

PRACA DYPLOMOWA INŻYNIERSKA

Aplikacja dla korepetytorów z
płatnościami PayPal w technologii
.NET Core

Bartosz Jurczewski, album 210209

Politechnika Łódzka

Wydział Fizyki Technicznej, Informatyki i Matematyki

Stosowanej

Rok akademicki 2019/2020

dr hab inż Aneta Poniszewska-Marańda

Spis treści

1	Wprowadzenie	3
1.1	Problematyka	3
1.2	Cel i założenia projektu	5
1.3	Struktura pracy inżynierskiej	6
2	Idea ogłoszeń internetowych	7
2.1	Usługa jako przedmiot ogłoszenia	8
2.2	Płatności internetowe i ich bezpieczeństwo	11
2.2.1	Zwykły przelew bankowy	11
2.2.2	Przelewy pay-by-link	12
2.2.3	Karta płatnicza	14
2.2.4	Portfele elektroniczne	15
2.3	Obecne rozwiązania	16
2.3.1	Płatności	16
2.3.2	Serwisy dla korepetytorów	18
3	Techniczne aspekty <i>Find-A-Tutor</i>	20
3.1	Wymagania	20
3.1.1	Wymagania funkcjonalne	20
3.1.2	Wymagania нефункционалне	22
3.2	Narzędzia i technologie użyte w projekcie	23
3.2.1	.NET Core 2.2	24
3.2.2	C#	25
3.2.3	Kontrola dostępności serwisu	26
3.2.4	Entity Framework Core	27

3.2.5	JWT	28
3.2.6	Swagger UI	33
3.3	Architektura aplikacji - Backend	34
3.3.1	Rdzeń	36
3.3.2	Infrastruktura	36
3.3.3	API	37
3.4	Architektura aplikacji - Frontend	40
3.4.1	MVC	40
3.5	Warstwa bazy danych	44
4	Warstwa bazy danych	45
5	Wnioski	45
6	Indeks rysunków	46
7	Indeks tabel	47
8	Bibliografia	48

1 Wprowadzenie

Otacza nas rewolucja cyfrowa. Każdy aspekt naszego życia przenosi się do internetu. Jednym z nich jest rynek usług - korepetycji. Aplikacja internetowa "Find-A-Tutor" jest odpowiedzią na tę rewolucję; pozwala na łatwe zarządzanie korepetycjami - dla ucznia i nauczyciela.

1.1 Problematyka

Niewątpliwie czynnikiem który kształtował ekonomie, rynek czy cywilizacje zachodnią jako ogół była zawsze technologia. To ona bezpowrotnie zmieniła oblicze życia w latach 1780–1840 podczas rewolucji przemysłowej i to właśnie ona na nowo definiuje działania naszego świata podczas rewolucji cyfrowej. Początek tej rewolucji datowany jest na lata 70 XX wieku, ale jej koniec nie został jeszcze wyznaczony, ponieważ przyjmuje się, że trwa ona do dziś.

W roku 2019 prawie 60% populacji świata miała dostęp do komputera [1], a w samym 2019 zostało ich sprzedane prawie 230 tysięcy jednostek[2]. Obie te liczby przez ostatnie dekady miały tendencję wzrostową, co tylko jeszcze bardziej umacnia obecność technologii w naszym życiu. Niebywałą trudność może sprawić nam znalezienie takiej dziedziny życia która jeszcze nie została zmodyfikowana przedrostkiem "e" jako jednoznaczny sygnał cyfryzacji.

Pojęcie otaczającego nas wolnego rynku obejmuje również konkurencję. W warunkach idealnych, jest to model "konkurencji doskonałej", w którym to każdy z dostawców, swoją jakością produktu może rywalizować o pieniądze konsumenta. W dzisiejszych czasach konsument którego np. interesuje

wybranie i zakup książki będzie miał do wyboru ponad 10 portali, takich jak www.taniaksiążka.pl, swiatksiazki.pl, bonito.pl czy aros.pl. Każdy z tych portali oferuje ponad 400 tysięcy pozycji jak i wiele udogodnień np. bezpłatny odbiór w jednej z 125 księgarni w Polsce [3].

Przyglądając się innym branżom stosującym e-commerce (ang. handel elektroniczny) takim jak: AGD/RTV, artykuły medyczne, zoologiczne, dom i wnętrze, mililitra, motoryzacja, odzież i obuwie, sport, telefony, zdrowie i uroda, zegarki i biżuteria czy nawet żywność widzimy bogatość tych rozwiązań wraz z setkami udogodnień dla potencjalnego klienta.

Obok rynku produktów znajduje się rynek usług. Najczęściej reprezentowany przez serwis aukcyjny lub ogłoszeniowy w którym to dany specjalista może zaoferować swój czas i umiejętności za określoną kwotę. Popularnymi i największymi portalami używanymi do tego są między innymi olx.pl lub allegro.pl. Ze względu na specyfikę usługi, która nie jest namacalna tak jak produkt, przedstawienie jej w sposób dokładny może wiązać się z pewnymi trudnościami. Np. brak zdjęcia, brak możliwości zmierzenia jej jak produktu czy też po prostu mniejsza ilość sposobów na opisanie jej. Sprawia to, że są one najczęściej dostępne jako dodatkowo opcja do serwisu handlowego.

Specyficzną niszą są korepetycje. Oferowane nie tylko przez nauczycieli akademickich lub innych szkół, ale także studentów lub po prostu znawców danej dziedziny. Tradycyjnym sposobem znalezienia korepetytora było szukanie go przez polecenie lub znalezienie papierowego ogłoszenia. Kształt jak i długość lekcji były najczęściej dostosowywane dopiero przy spotkaniu z daną osobą.

Ostatnim analogowym aspektem korepetycji są oczywiście płatności -

dokonywane gotówką. Płatność taka zajmuje zawsze więcej czasu, wiąże się często z wydawaniem reszty i jest nienamierzalna (zależnie od sytuacji jest to zaleta lub wada). Od strony płaćącego wiąże się to z brakiem jakiegokolwiek historii płatności, a od strony odbiorcy płatności związane jest z jakimkolwiek brakiem historii wpływów który np. pozbawia go możliwości analizowania jego działalności od strony finansowej.

W tradycyjnym modelu korepetycji to nauczyciel ogłasza się jako osoba świadcząca te usługi. Brakuje jednak rozwiązań, a w szczególności w świecie cyfrowym, które pozwoliłyby ogłaszać się uczniom jako poszukującym korepetycji. Takie rozwiązanie niewątpliwie powinno łączyć zalety serwisów ogłoszeniowych, odwrócony model ogłaszania się (jako ucznia która tych korepetycji poszukuje) i popularnych płatności on-line. Całość oczywiście powinna być spięta w prosty i intuicyjny interfejs który swoim minimalizmem nie przytłoczy użytkownika. Co często ma miejsce w serwisach powstałych kilka lat temu i zbyt bogatych w treść.

1.2 Cel i założenia projektu

Bezpośrednim celem pracy jest dostarczenie aplikacji webowej która w łatwy sposób pozwoli na zarządzanie korepetycjami ze strony ucznia jak i nauczyciela. Aplikacja pozwoli nie tylko na ogłaszanie się i podejmowanie tych ogłoszeń, ale również na płatności internetowe PayPal'em - jako uniwersalne rozwiązanie płatności on-line.

Projekt składa się z dwóch warstw: Backend i Frontend. Frontend został napisany w technologii ASP.NET Razor (generujące dynamiczne strony internetowe po stronie serwera) i fundamentalnych technologii frontendowych

takich jak: HTML, CSS, JavaScript i Bootstrap. Wszelkie dane i operacje są wysyłane do Backendu przez protokół HTTP, za pośrednictwem JSONa. Backend to RESTful WebApi które zostało napisane w najnowszym obecnym czasie frameworku .NET Core 2.2.

Funkcjonalność rejestracji i logowania do konta, została napisana wykorzystując JWT (Json Web Token) do uwierzytelnienia użytkownika.

Aby zapewnić bezstanowość API, dane są przechowywane w bazie relacyjnej Microsoft SQL Server.

Dla skalowalności rozwiązania wszystkie usługi zostały zamknięte w kontenerach Docker, bazujących na systemie operacyjnym Linux.

1.3 Struktura pracy inżynierskiej

Pierwszy rozdział jest wstępem do opisanego problemu. Drugi opisuje ogólną ideę ogłoszeń internetowych, skupia się także nad płatnościami internetowymi, jak i opisuje ich bezpieczeństwo. Na końcu przedstawia obecne rozwiązania. Rozdział trzeci skupia się na technicznych aspektach projektu. Jest to dokumentacja techniczna opisująca architekturę aplikacji (backend) i przedstawienie warstwy bazy danych jak i graficznej (frontend). Czwarty rozdział jest opisem działania aplikacji z poziomu użytkownika. Koniec pracy to rozdział piąty który jest podsumowaniem całego projektu.

Rynek usług sprzedawanych za pomocą internetu nie jest rynkiem w całości zagospodarowanym. Ciągłe istnieją w niej niższe takie jak rynek korepetycji. Aplikacja Find-A-Tutor została przygotowana z myślą o tej niższej.

2 Idea ogłoszeń internetowych

W czasach przed internetowych, jeśli dana osoba chciała umieścić ogłoszenie lub też przejrzeć listę ogłoszeń miała do wyboru następujące opcje:

- gazeta/czasopismo
- książka telefoniczna
- lokalny bazar (rynek produktów)
- ogłoszenie na słupie ogłoszeniowym
- opowiedzieć/zapytać o to znajomych

Wszystkie opcje mimo swojej prostoty w użytkowaniu, łączyły pewne wady. Podstawową wadą był brak współdzielenia listy ogłoszeń między sobą (np. sekcja ogłoszeń jednej gazety z inną gazetą) lub z innym medium (np. gazeta z czasopismem). Ponad to: brak możliwości wyszukiwania po słowach kluczowych, bezpiecznych płatności na odległość czy też sprawdzenia opinii o danym sprzedawcy.

Po nadejściu rewolucji cyfrowej wszystkie analogowe rozwiązania ogłoszeń zaczęły odchodzić w niepamięć. Należy tutaj wspomnieć o eBayu - największy serwis aukcji internetowych na świecie, który był inspiracją dla autorów - www.allegro.pl [4]. Jest to serwis e-commerce (rozszerzony o ogłoszenia i tablice usług), który swoją ofertą dociera do 57,9% polskich internautów [5].


Jednym z największym serwisem ogłoszeniowym na świecie, a największym na rynku indyjskim, brazylijskim i **polskim** jest serwis www.olx.pl [6],


który jest częścią grupy OLX Sp. z o.o. W Polsce, w styczniu 2019, olx.pl zanotował 14,4 mln użytkowników, którzy wygenerowali 1,8 miliarda odsłon [7].

Jak pokazują dane rynek ogłoszeń jest bardzo dobrze zagospodarowany i cieszy się ogromną popularnością.

2.1 Usługa jako przedmiot ogłoszenia

Przykładowym portalem który posłuży mi do analizy systemu jest największy serwis ogłoszeniowy w Polsce - www.olx.pl.



 bartek.jurczewski ▾

Aplikacja mobilna OLX

Dodaj wygodniej przez aplikację

Szybkie dodawanie ze zdjęciami prosto z telefonu • Powiadomienie o każdej odpowiedzi


Zaczynamy!

Wpisz tytuł*

✓

46 znaków pozostało

Wybierz kategorię*


[Usługi i Firmy](#) » [Usługi](#) » [Korepetycje](#) » [Matematyka](#)

✓

Stawka za godzinę*
 zł

Prywatnie czy jako firma*

▼

Opis*

9000 znaków pozostało

Dodaj zdjęcia

Dzięki nim możesz zyskać nawet 3 razy więcej odpowiedzi

To będzie Twoje zdjęcie główne

+

+

+

+

+

+

+

+

Problem z dodaniem zdjęć? Użyj prostszego formularza

Poniższe dane dla Twojej wygody zapamiętamy

Możesz je zaktualizować w Ustawieniach »

Twoje dane kontaktowe

Podaj lokalizację*

Podaj swoje imię*

Adres e-mail*

✓

📞

Podaj numer telefonu

☐ Wyrażam zgodę na używanie przez Grupę OLX sp. z o.o. środków komunikacji elektronicznej oraz telekomunikacyjnych urządzeń końcowych w celu przesyłania mi informacji handlowych oraz prowadzenia marketingu (np. newsletter, wiadomości SMS) przez Grupę OLX sp. z o.o., podmioty powiązane i partnerów biznesowych. Moja zgoda obejmuje numery telefonów i adresy e-mail wykorzystywane podczas korzystania z usług Grupy OLX Sp. z o.o. Wyrażoną zgodę można wycofać lub ograniczyć w dowolnej chwili za pomocą odpowiednich ustawień konta lub zgłaszając nam takie żądanie.

Zobacz przed dodaniem

Rys. 2.1: Dodawanie ogłoszenia na serwisie OLX

9

Użytkownik ma do wypełnienia następujące pola:

- Tytuł
- Kategoria (w tym przypadku Usługi)
- Stawka za godzinę
- Jakim jest podmiotem gospodarczym
- Opis
- Dodanie zdjęć
- Dane kontaktowe (lokalizacja, imię, nazwisko, adres e-mail, numer telefonu)

Jedynym polem które wyróżnia usługę od przedmiotu jest pole "stawka za godzinę". Zgodnie z ideą tego serwisu, użytkownik nie ma możliwości dodania oceny/recenzji po skorzystaniu z jakiegokolwiek usługi, a co za tym idzie porównania ich po ocenach. Dodatkowo serwis ten, całkowicie odrzuca element płatności internetowych (w opozycji do eBaya czy Allegro) i polega wyłącznie na płatności gotówką. Płatności przelewami między użytkownikami są odradzane przez sam OLX, ale są oczywiście możliwe z teoretycznego punktu widzenia. Nie ma jednak wtedy żadnej gwarancji że do takiej wymiany dojdzie, ani możliwości reklamacji czy zwrotu po np. słabo wykonanej usłudze czy otrzymaniu produktu niezgodnego z opisem. Odpowiedzią rynku i rozwoju technologicznego są płatności przez internet.

2.2 Płatności internetowe i ich bezpieczeństwo

Rewolucja cyfrowa dotknęła nie tylko rynku usług i produktów ale także rynku płatności. W samym roku 2016 tylko 16% kupujących w Polsce opłaciło swoje zakupy gotówką [8]. Liczba rozwiązań płatności internetowych rośnie z każdym rokiem. Aktualnie na rynku jest ich kilka. Każde z nich różni się pod względem komfortu użytkowania, szybkości jak i bezpieczeństwa.

2.2.1 Zwykły przelew bankowy

Jest to najprostsze i najbardziej analogowe rozwiązanie. Zaraz po złożeniu zamówienia, dostajemy potwierdzenie mailem wraz z numerem konta sprzedawcy, kwoty zamówienia i tytułem przelewu (który najczęściej jest numerem zamówienia). Na kupującym ciąży zalogowanie się na stronę banku, utworzenie nowego przelewu i co istotne wypełnienie ich danymi. Oprócz możliwości pomyłki np. przekopiujemy nie pełny numer zamówienia pomijając jeden znak, ryzykiem o którym musimy pamiętać jest obecność złośliwego oprogramowania. Przykładem może być głośna sprawa z roku 2018, w którym to firma ESET natrafiła na konia trojańskiego - BackSwap. Portal www.zaufanatrzeciastrona.pl opisuje go następująco: "Do wstrzyknięcia dochodzi w momencie, gdy klient zleca przelew. Złośliwy kod podmienia wtedy numer rachunku, na który ma zostać wysłany przelew ofiary. Na ekranie komputera tego nie widać – zmiana dotyczy informacji, które przeglądarka wysyła do banku. Przestępcy nie atakują wszystkich przelewów – definiują konkretny przedział kwotowy, który ich interesuje. Ostatnio celowali w kwoty między 10 000 a 20 000 PLN." [9]. Takie ataki są niestety ciągle spotykane. Niektóre banki jak np. ING Bank Śląski, po skopiowaniu numeru

rachunku bankowego usuwa dwie pierwsze cyfry i prosi klienta o uzupełnienie ich. Bank próbuje wymusić podwójne sprawdzenie numeru konta. Efektem ubocznym będzie możliwość pomyłki przy ponownym wprowadzaniu go.

Szybkość takiego rozwiązania jest zależna od godziny wysłania przelewu. Jest to spowodowane godzinami sesji rozliczeniowych w danych bankach. Najczęściej jeśli przelew zostanie wysłany do południa, dotrze on pod wieczór. W przypadku późniejszych przelewów, dojdzie on kolejnego dnia roboczego. Wszystkie przelewy w święta i weekendy również dochodzą dopiero kolejnego dnia roboczego. Taka płatność opóźnia wysyłkę/realizację usługi, ponieważ zostanie ona wykonana dopiero wtedy gdy na rachunku pojawią się przelane środki.

2.2.2 Przelewy pay-by-link

Ta forma płatności cieszy się największą popularnością w Polsce [10]. Po wykonaniu zakupów, wybieramy opcję przelewu online, następnie z listy banków, pokazanych na rysunku 2.2 wybieramy ten z którego chcemy dokonać płatności.

2.2.3 Karta płatnicza

W maju 1998 firma Sequoia Data Corp. wprowadziła na rynek CompuMarket, pierwszy internetowy system e-commerce obsługujący płatności kartą kredytową. Od tego czasu płatności kartą zyskały wiele nowych uprawnień jednak ich rdzeń pozostał nietknięty. Aby zrealizować płatność, kupujący jest proszony o wpisanie trzech danych: numeru karty, daty jej ważności oraz trzycyfrowego kodu CVC/CVV znajdującego się na odwrocie karty. Po ich zweryfikowaniu, płatność zostaje zakończona. Ta metoda jest tak samo szybka jak pay-by-link ale mniej wygodna, ponieważ za każdym razem musimy ponownie wpisać dane karty.

Tradycyjny model zakładał nasze zaufanie do portalu w którym dokonujemy zakupów, ponieważ osoba posiadająca owe dane karty mogła samodzielnie dokonać płatności bez naszej zgody. Odpowiedzią na to jest coraz częściej obecna usługa bankowa 3D Secure. Polega ona na dodatkowym etapie weryfikacji. Po poprawnym wpisaniu danych karty i ich zweryfikowaniu, klient otrzymuje SMSa z kodem weryfikacyjnym. Dopiero po wprowadzeniu go, transakcja jest finalizowana.

Zaletą która wyróżnia ten rodzaj płatności jest usługa chargeback. Portal www.najlepszekonto.pl opisuje ją następująco: "Chargeback, czyli obciążenie zwrotne, to usługa dostępna tylko dla kart płatniczych. Polega ona na zwrocie środków z konta sprzedawcy, jeśli kupiony przez Ciebie towar nie spełnił Twoich oczekiwań: miał wady, różnił się od tego, co obiecywał sprzedawca, lub w ogóle go nie otrzymałeś. Jeśli próby wymiany towaru w sklepie nie powiodą się, możesz zwrócić się z prośbą o rozstrzygnięcie sprawy przez bank, który jest wystawcą Twojej karty." [11].

Odmianą fizycznej karty płatniczej jest karta wirtualna. Taka karta usłuży tylko do płatności internetowych. Nie użyjemy jej np. podczas zakupów w sklepie. Po jej wyrobieniu, bank nie przesyła nam jej pocztą, a po prostu podaje nam ich dane (numer, data ważności i kod CVC/CVV). Niewątpliwą zaletą jest bezpieczeństwo - nie padniemy ofiarą skimmingu (nielegalne skopiowanie zawartości paska magnetycznego w celu utworzenia kopii). Główną wadą takie rozwiązania jest dostępność w Polsce. Aktualnie tylko dwa banki udostępniają taką usługę - i to za miesięczną opłatą.

Zagranicznymi serwisami które utworzenie wirtualnych kart, jest np. amerykański www.privacy.com. Portal ten pozwala na proste i Nielimitowane tworzenie kart wirtualnych, jak możemy przeczytać na stronie firmy, "jednym kliknięciem". Klient ma możliwość tworzenia karty per portal (np. jedna karta do Netflixa, kolejna do opłacenia Spotify), dodatkowo może ustawić limit obciążenia dla danej strony, wstrzymać płatność lub całkowicie zablokować niechciane opłaty.

2.2.4 Portfele elektroniczne

Hybrydą wymienionych wcześniej rozwiązań są portfele elektroniczne zwane również portfelami cyfrowymi. Rozwiązanie te charakteryzuje się następującymi funkcjonalnościami: portfel wirtualny (rozumiany jako pula środków do wykorzystania) wraz z możliwością przelania ich bezpośrednio na rachunek bankowy, płatność wieloma podpiętymi kartami (kredytowymi jak i debetowymi), także w innych walutach, możliwość określenia adresu dostawy klienta i innych informacji ułatwiających sfinalizowanie zamówienia. Kolejnym udogodnieniem jest przechowywanie środków w różnych walutach,

dzięki czemu użytkownik nie jest obciążony podwójnym przewalutowaniem.

Niektóre portfele oferują też świadczenie usług bez jakiegokolwiek przelanych środków i potrafią służyć jako zaufany pośrednik płatności. Jeśli tylko mamy podpętą kartę/karty pod taki portfel, możemy w dowolnym miejscu zapłacić owym portfelem, a należność zostanie ściągnięta natychmiast z naszej wybranej karty. Dzięki czemu nie jesteśmy wystawieni na potencjalne wady używania samej karty jako środka płatności opisanych w rozdziale 2.2.3.

2.3 Obecne rozwiązania

Przez lata rozwiązania na rynku usług jak i płatności ewoluowały. Często stając się osobnym bytem, a nie dodatkiem jak na początku ich istnienia. Firmy odpowiedzialne za nie, przez lata starały się udoskonalić swój produkt i jak najlepiej go wypromować. Poniższe zestawienie zostało podzielone na dwie grupy: płatności (jako rdzeń serwisu usług) i serwisy dla korepetytorów.

2.3.1 Płatności

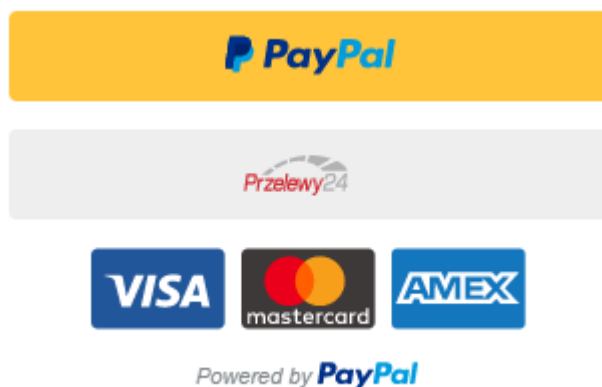
Szukając odpowiedniego rozwiązania do obsługi płatności, autor miał na celu trzy główne cechy: szybkość (jak szybko środki trafią na konto), wygodę (a w tym szybkość wykonania samej płatności) i bezpieczeństwo (transakcji jak i procesów około transakcyjnych). Każde z czterech dostępnych rozwiązań zostało szczegółowo opisane w rozdziale 2.2, właśnie pod kątem tych trzech cech. Stosując takie kryterium, autor uznaje portfel elektroniczny jako zwycięzce w tych trzech kategoriach. Dlatego w dalszym rozważaniu obecnych rozwiązań będzie brał tylko je pod uwagę.

	PayPal	Visa Checkout	Masterpass
Język polski	✓	✓	✓
Podpięcie adresu dostawy	✓	✓	✓
Uwierzytelnianie mailem	✓	✓	✗
Używanie kart różnych firm	✓	✗	✗
Możliwość dwustopniowego uwierzytelniania	✓	✗	✓
Możliwość przechowywania środków w portfelu	✓	✗	✗

Tab. 2.1: Porównanie portfeli elektronicznych

Zestawienie pokazane w tabeli 2.1 pokazuje, że PayPal wygrywa w dwóch funkcjonalnościach z Visa Checkout i Masterpassem: użycie kart różnych firm i możliwość przechowywania środków w portfelu.

Ponadto, VisaCheckout i Masterpass są realizowane jako zamknięte środowiska płatnicze. Po kliknięciu na przycisk oznaczony ich logiem, zostajemy przeniesieni do realizacji płatności. PayPal zdecydował się na inne rozwiązanie. Ich rozwiązanie daje klientowi możliwość zapłaty kartą (Visa, MasterCard i Amex), polskim portalem Przelewy24 (który agreguje płatności pay-by-link i kartą - omówione w rozdziale 2.2) i portfelem cyfrowym PayPal.



Rys. 2.3: Smart Payment Button

PayPal określa swój przycisk jako **Smart Payment Button** (ang. inteligentny przycisk do płatności), przedstawiony na rysunku 2.3. Oprócz łączenia kilku sposobów płatności, może różnić się on w każdym kraju. Np. w Polsce wspiera on portal Przelewy24. Dzięki elastyczności *smart payment button* możemy być pewni że zostanie on dostosowany do kraju z którego zostaje dokonywane zamówienie, co potencjalnie może zwiększyć zyski ze względu na bogactwo opcji płatności.

Podsumowując, w grupie portfeli elektronicznych, rozwiązanie PayPal wspiera wszystkie kluczowe i istotne dla mnie funkcjonalności. Dlatego też, to ono zostanie zaimplementowane w aplikacji *Find-A-Tutor* w celu wspierania płatności on-line.

2.3.2 Serwisy dla korepetytorów

Na rynku polskim są obecnie trzy największe serwisy dla korepetytorów. Są to www.e-korepetycje.net, korepetycje.net i wspomniany już wcześniej

portal, nie tylko do korepetycji - olx.pl. Poniżej zestawiono ze sobą cechy tych portali. Cechy zostały zaczerpnięte z rozdziału 1.1 pokrywającego problematykę pracy.

	e-korepetycje.net	korepetycje.net	olx.pl
Możliwość płatności online	✗	✗	✗
Ogłoszenia dodawane przez uczniów	✓	✗	✓
Minimalistyczny design	✓	✗	✗
Otwarte API	✗	✗	✗
Aplikacja mobilna	✗	✗	✓

Tab. 2.2: Porównanie serwisów dla korepetytorów

Żaden z portali zestawionych powyżej nie obsługuje funkcjonalności tak podstawowej w dzisiejszym świecie jak płatności elektroniczne. Tylko jeden z nich oferuje przyjazny dla oka i wpisujący się w obecne trendy - minimalisty design. Dodatkowo e-korepetycje.net i korepetycje.net nie posiadają otwartego API, które może posłużyć do stworzenia w przyszłości aplikacji mobilnych. Jednym portalem który ją dostarcza, jest OLX.pl.

Obserwując rynek korepetycji, brakuje na nim rozwiązania które wspiera płatności online. Taką aplikacją jest Find-A-Tutor. Dodatkowo, backend został stworzony jako otwarte API, dzięki czemu w przyszłości serwis będzie otwarty na powstanie aplikacji mobilnych. Fundamentalną funkcjonalnością będzie dodawanie ogłoszeń przez studentów. Całość została spięta minimalistycznym i nieprzytłaczającym designem.

3 Techniczne aspekty *Find-A-Tutor*

3.1 Wymagania

W tym rozdziale autor opisuje wymagania aplikacji webowej *Find-A-Tutor*, dzieląc je na funkcjonalne i нефункционалне.

3.1.1 Wymagania funkcjonalne

Wymagania funkcjonalne to spisane zasady, jak aplikacja ma się zachować po wystąpieniu danego zdarzenia/zapytania. Wymagania te jasno określają funkcje jakie ma mieć aplikacja, problemów jakie ma ona rozwiązywać, a także opisują jej ogólną wizję [12].

Rejestracja jako uczeń lub korepetytor

Formularz rejestracji pozwala na rejestrację jako użytkownik z wyróżnieniem ucznia i użytkownika. Oba ta konta mają inne uprawnienia i funkcjonalności opisane poniżej.

Logowanie jako uczeń, korepetytor i administrator

Dostępny panel logowania dla użytkownika (ucznia lub korepetytora). Dodatkowym rodzajem konta jest administrator, który ma pełne uprawnienia do działania na systemie. Takie konto nie posiada panelu użytkownika, ale może dokonywać wszelakich operacji opisanych w osobnym punkcie, używając zapytań do API.

Panel ucznia

Panel dostępny dla użytkownika zalogowanego jako student. Umożliwia podgląd wszystkich ogłoszeń zgłoszonych przez danego ucznia. Dodatkowo powinien on wyświetlać podstawowe dane takie jak: data utworzenia, datę

podjęcia ogłoszenia, opis, datę wygaśnięcia ogłoszenia, przedmiot, status płatności i status ogłoszenia. Logowanie następuje przez przeglądarkę internetową.

Podgląd szczegółów ogłoszenia

Uczeń powinien mieć dostęp do widoku szczegółowego ogłoszenia, który pokazuje wszystkie informacje na temat ogłoszenia.

Dodawanie ogłoszeń

Możliwość dodania ogłoszenia przez użytkownika - ucznia. Użytkownik uzupełnia następujące dane: opis, data przedawnienia ogłoszenia, przedmiot (wybrany z listy przedmiotów).

Płatności online za lekcje

Po przypisaniu ogłoszenia do korepetytora, student ma możliwość opłacenia należności za ogłoszenie online. Po opłaceniu, ma to odzwierciedlenie w statusie.

Finalizacja ogłoszenia

Po odbytej lekcji, użytkownik zaznacza zakończenie takiej lekcji, tym samym zmieniając jej status.

Panel korepetytora

Użytkownik korepetytor, po zalogowaniu ma dostęp do swojego panelu ze wszystkimi dostępnymi ogłoszeniami i ze wszystkimi podjętymi przez niego ogłoszeniami. Dostępne są na nim następujące informacje: data utworzenia, datę podjęcia ogłoszenia, opis, datę wygaśnięcia ogłoszenia, przedmiot, status płatności i status ogłoszenia. Logowanie następuje przez przeglądarkę internetową.

Podjęcie ogłoszenia

Korepetytor ma możliwość podjęcie ogłoszenia z puli nieprzydzielonych ogłoszeń. Podaje on swoją stawkę godzinową wyrażoną w złotych. Stawka ta zostaje przypisana do ogłoszenia wraz z kwotą należną za całe ogłoszenie (stawka pomnożona przez liczbę godzin). Po operacji podjęcia, zmienia on swój status.

Oddanie ogłoszenia

Jeśli ogłoszenie, po podjęciu, nie zostało jeszcze opłacone, korepetytor ma możliwość zwrócenia ogłoszenia do ogólnej puli ogłoszeń., tym samym jego status wraca do stanu wejściowego.

Konto administratora

Po zalogowaniu się zapytanie, taki użytkownik może wykonywać operacje CRUD (ang. create, read, update and delete (pol. utwórz, odczytaj, aktualizuj i usuń)) na ogłoszeniach, użytkownikach i przedmiotach. Dodatkowo ma możliwość wszystkich operacji ucznia czy korepetytora.

3.1.2 Wymagania niefunkcjonalne

Wymagania niefunkcjonalne określają obszar jakości aplikacji. Wynikają często z wymagań funkcjonalnych lub architektonicznych. Źródłem tych wymagań są przyszli użytkownicy oraz klienci. Są to np. skalowalność, kompatybilność czy bezpieczeństwo [13].

Przenośność

Aplikacja powinna być łatwa w zmianie lokalizacji, środowiska, eksportu danych.

Skalowalność

Aplikacja powinna móc się skalować pionowo (zwiększenie mocy ser-

wera) i poziomo (zwiększenie instancji aplikacji).

Otwartość na rozszerzenia

Aplikacja powinna mieć otwarte API (backend) i być gotowa na nowe urządzenia które mogą z niej korzystać (np. mobilne).

Kompatybilność

Aplikacja powinna działać w technologiach uniwersalnych i powszechnych dla przeglądarkę na silniku Chromium (Chrome, Opera i wkrótce Edge), który na rynku polskim ma 80,65% podziału rynku [14].

Bezpieczeństwo

Dane użytkowników powinny być poufne i dostępne dla danego użytkownika po zalogowaniu.

Użyteczność

Aplikacja powinna być łatwa w obsłudze i powinna być okraszona minimalistycznym designem który nie przytłacza użytkownika.

Modularność

Warstwa bazy danych, graficzna i aplikacji (API) powinny być od siebie oddzielone i gotowe do zastąpienia.

3.2 Narzędzia i technologie użyte w projekcie

Podczas realizacji aplikacji zostały wybrane tylko te technologie i rozwiązania, które wyrobiły mocną pozycję w świecie inżynierii oprogramowania. Są to narzędzia dostępne od lat, sprawdzone w wielu aplikacjach i zalecane w projektach webowych. Dodatkowo za cel przyjął sobie takie pojęcia jak czysty kod, czysta architektura oraz korzystanie z jak największej liczby gotowych rozwiązań - tak aby nie rozwiązywać po raz kolejny tych

samych problemów.

3.2.1 .NET Core 2.2

.NET Core jest darmowym i otwartym frameworkiem dla systemów Windows, Linux i macOS. Jest wielo-platformowym następcą .NET Framework. Projekt jest rozwijany przez Microsoft i dystrybuowany na zasadach otwartego oprogramowania (ang. open source). Składa się z on narzędzi programistycznych, bibliotek oraz środowiska uruchomieniowego [16].

Charakteryzują go następujące cechy:

- Możliwość programowania i uruchamiania w systemach Windows, macOS i Linux,
- Zunifikowany scenariusz tworzenia interfejsu API sieci Web i internetowych sieci Web,
- System konfiguracji oparty na środowisku, który jest gotowy do chmury,
- Wbudowane wstrzykiwanie zależności,
- Lekki, wysoko wydajny modularny potok żądań HTTP,
- Otwarte i skoncentrowane na społeczności,
- Obsługa wielu języków programowania: C# , F# i także Visual Basic,
- Obsługa menadżera pakietów NuGet,
- Elastyczny w wdrożeniu, np. kontereryzacja za pomocą Dockera

3.2.2 C#

C# (ang. C sharp) jest językiem obiektowym, opracowanym dla firmy Microsoftu, na przełomie tysiąclecia przez zespół pod kierownictwem Andersa Hejlsberga. Jak możemy przeczytać w oficjalnej dokumentacji Microsoftu: "Programiści znający którykolwiek z tych języków (C, C++, Java) są zazwyczaj w stanie zacząć produktywnie pracować w C# w bardzo krótkim czasie. Składnia w języku C# upraszcza wiele złożoności C++ i zapewnia zaawansowane funkcje, takie jak typy dopuszczające wartości zerowe, wyliczenia, delegaty, wyrażenia lambda i bezpośredni dostęp do pamięci, których nie można znaleźć w Javie." [17]

Podczas opracowywania go, zespołowi przyświecały poniższe cele:

- obiektość,
- prostota i nowoczesność,
- wsparcie dla istniejących zasad Inżynierii Oprogramowania, takich jak:
 - silne typy,
 - automatyczne czyszczenie śmieci z pamięci (ang. garbage collector),
 - wykrywanie każdej próby odniesienia się do niezainicjowanych zmiennych

Jako język obiektowy C# obsługuje pojęcia enkapsulacji, dziedziczenia i polimorfizmu. Wszystkie zmienne i metody, w tym metoda Main, punkt wejścia aplikacji, są zawarte w definicjach klas. Klasa może dziedziczyć

bezpośrednio od jednej klasy nadrzędnej, ale może implementować dowolną liczbę interfejsów.

3.2.3 Kontrola dostępności serwisu

Celem kontroli stanu projektu jest uzyskanie niezależnej oceny, w dowolnym momencie cyklu życia projektu, tego jak dobrze program działa, czy działa zgodnie z jego celami i jak dobrze przestrzega najlepszej metodologii praktyki. Skuteczna kontrola stanu (ang. health check) zapewni dostęp do informacji o stanie zależności owego projektu np. czy program ma połączenie z bazą lub pokaże w jakiej "kondycji" jest nasz program np. jaki czas zajmuje mu połączenie i wywołanie prostego zapytania na bazie.

Rozwiązaniem które dostarcza kontrole stanów prawie trzydziestu różnych komponentów jest **AspNetCore.Diagnostics.HealthChecks** [19].

```
{
  status: "Healthy",
  totalDuration: "00:00:00.5868812",
  - entries: {
    - sqlserver: {
      data: { },
      duration: "00:00:00.2960600",
      status: "Healthy",
    },
    - findATutorContext-HealthCheck: {
      data: { },
      duration: "00:00:00.2253641",
      status: "Healthy",
    },
  },
}
```

Rys. 3.1: Kontrola kondycji aplikacji *Find-A-Tutor*

Rysunek 3.1 pokazuje stan zależności projektu (backend), wraz z

dostępem do bazy danych i czasem który potrzebował na wywołanie zapytania na bazie.

3.2.4 Entity Framework Core

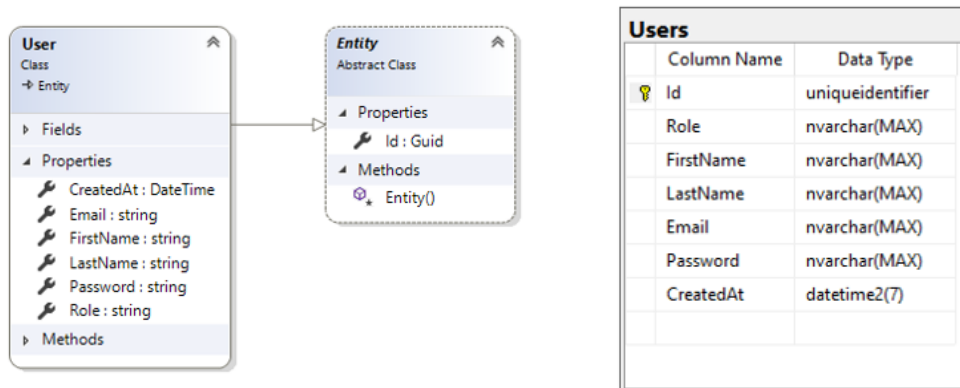
Entity Framework (EF) Core to rozszerzalny, lekki, otwarty (ang. open source) i wielo-platformowy mapper obiektowo-relacyjny (ang. Object-Relational Mapping ORM). Umożliwia on programistom pracę z bazą danych używając obiektów .NETowych. Dodatkowo, w większości eliminuje potrzebę pisania kodu do połączenia się z bazą danych.

Dzięki EF Core dostęp do danych odbywa się za pomocą modelu. Model EF przechowuje szczegółowe informacje na temat mapowania klas (wraz z ich polami) na tabele i kolumny bazy danych.

EF obsługuje trzy sposoby na realizację mapowania:

- Code First: Programista określa model przez kod. Następnie, EF generuje odpowiedni bazodanowy model na podstawie kodu i ustawień konfiguracyjnych dostarczonych przez programistę.
- Database First: tworzenie modelu z istniejącej bazy danych.
- Model First: do stworzenia bazy danych z istniejącego modelu, używa się eksportu EF.

Przykładowy sposób mapowania klasy na struktury bazodanowe, przedstawiono na rysunku 3.2.



(a) Kod klasy User (C#)

(b) Tabela Users w bazie danych

Rys. 3.2: Mapowanie klas na struktury bazodanowe

Projekt został zrealizowany według pierwszego podejścia: Code First. Przygotował on Domenę, którą następnie za pomocą migracji odwzorował na nowo utworzonej bazie danych.

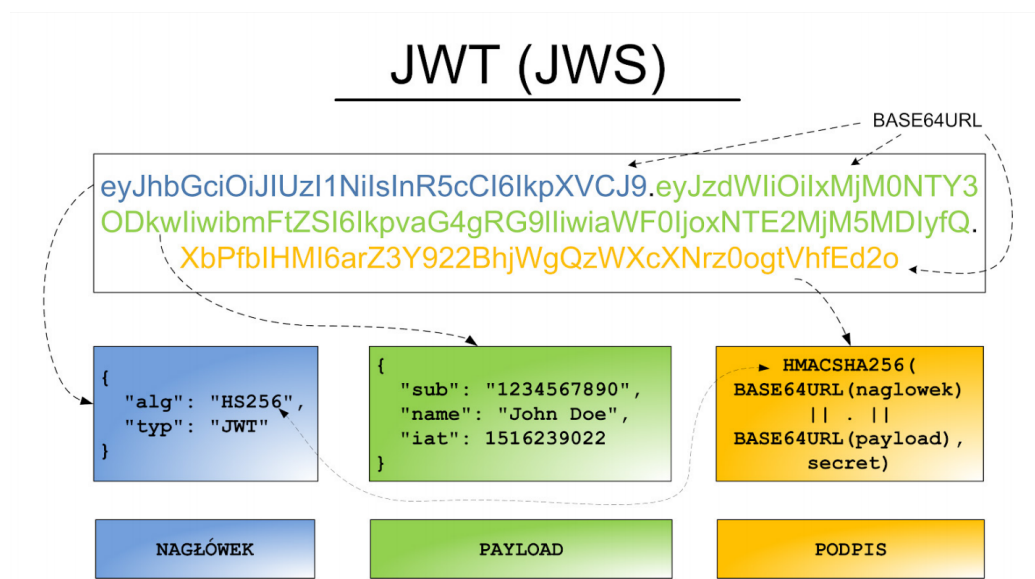
3.2.5 JWT

JSON Web Token (w skrócie JWT) to otwarty standard (RFC 7519), który odpowiada za wymianę danych pomiędzy stronami (klient-serwer) w bezpieczny sposób poprzez obiekt w formacie JSON. Informacje przekazywane w nim mogą zostać zweryfikowane przy użyciu cyfrowego podpisu, który jest zawarty w tokenie. Podpis jest generowany za pomocą algorytmu HMAC lub klucza publicznego/prywatnego RSA lub ECDSA.

JWT może zostać wykorzystany przy:

- Autoryzacji
- Transmisji danych

Struktura JWT, w swojej wynikowej postaci, składa się z trzech części oddzielonych kropkami. Są to kolejno nagłówek (ang. Header), zawartość (ang. Payload), podpis (ang. Signature). Podział ten jest pokazany na rysunku 3.3



Rys. 3.3: Przykład podstawowego JWT [21]

Nagłówek

```

1  {
2  |   "alg": "HS256",
3  |   "typ": "JWT"
4  }
```

Rys. 3.4: Przykład nagłówka tokenu

Nagłówek (pokazany na rysunku 3.4) zawiera dwie informacje: rodzaj

tokena np. JWT oraz jaki algorytm został użyty do wygenerowania podpisu np. HMAC-SHA256 (zapisywany jako HS256). W postaci końcowej (token), nagłówek jest zapisywany za pomocą algorytmu BASE64URL.

Zawartość

```
1  {  
2  |   "loggedInAs" : "admin",  
3  |   "iat" : 1422779638  
4  }
```

Rys. 3.5: Przykład zawartości tokenu

Środkowa część jest odpowiedzialna za przechowywanie danych, które chcemy zawrzeć w tokenie. Są to roszczenia (ang. claims). Specyfikacja JWT definiuje siedem zarejestrowanych roszczeń. Roszczenia niestandardowe są często wykorzystywane, w zależności od celu tokena. Przykład pokazany na rysunku 3.5 pokazuje *Issued At Time*, "iat" czyli datę wydania roszczenia i roszczenie niestandardowe *loggedInAs*.

Zawartość również jest kodowana za pomocą algorytmu BASE64URL.

Podpis

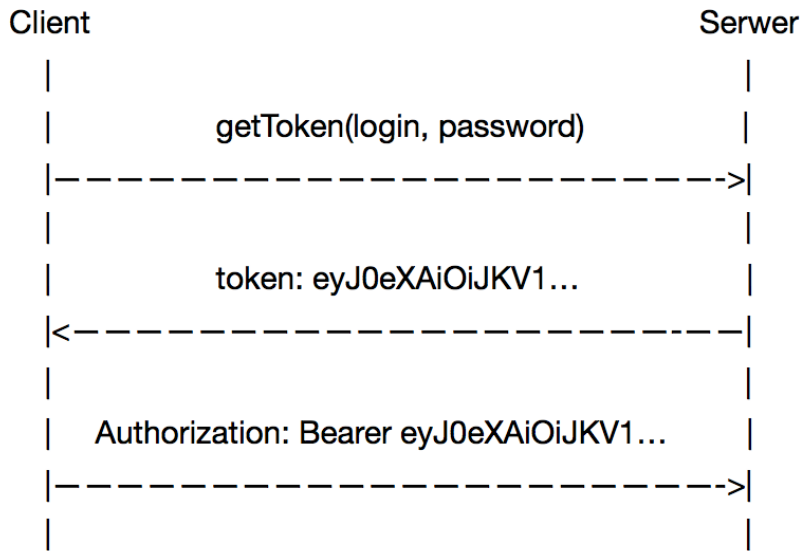
```
1 HMAC-SHA256(  
2   base64urlEncoding(header) + '.' +  
3   base64urlEncoding(payload),  
4   secret  
5 )
```

Rys. 3.6: Sposób obliczania podpisu

Podpis cyfrowy potwierdza autentyczność danych zapisanych w tokenie. Walidacja podpisu daje nam gwarancje że nadawca wiadomości jest tym za kogo się podaje.

Podpis haszowany jest za pomocą wybranego algorytmu (w pokazanym przykładzie - rysunek 3.4 był to HMAC-SHA256). Do tej procedury potrzebujemy trzech danych: zakodowanego nagłówka za pomocą algorytmu BASE64URL, zakodowanej zawartości za pomocą algorytmu BASE64URL i sekretu.

Sekret to mówiąc najprościej hasło. Serwis autoryzacyjny (w naszym przypadku serwer z API) przechowuje takie hasło i wykorzystuje przy generowaniu całego tokenu. Hasło musi być tajne, a jego kompromitacja oznacza że każdy może wygenerować taki token i przejść autoryzację w naszym portalu.



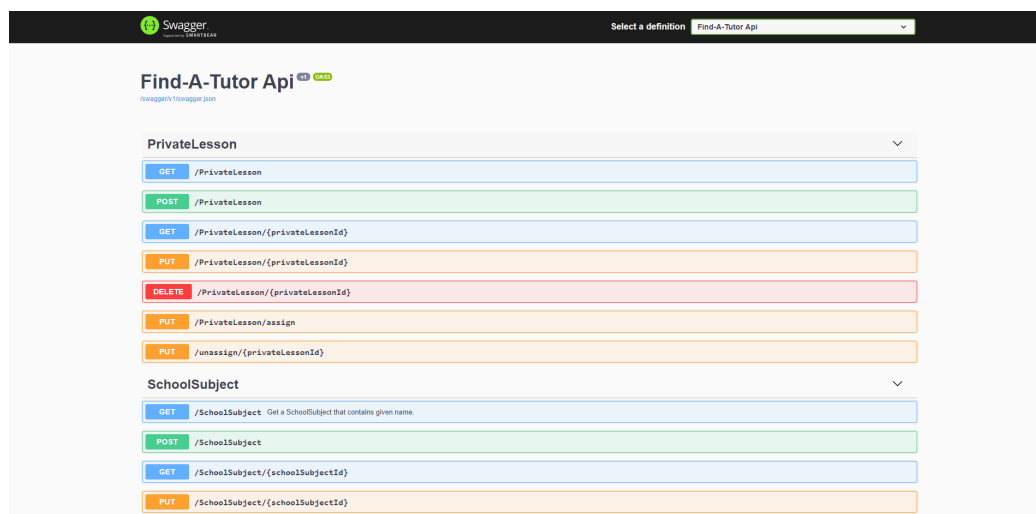
Rys. 3.7: Działanie tokenu w praktyce [22]

Rysunek 3.7 pokazuje trzy akcje:

1. Logowanie (proces uzyskania tokenu), klient przesyła login i hasło
2. Serwer (*Find-A-Tutor.API*) po pomyślnym zweryfikowaniu danych, generuje token JWT zgodnie z algorytmem pokazanym w tym rozdziale. Zwraca go jako odpowiedź do poprzedniego zapytania (logowanie)
3. Użytkownik podczas wysyłania zapytania do API przekazuje także wcześniej otrzymany token. Jeśli token jest poprawny, serwer zwróci odpowiedni dane. Jeśli nie, zwróci komunikat o nieprawidłowym tokenie (lub o jego wygaśnięciu)

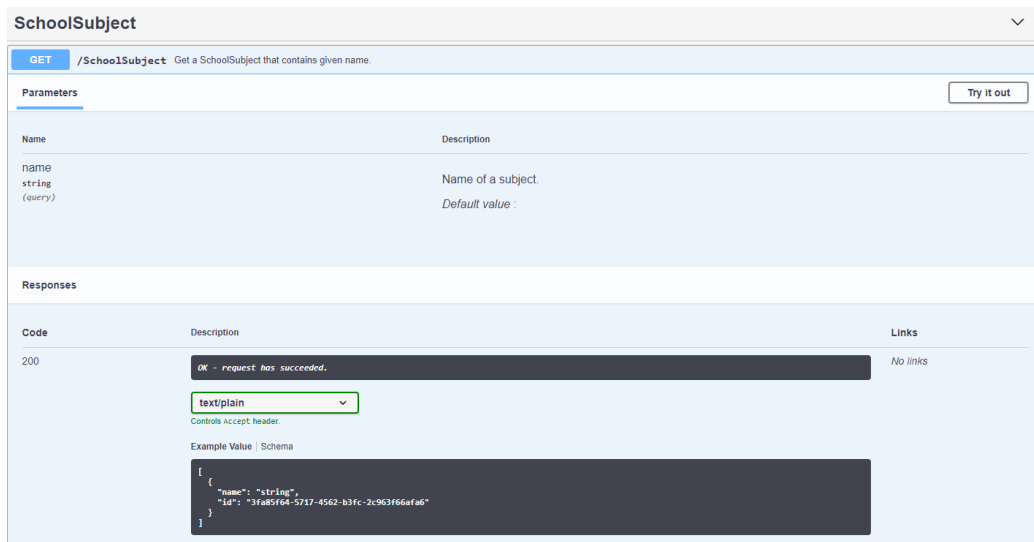
3.2.6 Swagger UI

Swagger to otwarty framework wspierany przez duży ekosystem narzędzi który pomaga programistom projektować, budować, dokumentować i użytkować WebApi. Swagger UI pozwala na wizualizację WebApi. Użytkownik ma możliwość zapoznać się ze wszystkimi endpointami (adresami) pod które może kierować zapytania. Widzi on jakie są dostępne kontrolery oraz jakie akcje udostępnia. Rysunek 3.8 pokazuje akcje dla dwóch kontrolerów (PrivateLesson i SchoolSubject).



Rys. 3.8: Interfejs Swagger UI dla aplikacji *Find-A-Tutor*

Dodatkowo, użytkownik ma możliwość przetestowania każdego zapytania, a także zapoznania się z możliwymi odpowiedziami serwera (wraz z modelem), opisem i nazwą argumentów. Na rysunku 3.9, jest to zapytanie **GET** o listę wszystkich przedmiotów szkolnych.



Rys. 3.9: Szczegóły zapytania GET

Panel Swagger UI jest dostępny pod adresem /swagger.

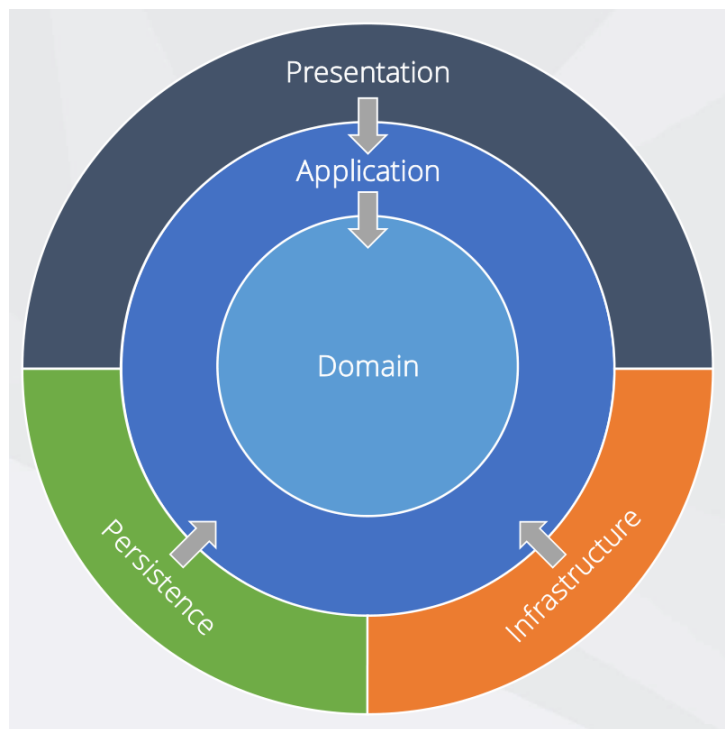
3.3 Architektura aplikacji - Backend

Aplikacje które są zgodne z *Zasadą Odwrócenia Zależności* (ang. Dependency Inversion Principle), a także z zasadami projektowania opartego na domenie (ang. Domain-Driven Design - DDD) mają tendencję do uzyskiwania podobnej architektury. Ta architektura na przestrzeni lat nosiła wiele nazw. Jedną z pierwszych nazw była *Architektura Heksagonalna* (ang. Hexagonal Architecture), a następnie *Porty I Adaptery* (ang. Ports-and-Adapters). Co raz częściej nazywa się ją *Architekturą Cebulową* (ang. Onion Architecture) lub *Czystą Architekturą* (ang. Clean Architecture). Autor w swojej pracy używa właśnie tej drugiej nazwy.

Czysta Architektura stawia logikę biznesową i model aplikacji w środku

aplikacji. Zamiast uzależniania logiki biznesowej od dostępu do danych lub innych problemów związanych z infrastrukturą, zależność ta jest odwrócona: szczegóły infrastruktury i jej implementacji zależą od rdzenia aplikacji. Osiąga się to poprzez zdefiniowanie interfejsów lub abstrakcji w rdzeniu aplikacji, które są następnie implementowane przez typy zdefiniowane w warstwie infrastruktury. Należy podkreślić że poprawnie wymodelowana domena nie posiada żadnych referencji do innych warstw aplikacji.

Częstym sposobem wizualizacji tej architektury jest użycie szeregu koncentrycznych kół, podobnych do cebuli. Rysunek 3.10 pokazuje *Czystą Architekturę* przy użyciu tego stylu reprezentacji architektonicznej. Nazwa *Cebulowa Architektura* zawdzięcza nazwę właśnie temu stylowi.



Rys. 3.10: Czysta Architektura [15]

Autor w swoim projekcie zastosował właśnie takie rozwiązanie, ze względu na czystość, prostotę implementacji, łatwość zmian (np. wymiany infrastruktury na zupełnie nową) i przede wszystkim czytelność [15].

3.3.1 Rdzeń

Centralnym elementem architektury jest rdzeń. Składa się on z *Warstwy domeny* i *Warstwy aplikacji*, pokazanej na rysunku 3.10. Obie te warstwy zawierają logikę biznesową i logikę aplikacji (wraz z zdefiniowanymi typami).

Zgodnie z *PI* (ang. Persistence Ignorance) i *Zasadą Ignorowania Infrastruktury* (ang. Infrastructure Ignorance) ta warstwa musi całkowicie ignorować szczegóły dotyczące trwałości danych (np. zapis i odczyt). Te zadania powinny zostać wykonane przez warstwę infrastruktury. Dlatego ta warstwa nie powinna przyjmować bezpośrednich zależności od infrastruktury, co oznacza że ważne jest aby klasy reprezentujące encję modelowe były obiektami POCO.

Domenowe encję nie powinny mieć żadnej bezpośredniej zależności (jak dziedziczenie) do frameworka służącego do dostępu danych jak np. Entity Framework (technologia szczegółowo opisana w rozdziale 3.2.3).

3.3.2 Infrastruktura

Warstwa infrastruktury i utrwalenia (ang. persistence) opisuje sposób w jaki dane które są początkowo trzymane w klasach modelowych (w pamięci) są utrwalane w bazie danych lub w jakiegokolwiek innej postaci. Przykładem takiego utrwalenia (który Autor zaimplementował) jest Entity Framework Core. Napisany kod służy do połączanie się z bazą danych przez DbContext,

który jest reprezentacją bazy danych jako obiekt. Dzięki temu w łatwy sposób możemy operować danymi w naszej bazie danych.

Ta warstwa zależy od warstw niżej - aplikacji i domeny.

3.3.3 API

Warstwą odpowiedzialną za kontakt "na zewnątrz" jest warstwa prezentacji. W projekcie backendowym, zaimplementowana została ona jako WebApi, wraz z najpopularniejszym stylem architektury REST. Za format wymiany danych posłużył JSON.

RESTful WebAPI

WebApi jest to aplikacja z którą komunikacja jest ograniczona do protokołu HTTP. Protokół ten zawiera 9 metod przy pomocy których możemy sterować WebApi. HTTP jest niezależny od platformy i jest bezstanowy - nie przechowuje żadnych informacji poprzednich transakcjach z klientem).

Aplikacja taka wystawia na świat adresy URL, dalej zwane endpointami. Endpointy są rdzeniem aplikacji WebAPI. To właśnie do nich użytkownik końcowy kieruje zapytania. Jako przykładem, Autor posłuży się następującym endpointem - **http://domena/privatelesson/3**. Użytkownika interesuje zasób "privatelesson" o ID równym 3. Korzystając z jednej metod protokołu HTTP - GET, wysyłamy owe zapytanie.

Gdyby użytkownik skorzystał z metody PUT, mógłby on zaktualizować dane znajdujące się pod tym endpointem. Wszystkie metody są opisane przez dokument **RFC2616** [23].

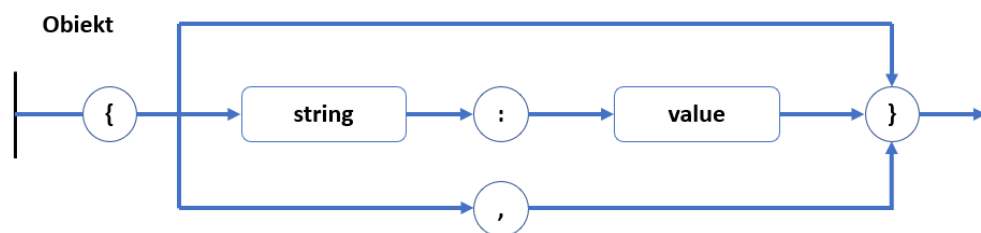
REST (ang. Representational State Transfer) jest wzorcem architekton-

icznym który narzuca dobre praktyki przy tworzeniu aplikacji. RESTful WebAPI to WebApi które stosuje się do zasad wzorca REST i jest zaimplementowane na bazie protokołu HTTP.

JSON

JavaScript Object Notation, w skrócie JSON to otwarty i lekki format wymiany danych komputerowych. Charakteryzuje się tekstem czytelnym dla człowieka. Używany jest do przesyłania obiektów danych składających się z par atrybut-wartość i tablicowych typów danych (lub każdej innej wartości możliwej do serializacji) [20].

Sam zapis jest dokonywany tylko przy użyciu sześciu znaków: {, }, [,], : (dwukropek) oraz , (przecinek). Proces ten został zwizualizowany na rysunku 3.11.



Rys. 3.11: Schemat zapisu danych typu obiekt w formacie JSON

JSON jest niezależny od języka. Został zaczerpnięty z JavaScriptu, ale prawie wszystkie nowoczesne języki programowania zawierają funkcjonalność generowania i parsowania danych sformatowanych za pomocą JSONa. Dlatego też stał się on uniwersalnym formatem do deserializacji danych, przesyłania ich do/z WebApi, jak i nawet przechowywaniu ich na dysku.

W *Find-A-Tutor* stał się on jednym formatem wymiany danych - WebAPI, przyjmuje i zwraca dane w JSONie.

```
1  {
2    "value": [
3      {
4        "id": "ab0162e3-fac8-478d-90e2-148b3611aaf0",
5        "studentId": "87791f6e-8141-4bb2-ae95-366e2bdc8b35",
6        "createdAt": "2019-12-22T11:29:23.3739482",
7        "updatedAt": "0001-01-01T00:00:00",
8        "relevantTo": "2020-07-19T12:41:51.297983",
9        "description": "Nowe ogłoszenie - potrzeba korepetycji z biologii",
10       "isAssigned": false,
11       "isPaid": false,
12       "isDone": false,
13       "time": 2.0,
14       "pricePerHour": 0.0,
15       "totalPrice": 0.0
16     },
17     {
18       "id": "54d2edaf-cebd-4d21-a0e1-3b5d6ac79519",
19       "studentId": "87791f6e-8141-4bb2-ae95-366e2bdc8b35",
20       "tutorId": "bd37598c-e1a3-4e52-bc8b-ec65f9d8220f",
21       "createdAt": "2019-11-17T16:33:29.3759055",
22       "takenAt": "2019-11-18T14:03:49.9786449",
23       "updatedAt": "2019-11-18T14:03:49.9786451",
24       "relevantTo": "2020-11-17T00:00:00",
25       "description": "Potrzebne korepetycje bardzo pilne, student 4 roku",
26       "isAssigned": true,
27       "isPaid": true,
28       "isDone": false,
29       "time": 3.0,
30       "pricePerHour": 1.0,
31       "totalPrice": 3.0
32     }
33   ],
34   "isSuccess": true
35 }
```

Rys. 3.12: Rezultat wysłanego zapytania do WebAPI

Na rysunku 3.12 pokazany jest rezultat zapytania metodą GET do WebApi. Są to wszystkie ogłoszenia danego użytkownika (dodane z jego konta).

3.4 Architektura aplikacji - Frontend

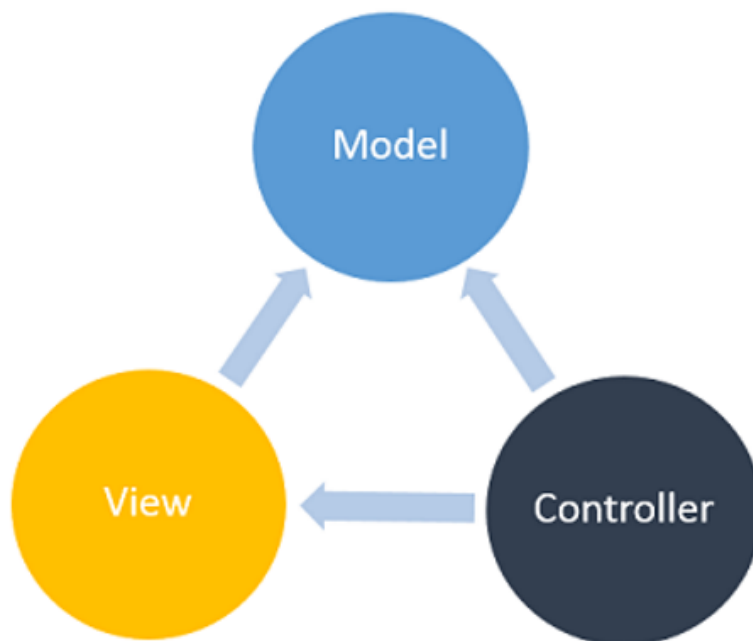
Najmniejsza możliwa liczba projektów dla architektury aplikacji to jeden. W takiej aplikacji, cała logika jest zawarta w jednym projekcie, kompilowana do jednego pliku wykonywalnego i wdrożona jako jedna całość.

Ze względu na podział backend - frontend, to właśnie API (backend) odpowiada za faktyczną logikę projektu. I to przez ten interfejs, aplikacja frontendowa porozumiewa się z serwerem czy bazą danych.

W praktyce każda akcja wykonana na froncie (np. logowanie, pobieranie danych) odbywa się poprzez wysłanie odpowiedniego zapytania do API.

3.4.1 MVC

Model-view-controller (pol. Model-Widok-Kontroler), to wzorzec architektoniczny powszechnie używany do opracowania aplikacji posiadających graficzne interfejsy użytkownika.



Rys. 3.13: Model-Widok-Kontroler [24]

MVC dzieli aplikację na trzy części:

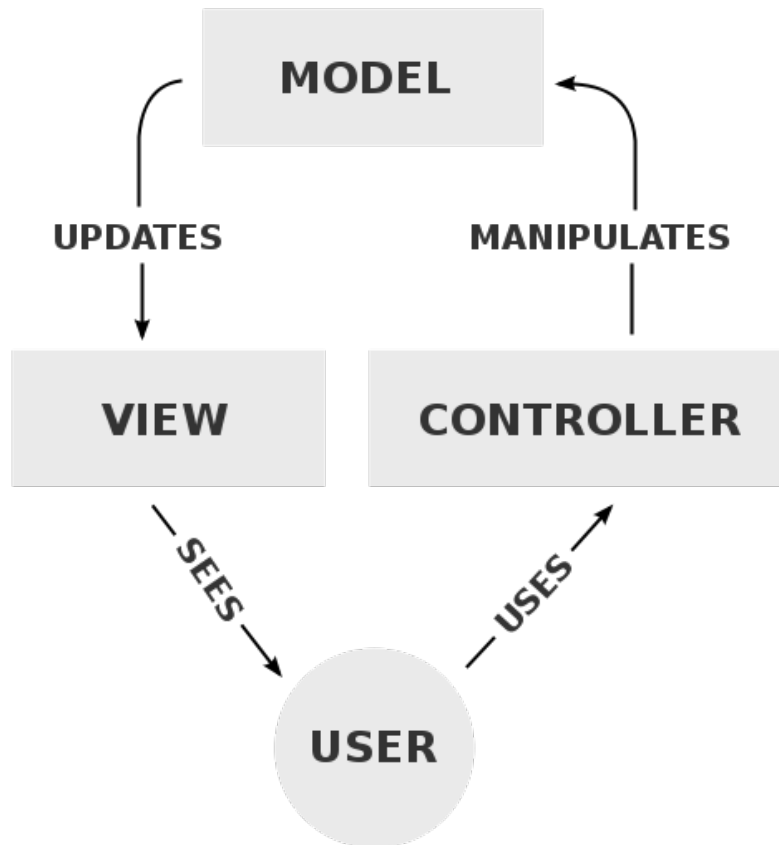
- Model - logika biznesowa
- Widok - graficzne interfejsy dla użytkownika (UI od ang. user interface)
- Kontroler - obsługuje żądania użytkownika i konwertuje je na polecenia widoku lub modelu

Wszystkie jego części są ze sobą połączone, pokazuje to rysunek 3.13. Aplikacja frontendowa została zaprojektowana używając właśnie tej architektury. Prześledźmy zatem pojedynczą akcję użytkownika - logowanie:

1. Użytkownik zostaje zwrócony odpowiedni UI (Widok)

2. Użytkownik wpisuje w pola login i hasło
3. Aplikacja obsługuje te dane i przekazuje dalej (Kontroler)
4. Aplikacja opakuje owe dane (JSON) i wysyła jako zapytanie HTTP do WebApi (Model)
 - WebApi otrzymuje dane i je przetwarza, cały proces został opisany w rozdziale 3.2.5
5. WebApi zwraca dane do aplikacji, są one rozpakowywane, i przekazywane dalej (Model)
6. Model aktualizuje Widok, co jest widoczne dla użytkownika

Przepływ danych został pokazany na rysunku 3.14.



Rys. 3.14: Zasada organizacji MVC [25]

Przedstawiony scenariusz pokazuje że frontend został napisany na zasadach klienta który obsługuje żądania użytkownika i usprawnia (np. przez UI) komunikacje użytkownika z serwerem. Rozdzielenie tych dwóch projektów ma następujące zalety:

- Modularność - w razie potrzeby jesteśmy w stanie wymienić jeden z projektów na inny

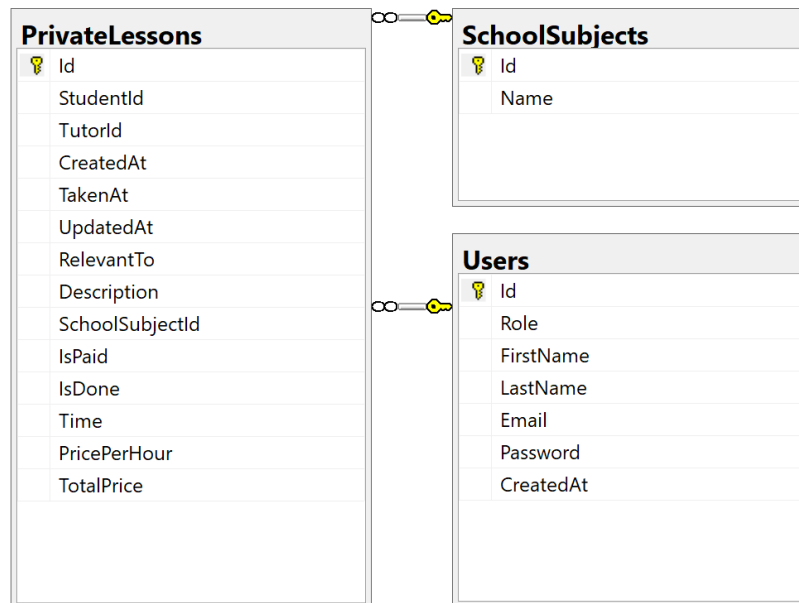
- W razie zmiany popularny stylów graficznych, można w łatwy sposób napisać sam frontend od nowa, dzięki czemu cała aplikacja zostaje nietknięta
- Ułatwienie w utrzymaniu i pracy nad projektem
- Możliwość pracy w dwóch osobnych zespołach na obu projektach

3.5 Warstwa bazy danych

Ze względu na charakterystykę aplikacji, Autor zdecydował się na bazę relacyjną. Zgodnie z wymodelowaną domeną opisaną w rozdziale 3.3.1 i korzystając z mappera obiektowo-relacyjnego (EF Core) opisanego w rozdziale 3.2.3, struktura bazodanowa została wygenerowana za pomocą migracji. Autor zastosował obsługiwany sposób Code First, czyli na podstawie kodu, EF wygenerował model bazodanowy.

Dodatkowo, Autor wybrał *Microsoft SQL Server* jako silnik bazodanowy, mając na uwadze że i owy silnik, jak i platforma .Net Core (wraz z C#) została opracowana przez firmę Microsoft.

Poniżej, rysunek 3.15 przedstawia strukturę bazodanową:



Rys. 3.15: Struktury bazodanowe aplikacji *Find-A-Tutor*

Podczas powstawania aplikacji Find-A-Tutor Autor postarał się aby była ona od samego początku dobrze zaprojektowana tak aby później modyfikacje nie komplikowały nadmiernie projektu.

4 Warstwa bazy danych

5 Wnioski

6 Indeks rysunków

2.1	Dodawanie ogłoszenia na serwisie OLX	9
2.2	Przykładowa płatność pay-by-link	13
2.3	Smart Payment Button	18
3.1	Kontrola kondycji aplikacji <i>Find-A-Tutor</i>	26
3.2	Mapowanie klas na struktury bazodanowe	28
3.3	Przykład podstawowego JWT [21]	29
3.4	Przykład nagłówka tokenu	29
3.5	Przykład zawartości tokenu	30
3.6	Sposób obliczania podpisu	31
3.7	Działanie tokenu w praktyce [22]	32
3.8	Interfejs Swagger UI dla aplikacji <i>Find-A-Tutor</i>	33
3.9	Szczegóły zapytania GET	34
3.10	Czysta Architektura [15]	35
3.11	Schemat zapisu danych typu obiekt w formacie JSON	38
3.12	Rezultat wysłanego zapytania do WebAPI	39
3.13	Model-Widok-Kontroler [24]	41
3.14	Zasada organizacji MVC [25]	43
3.15	Struktury bazodanowe aplikacji <i>Find-A-Tutor</i>	45

7 Indeks tabel

2.1	Porównanie portfeli elektronicznych	17
2.2	Porównanie serwisów dla korepetytorów	19

8 Bibliografia

- [1] World Internet Users and 2019 Population Stats <https://www.internetworldstats.com/stats.htm>, dostęp 24-11-2019
- [2] Computers sold this year <https://www.worldometers.info/computers/>, dostęp 24-11-2019
- [3] Ranking sklepów internetowych 2019 <http://static.opineo.pl/press/dl/ranking-sklepow-internetowych-opineo-2019.pdf>,
dostęp 28-11-2019
- [4] Bakker o Allegro.pl: Cel był jasny. W e-commerce mógł być tylko jeden gracz <https://polskatimes.pl/bakker-o-allegropl-cel-byl-jasny-w-ecommerce-mogl-byc-tylko-jeden-gracz/ar/664791>, dostęp 28-11-2019
- [5] Wyniki badania Gemius/PBI za lipiec 2017 <https://www.gemius.pl/wydawcy-aktualnosci/wyniki-badania-gemiuspbi-za-lipiec-2017.html>,
dostęp 28-11-2019
- [6] Meet OLX, the biggest Web company you've never heard of <https://fortune.com/2014/10/29/olx-emerging-markets/>,
dostęp 28-11-2019
- [7] Wyniki badania Gemius/PBI za styczeń 2019 <https://www.gemius.pl/wszystkie-artykuly-aktualnosci/>

- wyniki-badania-gemiuspbi-za-styczen-2019.html, dostęp 28-11-2019
- [8] Sondaż: Jak Polacy płacą w Internecie? <https://www.kir.pl/o-nas/aktualnosci/sondaz-jak-polacy-placa-w-internecie,142.html>, dostęp 30-11-2019
- [9] Klienci PKO BP, BZ WBK, mBanku, ING i Pekao na celowniku nowego malware <https://zaufanatrzeciastrona.pl/post/klienci-pko-bp-bz-wbk-mbanku-ing-i-pekao-na-celowniku-nowego-malware/>, dostęp 30-11-2019
- [10] Jak Polacy kupują i płacą przez internet? Co lubią, czego się boją? <https://www.shoper.pl/blog/jak-polacy-kupuja-i-placa-przez-internet/>, dostęp 1-12-2019
- [11] Płatność kartą przez Internet <https://www.najlepszekonto.pl/platnosc-karta-przez-internet>, dostęp 2-12-2019
- [12] Wymagania funkcjonalne aplikacji internetowej <http://www.commint.pl/baza/wymagania-funkcjonalne-aplikacji-internetowej>, dostęp 10-12-2019
- [13] Wymagania niefunkcjonalne aplikacji internetowej <http://www.commint.pl/baza/wymagania-niefunkcjonalne-aplikacji-internetowej>, dostęp 10-12-2019
- [14] Browser Market Share Poland <https://gs.statcounter.com/browser-market-share/all/poland>, dostęp 10-12-2019

- [15] Clean Architecture with ASP.NET Core 2.2 <https://github.com/JasonGT/NorthwindTraders/blob/master/Docs/Slides.pdf>, dostęp 19-12-2019 https://www.youtube.com/watch?v=_lwCVE_XgqI, dostęp 20-12-2019
- [16] Wprowadzenie do platformy ASP.NET Core <https://docs.microsoft.com/pl-pl/aspnet/core/?view=aspnetcore-2.1>, dostęp 21-12-2019
- [17] Wprowadzenie do języka C# i systemu .NET Framework <https://docs.microsoft.com/pl-pl/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework>, dostęp 21-12-2019
- [18] Health checks in ASP.NET Core <https://docs.microsoft.com/en-gb/aspnet/core/host-and-deploy/health-checks?view=aspnetcore-3.1>, dostęp 21-12-2019
- [19] AspNetCore.Diagnostics.HealthChecks <https://github.com/xabari/AspNetCore.Diagnostics.HealthChecks>, dostęp 21-12-2019
- [20] Introducing JSON <https://www.json.org/json-en.html>, dostęp 21-12-2019
- [21] (Nie) bezpieczeństwa JWT (JSON Web Token) <https://sekurak.pl/json-security-ebook.pdf>, dostęp 25-12-2019
- [22] JSON Web Tokens (JWT) <https://blog.i-systems.pl/json-web-tokens-jwt/>, dostęp 27-12-2019

- [23] Hypertext Transfer Protocol – HTTP/1.1 <https://www.ietf.org/rfc/rfc2616.txt>, dostęp 28-12-2019
- [24] ASP .Net MVC – pojęcia podstawowe <https://www.wojciechseweryn.pl/2018/07/07/asp-net-mvc-pojecia-podstawowe/>, dostęp 30-12-2019
- [25] Paradygmat MVC Model-View-Controller <https://farmastron.pl/paradygmat-mvc-model-view-controller>, dostęp 30-12-2019