstu. Połączenie do bazy danych jest definiowane w appsetings.json, gdzie też znajduję się logowanie danych oraz konfiguracja JWT tokenu.



Rys. 6 konfiguracja aplikacji

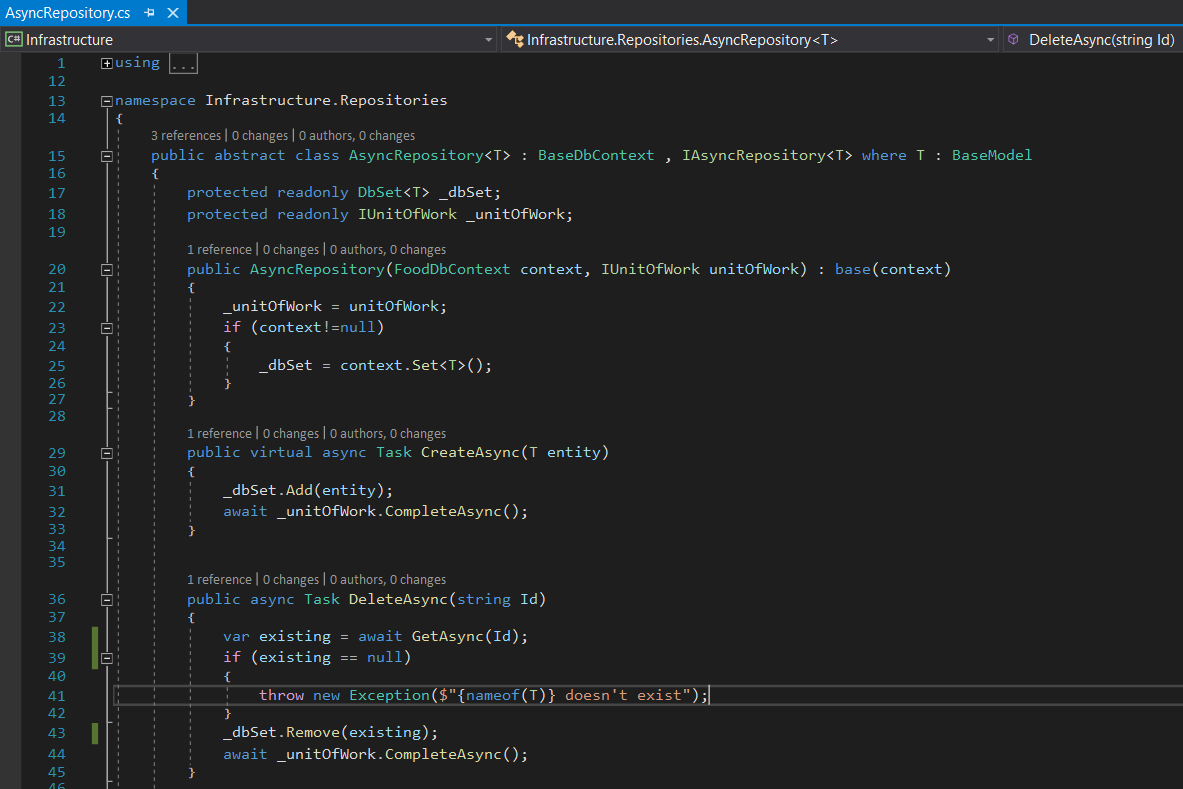
Właśnie kontrolery, które są punktami końcowymi aplikacji, przechowują się w module Application. Działa to w następny sposób:

1. Interfejsy definiują zachowywanie aplikacji, które są wstrzykiwane w Startup.cs w kontenerze IOC (*ang. Inversion of Controll*), by można było ich wykorzystać w kontrolerach.



Rys. 7 Wstrzykiwanie dostępu do serwisów za pomocą interfejsów

1. Klasy realizują metody interfejsów, które później będą wykorzystane w kontrolerze. W aplikacji też zdefiniowana klasa generyczna, która prowadzi dostęp do podstawowych metod bez konieczności ich implementowania.

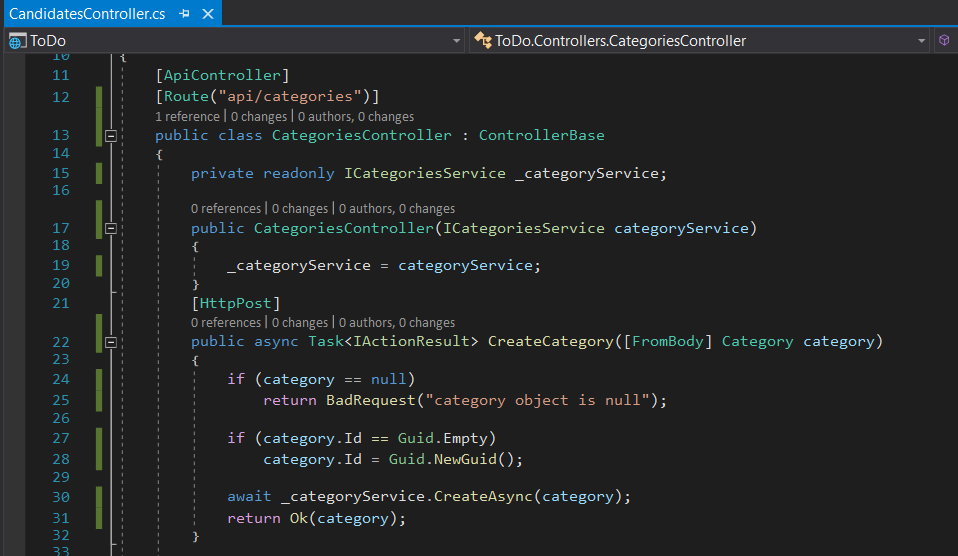


Rys. 8 Klasa generyczna która zapewnia dostęp do podstawowych metod CRUD

Takie podejście pozwala skrócić napisanie kodu, ponieważ:

* Nie tworzymy dla każdej klasy realizacje podstawowych metod
* Zachowanie aplikacji a różnych klas będzie podobne między sobą, w taki sposób kod jest bardziej czytelny
* Symbioza takich poleceń pozwala na łatwiejszą kontrolę kodu i wyłapanie bugów.

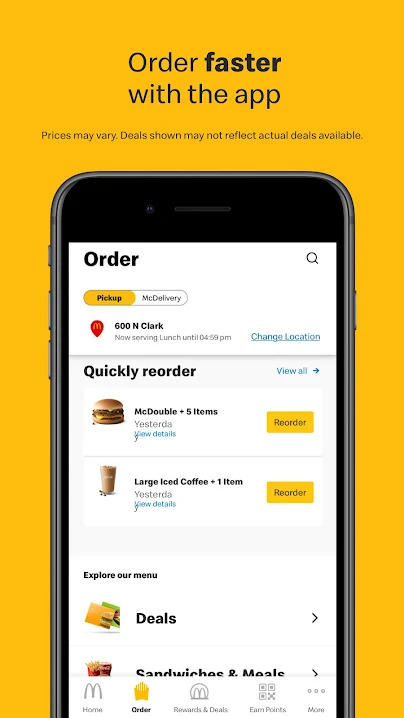
1. Kontrolery wykorzystują gotowe metody, logika jest ukryta, a zależności ładują się w kontenerach, co robi kod bardziej zabezpieczonym.



Rys. 9 Wykorzystywanie iniekcji oraz metod w kontrolerze

### Analiza rynku

Aplikacje takiego typu są powszechnie wykorzystywane w większych korporacjach, takich jak McDonald ’s, KFC, Subway i wiele innych. Przyczyną czemu takie aplikacji są wykorzystywane przez większe firm i ich wykorzystywanie będzie korzystnym dla mniejszych instytucji jest to, że telefony zawsze pod ręką i łatwe podejście do złożenia zamówienia robi to bardziej wygodnym i potencjalnie może przynieść pieniądze dla instytucji. Nie traci się czas na komunikację i złożenie zamówienia, nie tworzy się kolejki do kasy i menu zawsze pod ręką Nie mniej ważny powód to marketing i ładna aplikacja, która przyciąga klientów.



Rys. 10 Jak wyglądają różne aplikacje, w tym przykładzie McDonald's  
Źródło: <https://play.google.com/store/apps/details?id=com.mcdonalds.app&hl=en_IN&gl=US>

Z punktu widzenia marketingu, różne punkty, które można zdobyć przy zakupie też motywują dostać nagrodę za określoną ilość. Jest to wygodniejsze niż na przykład zbierać naklejki, które jeszcze mogą się zgubić.



Rys. 11 Aplikacja subway  
Źródło: <https://www.appstoreapps.com/app/subway/>

Aplikacja subway pokazuje to, jak można zaimplementować bardzo ładny widok produktu z pełną kontrolą tego, co zamawiamy. Oczywiście że dostęp do żądanych funkcji nie powinien być zbyt skomplikowany, interfejs intuitywny, a aplikacja szybko działająca.

## Część praktyczna

W tym rozdziale zostaną zdefiniowane wymagania funkcjonalne oraz niefunkcjonalne, będą przeanalizowane najważniejsze przypadki użycia oraz procesy występujące w systemie. Zostanie opisany projekt danego systemu.

### Analiza wymagań

Analiza wymagań jest to proces, który prowadzi do decyzji, co powinien oferować produkt końcowy. Od tego, w jaki sposób ona została przeprowadzona zależy cały projektu. Wymagania funkcjonalne oraz niefunkcjonalne została przygotowane na podstawie aplikacji ze światu rzeczywistego, w oparciu na dobre praktyki programowania oraz wskazania z tematu bezpieczeństwa.

Możliwość działania z aplikacją, zaczynając od najważniejszych funkcjonalności to funkcje, które bezpośrednio dotyczą interakcji z instytucją, czyli wszystkie możliwe manipulacje z produktami a koszykiem zakupów, mniej priorytetowe to czat z restauracją oraz najmniej ważne to działanie z punktami.

Dodatkowe możliwe poruszania w programie to rozszerzone możliwości dla administratorów, czyli dodawanie oraz usuwanie elementów menu oraz przegląd dziennych zamówień z możliwością wyeksportowania do JSON. Takie narzędzie może być pomocne przy tworzeniu codziennych raportów i może zautomatyzować procesy biznesowe danej instytucji, na przykład z integracją systemu ERP.

### Specyfikacja wymagań

Wymagania funkcjonalne Wydajność, bezpieczność haseł, aktualność w świecie, grabbage collector, oop, , W celu wydajności ososbno backend, osobno Xamarin.

Wymagania niefunkcjonalne crosplatformowosс, musi być internet, W celu wydajności ososbno backend, multiwatkowosc, ososbno Xamarin. rozwój aplikacji. wsparcie developerów

### Diagram przypadków użycia

### Projektowanie systemu

### Implementacja

### Prezentacja produktu końcowego

### Testy (ewaluacja)

## Podsumowanie

## Literatura

1. <https://www.outerboxdesign.com/web-design-articles/mobile-ecommerce-statistics>
2. Rakitow A.I. Informatyka, technika w globalnych wymiarach historycznych. M.: INION RAN, 1998.104 s.
3. <https://gs.statcounter.com/os-market-share>
4. <https://www.statista.com/statistics/268251/number-of-apps-in-the-itunes-app-store-since-2008/>
5. <https://medium.com/xorum-io/cross-platform-mobile-apps-development-in-2021-xamarin-vs-react-native-vs-flutter-vs-kotlin-ca8ea1f5a3e0>
6. <https://docs.microsoft.com/pl-pl/xamarin/cross-platform/app-fundamentals/building-cross-platform-applications/understanding-the-xamarin-mobile-platform>
7. <https://refactoring.guru/design-patterns/strategy>
8. Eric Evans "Domain Driven Design"

## Spis rysunków

## Streszczenie

<http://applications.lt/localhost-for-xamarin/>

