

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

Promotor: dr hab. inż. Aneta Poniszewska-Marańda

Współpromotor: mgr inż. Krzysztof Stępień

Spis treści

1	Wprowadzenie	5
1.1	Problematyka	5
1.2	Cel i założenia projektu	7
1.3	Struktura pracy inżynierskiej	8
2	Idea ogłoszeń internetowych	9
2.1	Usługa jako przedmiot ogłoszenia	10
2.2	Płatności internetowe i ich bezpieczeństwo	12
2.2.1	Zwykły przelew bankowy	12
2.2.2	Przelewy pay-by-link	14
2.2.3	Karta płatnicza	15
2.2.4	Portfele elektroniczne	17
2.3	Analiza istniejących płatności i portali dla korepetytorów . . .	18
2.3.1	Płatności	18
2.3.2	Serwisy dla korepetytorów	20
3	Techniczne aspekty aplikacji <i>Find-A-Tutor</i>	22
3.1	Wymagania	22
3.1.1	Wymagania funkcjonalne	22
3.1.2	Wymagania нефункционалне	25
3.2	Narzędzia i technologie użyte w projekcie	26
3.2.1	.NET Core 2.2	26
3.2.2	C#	27
3.2.3	Kontrola dostępności serwisu	28
3.2.4	Entity Framework Core	29

3.2.5	JWT	31
3.2.6	Swagger UI	36
3.3	Architektura aplikacji – Backend	37
3.3.1	Rdzeń aplikacji	39
3.3.2	Infrastruktura aplikacji	40
3.3.3	API aplikacji	41
3.4	Architektura aplikacji – frontend	44
3.4.1	MVC	44
3.5	Warstwa bazy danych	47
4	Aplikacja z punktu widzenia użytkownika	49
4.1	Uwierzytelnianie	49
4.2	Panel użytkownika	50
4.2.1	Dodanie nowego ogłoszenia	51
4.2.2	Podjęcie ogłoszenia	54
4.2.3	Płatność	56
4.3	Wylogowanie	58
5	Podsumowanie	60
6	Indeks rysunków	62
7	Indeks tabel	64
8	Bibliografia	65

Abstract

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 ekonomię, rynek lub cywilizację 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-te XX wieku, ale jej koniec nie został jeszcze wyznaczony, ponieważ przyjmuje się, że trwa ona do dziś.

W 2019 roku prawie 60% populacji świata miała dostęp do komputera [1], a w samym 2019 zostało ich sprzedanych 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 interesuje na przykład wybranie i zakup książki będzie miał do wyboru ponad 10 portali, takich jak www.taniaksiążka.pl, swiatksiazki.pl, bonito.pl lub aros.pl. Każdy z tych portali oferuje ponad 400 tysięcy pozycji, jak i wiele udogodnień, na przykład bezpłatny odbiór w jednej z 125 księgarni w Polsce [3].

Przeglądając się innym branżom stosującym e-commerce, czyli 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, a nawet żywność, widzimy bogactwo tych rozwiązań wraz z setkami udogodnień dla potencjalnego klienta.

Obok rynku produktów znajduje się rynek usług. Najczęściej reprezentowany jest przez serwis aukcyjny lub ogłoszeniowy, w którym dany specjalista może zaoferować swój czas i umiejętności za określoną kwotę. Popularnymi i największymi portalami używanymi do tego celu 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, przykładowo brak zdjęcia, brak możliwości zmierzenia jej jak produktu lub po prostu mniejsza liczba sposobów na jej opisanie. Sprawia to, że usługi są najczęściej dostępne jako dodatkowa opcja do serwisu handlowego.

Specyficzną niszą są korepetycje. Oferowane są nie tylko przez nauczycieli akademickich lub nauczycieli innych szkół, ale także przez studentów lub po prostu przez znawców danej dziedziny. Tradycyjnym sposobem znalezienia korepetytora jest szukanie go przez polecenie lub przez papierowe ogłoszenia.

Kształt, jak i długość lekcji są 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 płacącego wiąże się to z brakiem jakiegokolwiek historii płatności, a od strony odbiorcy płatności związane jest z jakimkolwiek brakiem historii wpływów, który pozbawia go na przykład możliwości analizowania jego działalności od strony finansowej.

W tradycyjnym modelu korepetycji to nauczyciel ogłasza się jako osoba świadcząca te usługi. Brakuje jednak rozwiązań, w szczególności w świecie cyfrowym, które pozwoliłyby ogłaszać się uczniom jako poszukującym korepetycji. Takie rozwiązanie niewątpliwie powinno łączyć zalety serwisów ogłoszeniowych, odwróconego modelu ogłaszania się (jako ucznia, który 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

Celem niniejszej pracy dyplomowej jest stworzenie aplikacji webowej, która w łatwy sposób pozwoli na zarządzanie korepetycjami ze strony ucznia, jak i ze strony nauczyciela. Aplikacja ma pozwolić nie tylko na ogłaszanie się i podejmowanie tych ogłoszeń, ale również na płatności internetowe usługą PayPal, jako uniwersalnym rozwiązaniem płatności on-line.

Zakres pracy obejmuje implementacje aplikacji serwerowej w technologii *.Net Core* i aplikacji klienckiej w technologii *Razor Pages*. Ponadto, zostaną szczegółowo omówione i porównane aktualne rozwiązania na rynku płatności online, jak i na rynku korepetycji.

1.3 Struktura pracy inżynierskiej

Pierwszy rozdział pracy jest wstępem do opisanego problemu. Drugi prezentuje ogólną idee ogłoszeń internetowych, skupia się także na płatnościach internetowych oraz opisuje ich bezpieczeństwo. Na końcu przedstawia obecne systemy płatności. Rozdział trzeci pracy skupia się na technicznych aspektach projektu. Opisuje architekturę aplikacji serwerowej (backend) i interfejsu użytkownika (frontend). Czwarty rozdział jest opisem działania stworzonej aplikacji z punktu widzenia użytkownika.

2 Idea ogłoszeń internetowych

W czasach przedinternetowych, 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 przedmiot ogłoszenia znajomych.

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

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

Jednym z największych serwisów ogłoszeniowych 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 powyższe dane rynek ogłoszeń jest bardzo dobrze zagospodarowany i cieszy się ogromną popularnością.

2.1 Usługa jako przedmiot ogłoszenia

Przykładowym portalem, który posłużył do analizy jest największy serwis ogłoszeniowy w Polsce czyli www.olx.pl.

Rysunek 2.1 pokazuje panel dodawania nowych ogłoszeń. Użytkownik tego serwisu ma do wypełnianie następujące pola:

- tytuł ogłoszenia,
- kategoria ogłoszenia(w tym przypadku Usługi),
- stawka za godzinę,
- jakim jest podmiotem gospodarczym,
- opis ogłoszenia,
- dodanie zdjęć przedmiotu ogłoszenia,
- 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



Zaczynamy!

Wpisz tytuł*

Korepetycje z matematyki



46 znaków pozostało

Wybierz kategorię*



Usługi i Firmy » Usługi » Korepetycje » Matematyka

Zmień



Stawka za godzinę*

zł

Prywatnie czy jako firma*

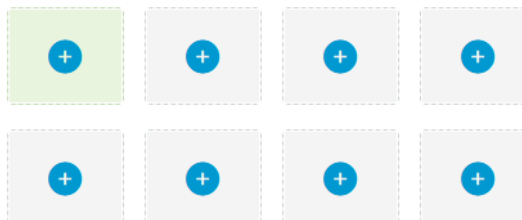


Opis*

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](#)

Twoje dane kontaktowe

Podaj lokalizację*

Podaj swoje imię*

Adres e-mail*

bartek.jurczewski@gmail.com



Podaj numer telefonu

Rys. 2.1: Dodawanie ogłoszenia na serwisie OLX

dodania oceny/recenzji po skorzystaniu z jakiegokolwiek usługi, a co za tym idzie porównania wykonawców usług według ocen. Dodatkowo serwis ten, całkowicie odrzuca element płatności internetowych (w przeciwieństwie do eBaya lub Allegro) i opiera się na płatnościach w świecie rzeczywistym. 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 lub zwrotu po przykładowo słabo wykonanej usłudze lub 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 2016 roku 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 i bezpieczeństwa.

2.2.1 Zwyczajny przelew bankowy

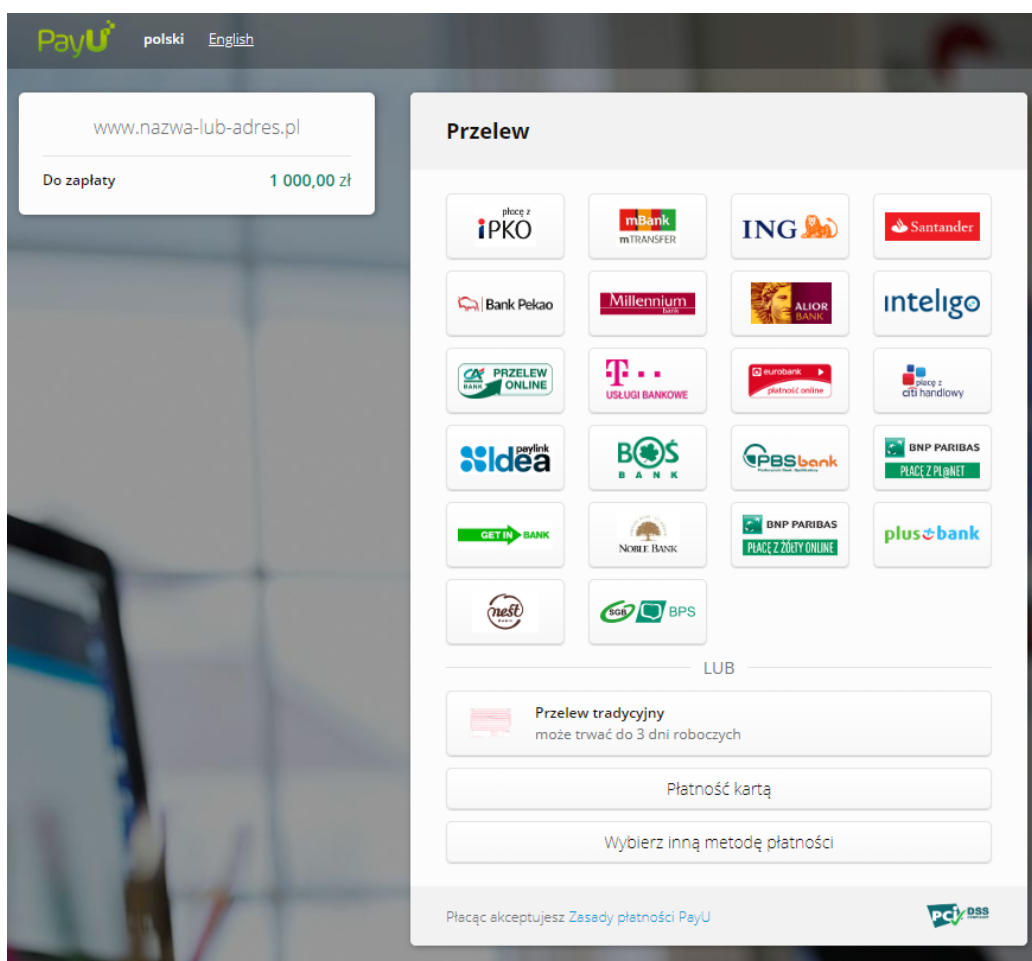
Przelew bankowy jest najprostszym i najbardziej analogowym rozwiązaniem. Zaraz po złożeniu zamówienia, dostajemy potwierdzenie zakupu wiadomością e-mail wraz z numerem konta sprzedawcy, kwotą zamówienia i tytułem przelewu (który najczęściej jest numerem zamówienia). Na kupującym ciąży zalogowanie się na stronę banku, utworzenie nowego przelewu i co istotne wypełnienie danymi. Oprócz możliwości pomyłki, na

przykład przekopujemy niepełny numer zamówienia, pomijając jeden znak, ryzykiem, o którym musimy pamiętać jest obecność złośliwego oprogramowania na naszym urządzeniu. Przykładem takiego oprogramowania może być głośna sprawa z 2018 roku, w którym to firma *ESET* natrafiła na konia trojańskiego *BackSwap*. Portal www.zaufanatrzeciastrona.pl opisuje to oprogramowanie 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 na przykład ING Bank Śląski, po skopiowaniu numeru rachunku bankowego usuwają dwie pierwsze cyfry i proszą klienta o uzupełnienie ich. Bank próbuje wymusić podwójne sprawdzenie numeru konta. Efektem ubocznym będzie możliwość pomyłki przy ponownym wprowadzaniu numeru.

Szybkość przelewu jest zależna od godziny wysłania przelewu. Jest to spowodowane godzinami sesji rozliczeniowych w danych bankach. Najczęściej, jeśli przelew zostanie wysłany do południa, dotrze on popołudniu. W przypadku późniejszych przelewów, dotrze do odbiorcy kolejnego dnia roboczego. Wszystkie przelewy wysyłane w święta i weekendy są dostarczone 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

Przelewy pay-by-link to forma płatności, która cieszy się największą popularnością w Polsce [10]. Po wykonaniu zakupów, wybieramy opcję przelewu online, a następnie z listy banków, pokazanych na rysunku 2.2 wybieramy ten z którego chcemy dokonać płatności.



Rys. 2.2: Przykładowa płatność pay-by-link

Po kliknięciu na guzik z logiem banku zostajemy przeniesieni na stronę

logowania do danego banku. Po wpisaniu loginu i hasła, pojawi się uzupełniony szablon przelewu, który kupujący musi tylko zatwierdzić. Ostatnim krokiem, kluczowym od strony bezpieczeństwa jest SMS z kodem autoryzacyjnym który otrzymamy. Klient musi przepisać owy kod, aby potwierdzić realizację przelewu. Dzięki temu nawet, jeśli ktoś pozna nasze dane logowania, nie będzie mógł ich użyć do płatności bez posiadania naszego telefonu. Alternatywnym i coraz bardziej popularnym rozwiązaniem jest potwierdzenie przelewu przez aplikację banku. Zanim jednak będzie to możliwe, użytkownik musi się do niej zalogować, co oznacza dodatkowy etap bezpieczeństwa.

Po poprawnym uwierzytelnieniu i potwierdzeniu transakcji, środki trafiają niemal natychmiast do sprzedawcy, dzięki czemu od razu będzie on w stanie zrealizować dane zamówienie.

2.2.3 Karta płatnicza

W maju 1998 roku 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 usprawnień, 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 poprawnym 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 płatności zakładał nasze zaufanie do portalu, w którym dokonujemy zakupów, ponieważ osoba posiadająca dane karty mogła

samodzielnie dokonać płatności bez naszej zgody. Odpowiedzią na to jest obecna coraz częściej 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 służy tylko do płatności internetowych. Nie użyjemy jej na przykład podczas zakupów w sklepie stacjonarnym. Po jej wyrobieniu, bank nie przesyła nam jej pocztą, a jedynie podaje nam ich dane, czyli numer, datę 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 proponują utworzenie *kart wirtualnych*, 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 por-

tal, czyli na przykład jedna karta do *Netflix*a, kolejna do opłacenia *Spotify* itd. Dodatkowo można 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 funkcjami:

- portfel wirtualny (rozumiany jako pula środków do wykorzystania) wraz z możliwością przelania ich bezpośrednio na rachunek bankowy,
- płatność wieloma podpętymi kartami (kredytowymi i debetowymi), także w innych walutach,
- możliwość określenia adresu dostawy klienta i innych informacji ułatwiających sfinalizowanie zamówienia.

Kolejnym udogodnieniem płatności jest przechowywanie środków w różnych walutach, dzięki czemu użytkownik nie jest obciążony podwójnym przewalutowaniem.

Niektóre portfele oferują też świadczenie usług bez jakiegokolwiek "przełanych" 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 pobrana natychmiast z naszej wybranej karty. Dzięki temu nie jesteśmy narażeni na potencjalne wady używania samej karty jako środka płatności, opisanych w sekcji 2.2.3.

2.3 Analiza istniejących płatności i portali dla korepetytorów

Przez lata rozwiązania na rynku usług jak i płatności ewoluowały, często stając się osobnym bytem, a nie dodatkiem do portalu, jak na początku ich istnienia. Firmy odpowiedzialne za nie, przez lata starały się udoskonalić swój produkt i jak najlepiej go wypromować. Poniższa analiza rozwiązań została podzielona na dwie grupy: płatności (jako rdzeń serwisu usług) i serwisów dla korepetytorów.

2.3.1 Płatności

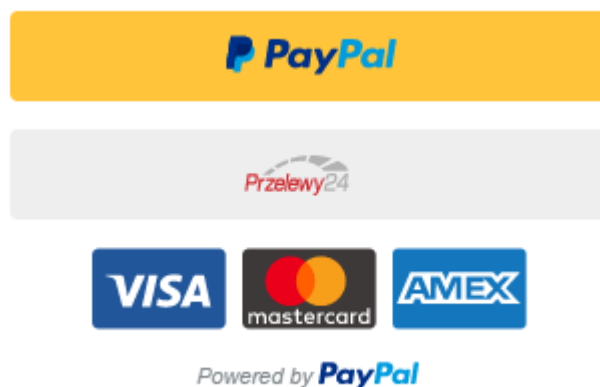
Szukając odpowiedniego rozwiązania do obsługi płatności, zdecydowano mieć na uwadze trzy główne cechy: szybkość (jak szybko środki trafiają na konto), wygodę (a w tym szybkość wykonania samej płatności) i bezpieczeństwo (transakcji i procesów około-transakcyjnych). Każde z czterech dostępnych rozwiązań zostało szczegółowo opisane w podrozdziale 2.2, właśnie pod kątem tych trzech cech. Stosując takie kryterium, można uznać portfel elektroniczny jako przodownika w tych trzech kategoriach. Dlatego w dalszym rozważaniu jest on tylko brany pod uwagę.

Tab. 2.1: Porównanie portfeli elektronicznych

	PayPal	Visa Checkout	Masterpass
Język polski	✓	✓	✓
Przypisanie adresu dostawy do konta	✓	✓	✓
Uwierzytelnianie wiadomości e-mail	✓	✓	✗
Używanie kart różnych firm	✓	✗	✗
Możliwość dwustopniowego uwierzytelniania	✓	✗	✓
Możliwość przechowywania środków w portfelu	✓	✗	✗

Zestawienie pokazane w tabeli 2.1 pokazuje, że PayPal wygrywa pod względem dwóch funkcji z Visa Checkout i Masterpass – kryterium użycia 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, użytkownik zostaje przeniesiony do realizacji płatności. PayPal zdecydował się na inne rozwiązanie, które daje klientowi możliwość zapłaty kartą (Visa, MasterCard i Amex), polskim portalem Przelewy24 (agreguje płatności pay-by-link i kartą omówione w podrozdziale 2.2) i portfelem cyfrowym PayPal.



Rys. 2.3: Smart Payment Button

PayPal określa swój przycisk jako **Smart Payment Button** (inteligentny przycisk do płatności), przedstawiony na rysunku 2.3. Oprócz łączenia kilku sposobów płatności, może różnić się on w każdym kraju. Na przykład w Polsce wspiera on portal Przelewy24. Dzięki elastyczności *smart payment button* można być pewnym, 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 funkcjonalności. Dlatego też, to ono zostało zaimplementowane w aplikacji *Find-A-Tutor* w celu wspierania płatności on-line.

2.3.2 Serwisy dla korepetytorów

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

wspomniany już wcześniej portal, nie tylko do korepetycji www.olx.pl. W tabeli 2.2 zestawiono ze sobą cechy tych portali. Cechy zostały pobrane z podrozdziału 1.1 pokrywającego problematykę pracy.

Tab. 2.2: Porównanie serwisów dla korepetytorów

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

Żaden z portali zestawionych powyżej nie obsługuje funkcji 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. Aplikacja serwerowa została stworzona jako otwarte API, dzięki czemu w przyszłości serwis będzie otwarty na powstanie aplikacji mobilnych. Fundamentalną funkcją będzie dodawanie ogłoszeń przez studentów. Ponadto aplikacja kliencka została zrealizowana z minimalistycznym i nieprzytłaczającym designem.

3 Techniczne aspekty aplikacji *Find-A-Tutor*

Stworzony projekt składa się z dwóch warstw: aplikacji po stronie serwera (backend) i aplikacji po stronie klienta (frontend). Frontend został napisany w technologii ASP.NET Razor (generujące dynamiczne strony internetowe po stronie serwera) i fundamentalnych technologiach 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 najnowszej obecnie wersji aplikacji szkieletowej .NET Core 2.2.

Funkcje rejestracji i logowania do konta zostały napisane przy wykorzystaniu JWT 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.

3.1 Wymagania

W poniższym podrozdziale opisano wymagania stworzonej aplikacji webowej *Find-A-Tutor*, dzieląc je na funkcjonalne i niefunkcjonalne.

3.1.1 Wymagania funkcjonalne

Wymagania funkcjonalne to założenia według, których aplikacja ma się zachować w określony sposób po wystąpieniu danego zdarzenia/zapytania. Wymagania te jasno określają funkcje, jakie ma mieć aplikacja, problemy

jakie ma ona rozwiązywać, a także opisują jej ogólną wizję [12].

Rejestracja jako uczeń lub korepetytor

Formularz rejestracji umożliwia rejestrację jako użytkownik z rolą ucznia i nauczyciela. Obie te role posiadają odrębne uprawnienia i funkcje opisane poniżej.

Logowanie jako uczeń, korepetytor i administrator

Dostępne są dwa odrębne panele logowania dla użytkownika (ucznia lub korepetytora). Dodatkowym typem konta jest konto administratora, które 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 ucznia jest dostępny dla użytkownika zalogowanego jako student. Użytkownik ma dostęp do listy wszystkich dodanych przez siebie ogłoszeń. W panelu widnieją następujące dane: data utworzenia ogłoszenia, data podjęcia ogłoszenia, opis ogłoszenia, data wygaśnięcia ogłoszenia, nazwa przedmiotu, status płatności i status ogłoszenia.

Podgląd szczegółów ogłoszenia

Panel ucznia umożliwia widok szczegółowy ogłoszenia, który pokazuje informacje dodatkowe takie jak: czas trwania lekcji, cenę za godzinę i całkowitą kwotę do zapłaty.

Dodawanie ogłoszeń

Ogłoszenia mogą być dodawane przez ucznia. Użytkownik wypełnia formularz zawierający następujące dane: opis ogłoszenia, data przedawnienia ogłoszenia oraz przedmiot (wybrany z listy przedmiotów).

Płatności online za lekcje

Po podjęciu ogłoszenia przez korepetytora uczeń ma możliwość opłacenia online należności za ogłoszenie. Po opłaceniu następuje zmiana statusu płatności.

Finalizacja ogłoszenia

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

Panel korepetytora

Użytkownik z rolą nauczyciela ma dostęp do panelu korepetytora. W panelu widnieją wszystkie dostępne ogłoszenia i wszystkie zrealizowane ogłoszenia przez zalogowanego nauczyciela. Panel korepetytora zawiera następujące informacje: data utworzenia ogłoszenia, data podjęcia ogłoszenia, opis ogłoszenia, data wygaśnięcia ogłoszenia, nazwa przedmiotu, status płatności i ogłoszenia.

Podjęcie ogłoszenia

Korepetytor ma możliwość podjęcia ogłoszenia z puli dostępnych ogłoszeń. Użytkownik zalogowany jako nauczyciel wprowadza 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, ogłoszenie zmienia status na "podjęte".

Oddanie ogłoszenia

Jeśli ogłoszenie, po podjęciu, nie zostało jeszcze opłacone, korepetytor ma możliwość zwrócenia ogłoszenia do ogólnej puli ogłoszeń, tym samym status zmienia się na "niepodjęte".

Konto administratora

Po zalogowaniu się z rolą administratora, użytkownik może wykonywać operacje CRUD (ang. *create, read, update and delete*) na ogłoszeniach, użytkownikach i liście przedmiotów. Ma możliwość tworzenia, odczytywania, aktualizowania i usuwania rekordów. Użytkownik zalogowany jako administrator może wykonywać wszystkie operacje dostępne dla 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. Do tych wymagań należą skalowalność, kompatybilność, bezpieczeństwo [13].

Przenośność

Aplikacja powinna być dostosowana do zmiany lokalizacji, środowiska, eksportu danych.

Skalowalność

Aplikacja powinna mieć skalowalność pionowo (zwiększenie mocy serwera) i poziomo (zwiększenie instancji aplikacji).

Otwartość na rozszerzenia

Aplikacja powinna mieć otwarte API (aplikacje po stronie serwera) i być gotowa na nowe urządzenia, które mogą z niej korzystać (urządzenia mobilne).

Kompatybilność

Aplikacja powinna działać w technologiach uniwersalnych i powszechnych

dla przeglądarek na silniku Chromium (Chrome, Opera i wkrótce Edge), który na rynku polskim ma 80,65% udziału w rynku [14].

Bezpieczeństwo

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

Użyteczność

Aplikacja powinna być łatwa w obsłudze i powinna zostać zrealizowana z minimalistycznym designem, który nie przytłacza użytkownika.

Modularność

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

3.2 Narzędzia i technologie użyte w projekcie

Do realizacji aplikacji zostały wybrane tylko te technologie i rozwiązania, które są w pełni darmowe do użytkowania, nawet w projektach komercyjnych, oferują stabilne wersje, które zostały wielokrotnie sprawdzone w setkach innych projektach webowych i są ciągle wspierane i rozwijane do najnowszych standardów.

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 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, dostosowany do pracy w chmurze,
- wbudowane wstrzykiwanie zależności,
- lekki, wysoko wydajny modularny potok żądań HTTP,
- publicznie dostępny kod i możliwość kontrybucji
- obsługa kilku języków programowania: C# , F# i także Visual Basic,
- obsługa menadżera pakietów *NuGet*,
- elastyczny w wdrożeniu, na przykład poprzez kontereryzacje za pomocą Dockera.

3.2.2 C#

Język C# jest językiem obiektowym, opracowanym dla firmy Microsoft, na przełomie tysiąclecia przez zespół pod kierownictwem Andersa Hejlsberga. Jak można przeczytać w oficjalnej dokumentacji języka: "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 języka, zespołowi przyświecały następujące założenia projektowe:

- 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* czyli punkt wejścia aplikacji, są zawarte w definicjach klas. Klasa może dziedziczyć bezpośrednio z 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, na temat tego jak dobrze program dzi-

iała lub czy działa zgodnie z jego założeniami. Skuteczna kontrola stanu (ang. *health check*) zapewnia dostęp do informacji o stanie zależności projektu. Jest to na przykład sprawdzenie czy aplikacja ma połączenie z bazą lub pokazanie w jakiej "kondycji" jest nasz program, na przykład jaki czas zajmuje mu połączenie i wywołanie prostego zapytania do bazy danych.

Rozwiązaniem, które dostarcza kontrolę 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 *Find-A-Tutor (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-*

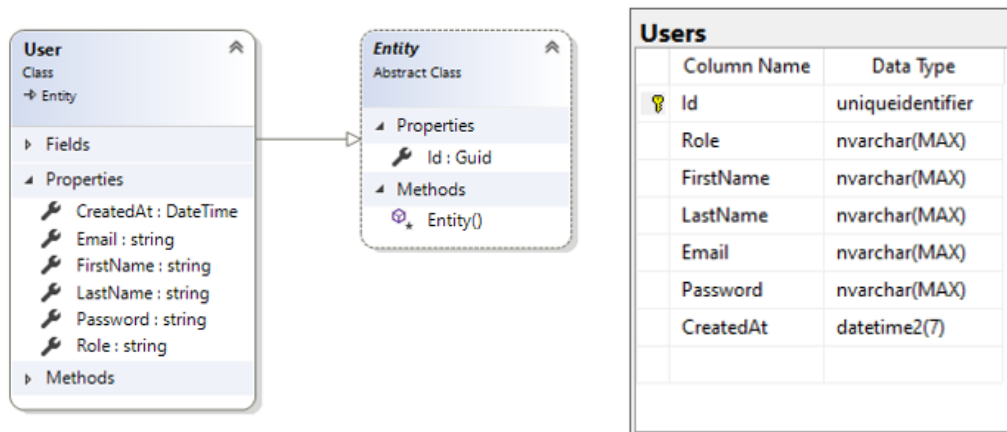
Relational Mapping ORM). Umożliwia on programistom pracę z bazą danych przy użyciu obiektów .NET. Dodatkowo, w większości eliminuje potrzebę pisania kodu do połączenia się z bazą danych.

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 Core obsługuje trzy sposoby realizacji mapowania obiektowo-relacyjnego:

- *Code First*: programista określa model obiektów przez kod. Następnie, EF generuje odpowiedni bazodanowy model na podstawie kodu i ustawień konfiguracyjnych dostarczonych przez programistę.
- *Database First*: tworzenie modelu obiektów z istniejącej bazy danych.
- *Model First*: do stworzenia bazy danych z istniejącego modelu używa się eksportu modelu stworzonego przez *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 *Find-A-Tutor* został zrealizowany według pierwszego podejścia, czyli *Code First*. Przygotowano domenę, którą następnie za pomocą migracji została odwzorowana na nowo utworzonej bazie danych.

3.2.5 JWT

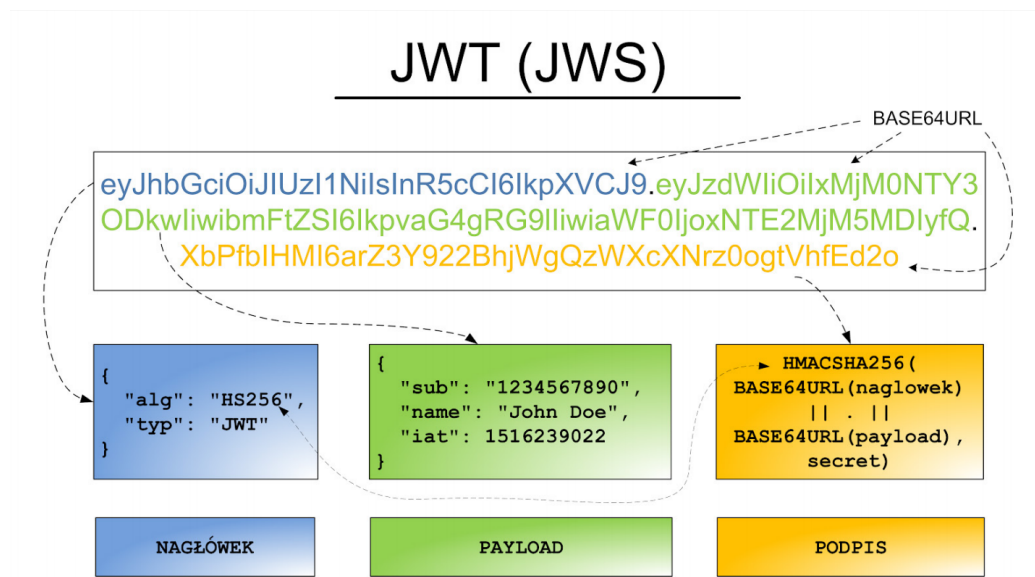
JSON Web Token (w skrócie JWT) to otwarty standard (RFC 7519), który odpowiada za wymianę danych pomiędzy stronami (klient-serwer) w bezpieczny sposób poprzez obiekt w formacie JSON. Informacje przekazywane w nim mogą zostać zweryfikowane przy użyciu cyfrowego podpisu, który jest zawarty w tokenie. Podpis jest generowany za pomocą algorytmu HMAC (*Hash-based Message Authentication Code* czyli kod uwierzytelnienia wiadomości oparty na skrócie) lub klucza publicznego/prywatnego RSA (Rivesta-Shamira-Adlemana) lub ECDSA (*Elliptic Curve Digital Signature Algorithm* czyli algorytm elektronicznego podpisu poprzez kryptografia krzywych elip-

tycznych).

JWT może zostać wykorzystany do:

- 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*) oraz podpis (ang. *signature*). Podział ten jest pokazany na rysunku 3.3.



Rys. 3.3: Przykład podstawowej struktury JWT [21]

Nagłówek w standardzie JWT


```

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

Rys. 3.4: Przykład nagłówka tokenu dla standardu JWT

Nagłówek (pokazany na rysunku 3.4) zawiera dwie informacje: rodzaj tokenu na przykład JWT oraz jaki algorytm został użyty do wygenerowania podpisu, na przykład HMAC-SHA256 (zapisywany jako HS256). W postaci końcowej (token), nagłówek jest zapisywany za pomocą algorytmu BASE64URL.

Zawartość w standardzie JWT

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

Zawartość również jest kodowana za pomocą algorytmu BASE64URL.

```

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

Rys. 3.5: Przykład zawartości tokenu dla standardu JWT

Podpis w standardzie JWT

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

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

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

Rys. 3.6: Sposób obliczania podpisu w standardzie JWT

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



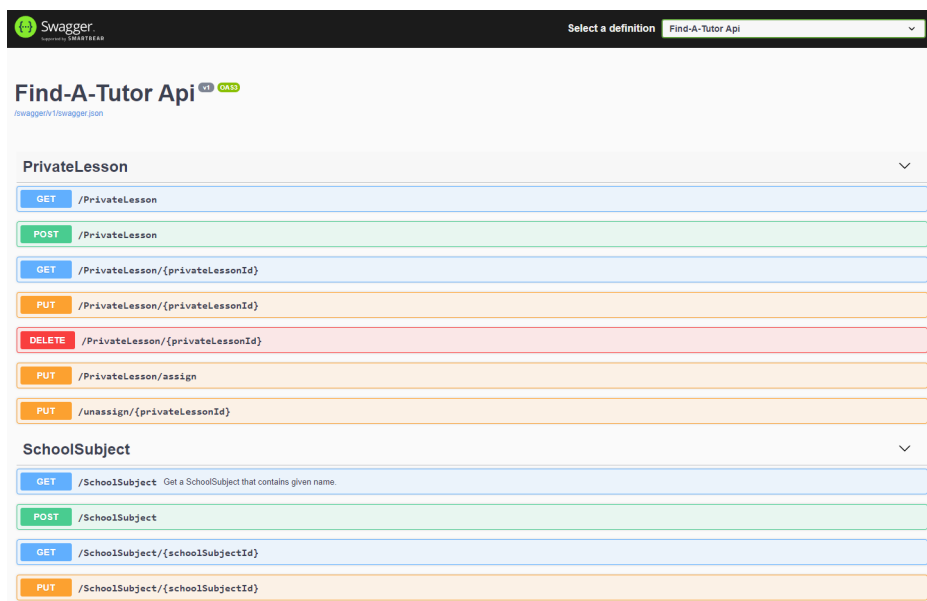
Rys. 3.7: Działanie tokenu w praktyce [22]

Rysunek 3.7 pokazuje proces logowania i wysłania zapytania do API:

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 opisanym w tej sekcji. 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).

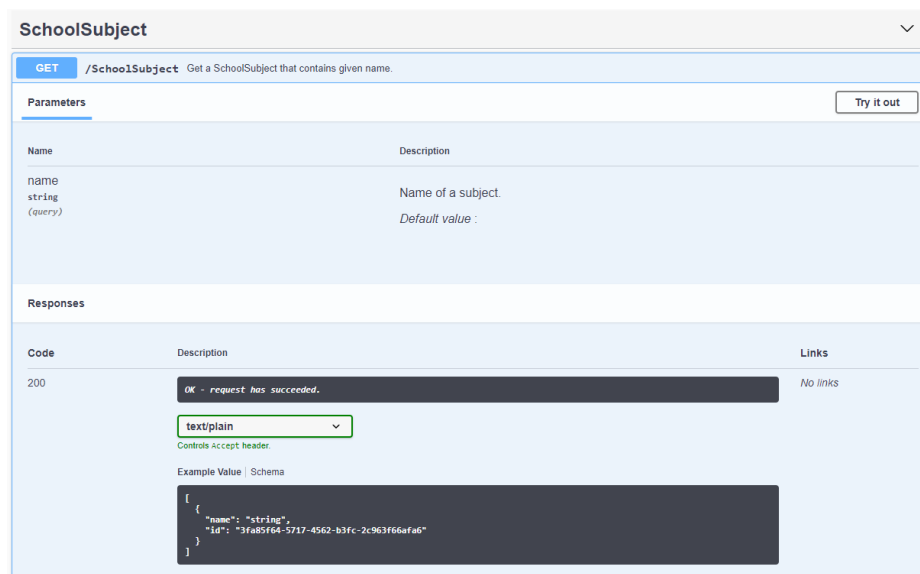
3.2.6 Swagger UI

Swagger UI 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 on 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

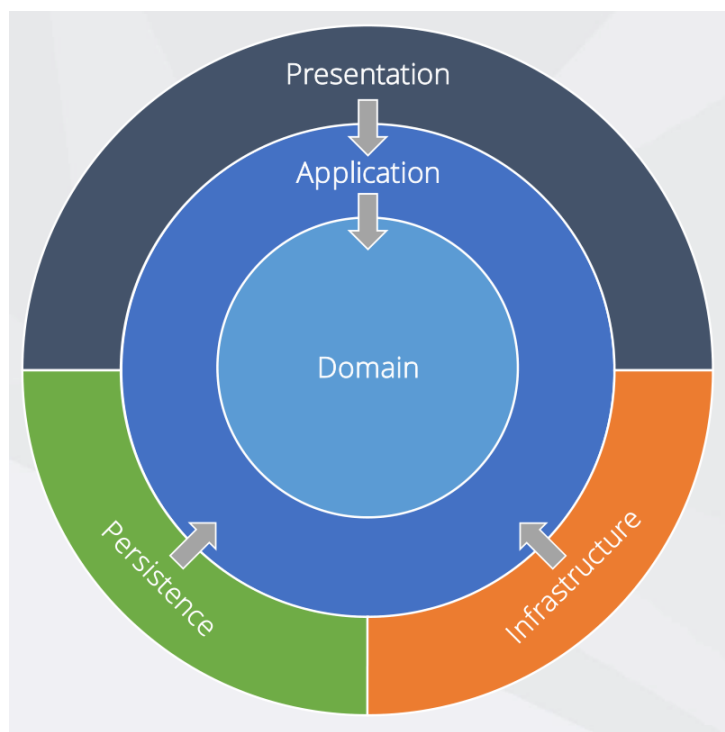
Panel Swagger UI jest dostępny pod adresem /swagger.

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*), następnie *Porty i Adaptery* (ang. *Ports-and-Adapters*). Coraz częściej nazywa się ją *Architekturą Cebulową* (ang. *Onion Architecture*) lub *Czystą Architekturą* (ang. *Clean Architecture*). W niniejszej pracy używa się właśnie tej drugiej nazwy.

Czysta Architektura stawia logikę biznesową i model aplikacji w jej środku. 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. Każda warstwa jest zależna od warstwy bliżej centrum – aż do domeny, która nie ma żadnych zależności. Warstwa aplikacji zależy od warstwy domeny, a pozostałe trzy (prezentacji, infrastruktury i utrwalenia) zależą od warstwy aplikacji. Należy podkreślić, że poprawnie wymodelowana domena aplikacji 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 takiemu ułożeniu warstw.



Rys. 3.10: Czysta Architektura [15].

W projekcie zastosowano właśnie takie rozwiązanie, ze względu na czystość, prostotę implementacji, łatwość zmian (na przykład wymiany infrastruktury na zupełnie nową), a przede wszystkim na czytelność [15].

3.3.1 Rdzeń aplikacji

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

Zgodnie z *PI* (ang. *Persistence Ignorance*) i *Zasadą Ignorowania Infrac-*

truktury (ang. *Infrastructure Ignorance*) *rdzeń* musi całkowicie ignorować szczegóły dotyczące trwałości danych (na przykład zapis i odczyt danych). Te zadania powinny zostać wykonane przez warstwę infrastruktury. Dlatego *rdzeń* nie powinien przyjmować bezpośrednich zależności od infrastruktury, co oznacza że ważne jest aby klasy reprezentujące encje modelowe były obiektami **POCO** (ang. *Plain Old CLR Object*) czyli *zwykłymi starymi obiektami CLR*, gdzie *CLR* to środowisko uruchomieniowe *.Net*. Termin **POCO** oznacza prosty obiekt, niezawierający logiki biznesowej i nieobciążony dziedziczeniem.

Domenowe encje nie powinny mieć żadnej bezpośredniej zależności (jak dziedziczenie) od frameworka, służącego do dostępu danych, jak na przykład *Entity Framework* (technologia szczegółowo opisana w sekcji 3.2.3).

3.3.2 Infrastruktura aplikacji

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 jest *Entity Framework Core*. Napisany kod służy do połączania się z bazą danych przez *DbContext*, który jest reprezentacją bazy danych jako obiektu. Dzięki temu w łatwy sposób można operować na danych w bazie danych.

Ta warstwa zależy od warstw leżących poniżej czyli warstw aplikacji i domeny.

3.3.3 API aplikacji

Warstwą odpowiedzialną za kontakt z otoczeniem aplikacji jest warstwa prezentacji. W projekcie "backendowym", została zaimplementowana 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 zachowuje żadnych informacji o poprzednich transakcjach z klientem.

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

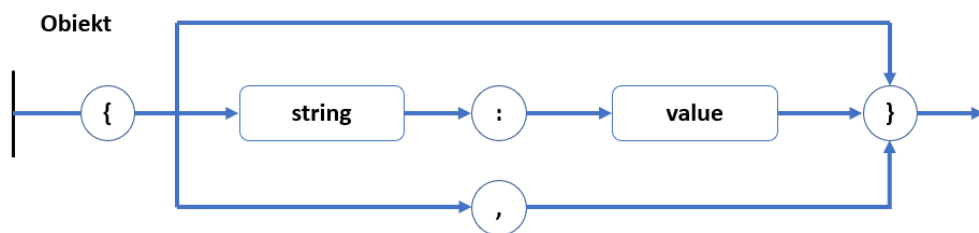
Gdyby użytkownik skorzystał z metody PUT, mógłby zaktualizować dane znajdujące się pod tym endpointem. Wszystkie metody są opisane przez dokument **RFC2616** [23].

REST (ang. *Representational State Transfer*) jest wzorcem architektonicznym, 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 (JSON) to otwarty i lekki format wymiany danych. Charakteryzuje się tekstem czytelnym dla człowieka. Używany jest do przesyłania obiektów danych, składających się z par "atribut-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. Wywodzi się on z języka JavaScript, ale prawie wszystkie nowoczesne języki programowania zawierają funkcję generowania i parsowania danych sformatowanych za pomocą JSONa. Dlatego też stał się uniwersalnym formatem do deserializacji danych, przesyłania ich do/z WebAPI, jak i przechowywaniu ich na dysku.

W aplikacji *Find-A-Tutor* stał się on formatem wymiany danych – WebAPI, przyjmuje i zwraca dane w formacie JSON.

```

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 – zwracane są wszystkie ogłoszenia danego użytkownika (dodane z jego konta).

3.4 Architektura aplikacji – frontend

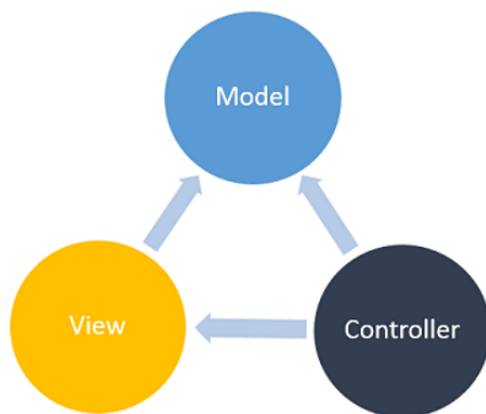
Najmniejsza możliwa liczba projektów dla architektury aplikacji to jeden. W takiej aplikacji, cała logika jest zawarta w jednym projekcie, kompilowana jest do jednego pliku wykonywalnego i wdrożona jako jedna całość.

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 lub z bazą danych.

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

3.4.1 MVC

Model-view-controller (*Model-Widok-Kontroler*) to wzorzec architektoniczny powszechnie używany do opracowania aplikacji, posiadających graficzny interfejs użytkownika (Rys. 3.13).



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

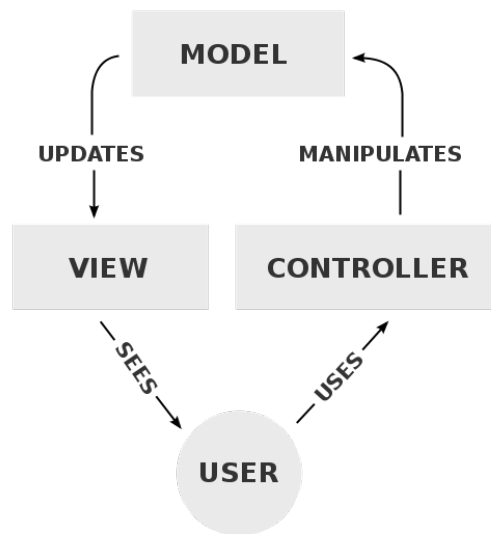
MVC dzieli aplikację na trzy części:

- Model – logika biznesowa.
- Widok – graficzny interfejs dla użytkownika.
- Kontroler – obsługuje żądania użytkownika i konwertuje je na polecenia widoku lub modelu.

Wszystkie części wzorca MVC są ze sobą połączone (Eysunek 3.13). Aplikacja frontendowa, projektu *Find-A-Tutor*, została zaprojektowana przy użyciu właśnie tej architektury. Prześledźmy zatem pojedynczą akcję użytkownika, na przykład logowanie:

1. Użytkownikowi zostaje zwrócony odpowiedni UI (*Widok*).
2. Użytkownik wpisuje w odpowiednie pola login i hasło .
3. Aplikacja obsługuje te dane i przekazuje je dalej (*Kontroler*).
4. Aplikacja opakuje te dane (JSON) i wysyła jako zapytanie HTTP do WebAPI (*Model*). WebAPI otrzymuje dane i je przetwarza.
5. WebAPI zwraca dane do aplikacji –są one rozpakowywane i przekazywane dalej (*Model*).
6. Model aktualizuje Widok, co jest widoczne dla użytkownika.

Przepływ danych między częściami wzorca *MVC* został pokazany na rysunku 3.14.



Rys. 3.14: Zasada organizacji wzorca projektowego MVC [25]

Przedstawiony scenariusz pokazuje że "frontend" został napisany jako aplikacji klienckiej który obsługuje żądania użytkownika i usprawnia (poprzez UI) komunikację użytkownika z serwerem. Rozdzielenie tych dwóch aplikacji, czyli serwer(*backend*) i klient (*frontend*), ma następujące zalety:

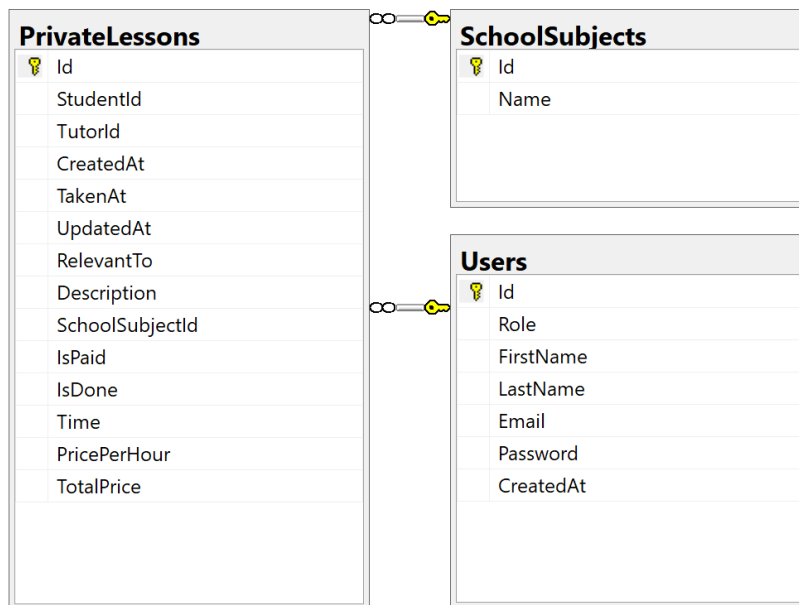
- *Modularność* - w razie potrzeby jesteśmy w stanie wymienić jedną z aplikacji na nową.
- W razie potrzeby zmiany szaty graficznej aplikacji można w łatwy sposób napisać sam "frontend" od nowa. Dzięki temu cała aplikacja zostaje niezmieniona.
- Ułatwienie w utrzymaniu aplikacji i pracy nad projektem.
- Możliwość pracy w dwóch osobnych zespołach nad dwoma projektami.

3.5 Warstwa bazy danych

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

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

Rysunek 3.15 przedstawia strukturę bazy danych dla aplikacji *Find-A-Tutor*:

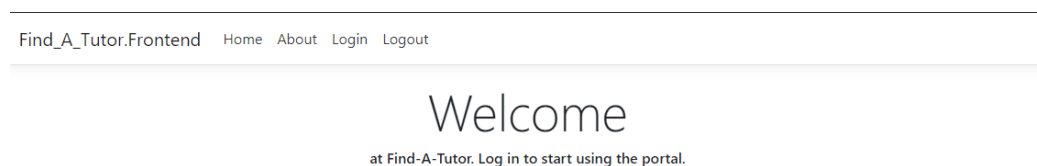


Rys. 3.15: Struktura bazy danych dla aplikacji *Find-A-Tutor*

Podczas powstawania aplikacji Find-A-Tutor starano się aby była ona od samego początku dobrze zaprojektowana tak, by później wszelkie zmiany nie komplikowały nadmiernie stworzonego projektu.

4 Aplikacja z punktu widzenia użytkownika

W poniższym rozdziale pokazano działanie aplikacji webowej *Find-A-Tutor*. Zaprezentowane widoki aplikacji zostały wykonane w przeglądarce Chrome w wersji 79. Na rysunku 4.1 pokazano ekran startowy aplikacji.

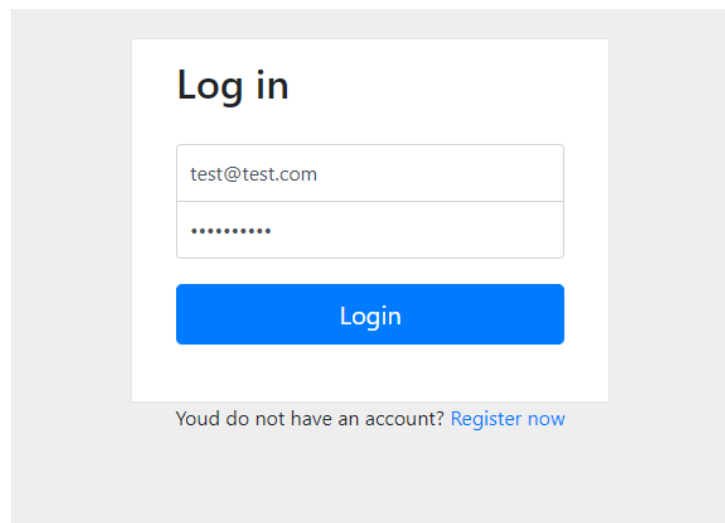


Rys. 4.1: Strona startowa aplikacji *Find-A-Tutor*

4.1 Uwierzytelnianie

Po kliknięciu na "Login", użytkownikowi ukazuje się panel logowania (Rys. 4.2).

Po uzupełnieniu pól (e-mail i hasło) i wciśnięciu przycisku "Login", dane są wysyłane do API. Jeśli uwierzytelnienie się powiedzie, to użytkownik jest przekierowywany do panelu użytkownika.



Rys. 4.2: Panel logowania do panelu użytkownika

Jeśli użytkownik nie ma założonego konta, może je założyć za pomocą linku "Register now", który przekierowuje go do podstrony z rejestracją, pokazaną na rysunku 4.3.

Użytkownik ma do wypełnienia następujące pola: imię, nazwisko, adres email, hasło i typ konta (uczeń lub korepetytor). Poprawność danych sprawdzana jest w locie, dzięki zastosowanym standardom (HTML 5) i po stronie serwera po kliknięciu przycisku "Register".

4.2 Panel użytkownika

W tej sekcji pokazano panel użytkownika i opisano akcje które użytkownik może wykonać za jego pomocą.

Register

First name

Last name

E-Mail

Password

Are you a tutor? ☒

[Register](#)

Rys. 4.3: Formularz rejestracji

4.2.1 Dodanie nowego ogłoszenia

Po pomyślnym uwierzytelnieniu użytkownika w roli ucznia, wyświetlany jest panel jego użytkownika (Rys. 4.4).

Find_A_Tutor.Frontend Home About Login Logout

Student panel

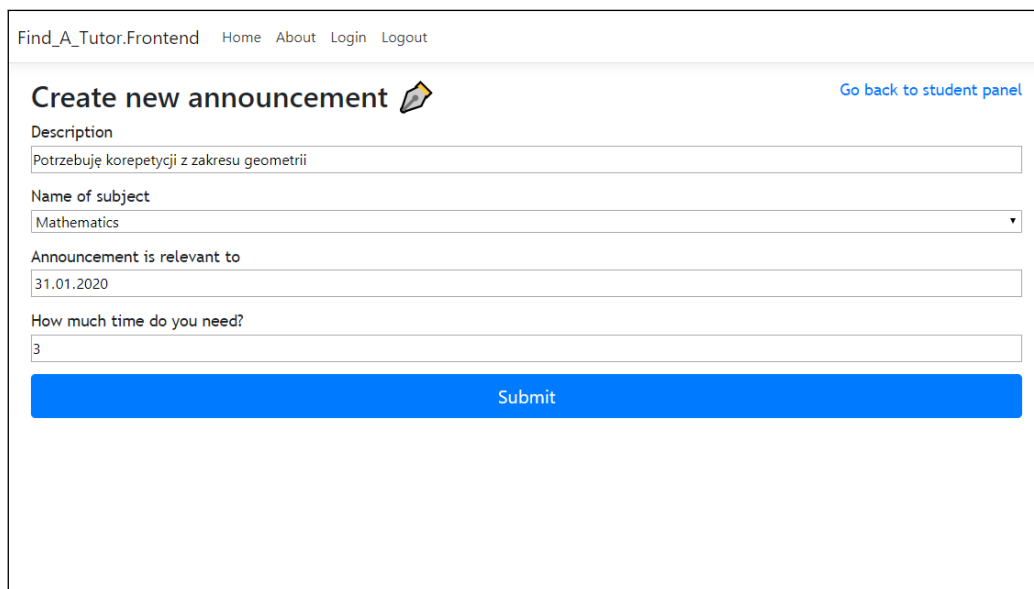
No announcements

[Create new announcement](#)

Rys. 4.4: Pusty panel ucznia w aplikacji *Find-A-Tutor*

Panel ten jest pusty. Aby dodać ogłoszenie, klikamy na link "Create new announcement" i zostajemy przekierowani do panelu dodawania nowego

ogłoszenia, pokazanego na rysunku 4.5.



The screenshot shows a web form titled "Create new announcement" with a pencil icon. At the top right is a link "Go back to student panel". The form contains the following fields:

- Description:** A text input field containing "Potrzebuję korepetycji z zakresu geometrii".
- Name of subject:** A dropdown menu with "Mathematics" selected.
- Announcement is relevant to:** A date input field containing "31.01.2020".
- How much time do you need?:** A text input field containing "3".


At the bottom of the form is a large blue button labeled "Submit".

Rys. 4.5: Formularz do dodania nowego ogłoszenia w aplikacji *Find-A-Tutor*

Użytkownik wprowadza tam następujące informacje: opis, datę ważności ogłoszenia i długość lekcji. Ponadto wybiera przedmiot nauczania z rozwijanej listy. Kiedy dane są wprowadzone, użytkownik klika przycisk "Submit". Następuje wysłanie danych do API. Jeśli wszystko zostanie poprawnie zwalidowane (na przykład data ważności nie jest datą historyczną), użytkownikowi pokaże się komunikat widoczny na rysunku 4.6.

Po powrocie do panelu użytkownika, widoczne jest w nim dodane ogłoszenie (Rys. 4.7), wraz z odpowiednimi informacjami o nim: datę utworzenia, datę podjęcia (na razie pusta), datę ważności, przedmiot nauczania, trzy flagi stanu (podjęte, zapłacone i wykonane) i link do podstrony z wszystkimi szczegółami ogłoszenia.

Find_A_Tutor.Frontend [Home](#) [About](#) [Login](#) [Logout](#)

Create new announcement 

[Go back to student panel](#)

Description

Opis

Name of subject


Mathematics

Announcement is relevant to

dd.mm.rrrr


How much time do you need?


Submit

Announcement was added 

Rys. 4.6: Komunikat po dodaniu nowego ogłoszenia przez ucznia w aplikacji *Find-A-Tutor*

Find_A_Tutor.Frontend [Home](#) [About](#) [Login](#) [Logout](#)

Student panel 

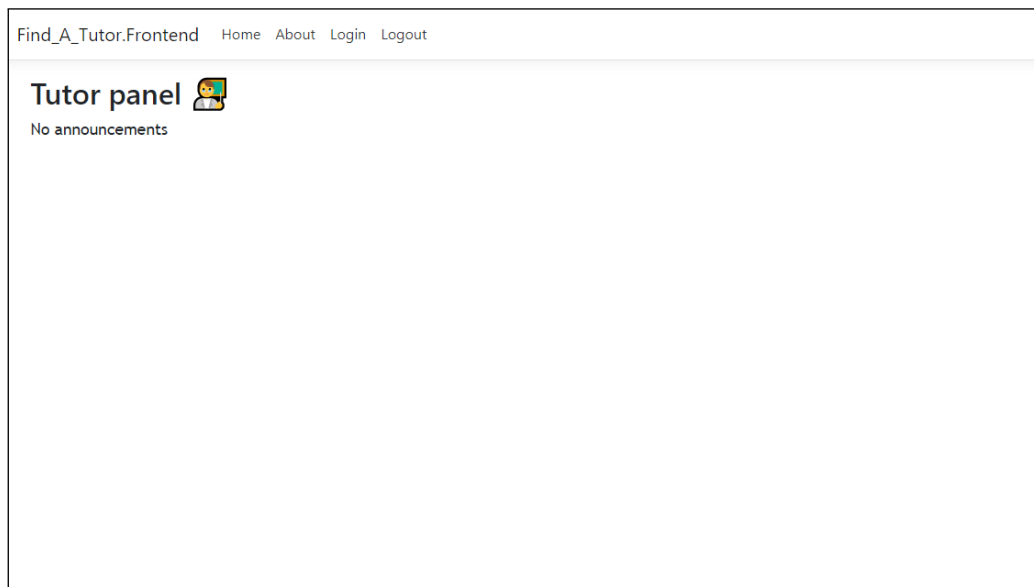
[Create new announcement](#) 

CreatedAt	TakenAt	RelevantTo	Description	Subject	Assigned?	Paid?	Completed?	Details
04.01.2020		31.01.2020	Potrzebuję korepetycji z zakresu geometrii	Mathematics	✗	✗	✗	Details

Rys. 4.7: Panel ucznia po dodaniu nowego ogłoszenia w aplikacji *Find-A-Tutor*

4.2.2 Podjęcie ogłoszenia

Po pomyślnym uwierzytelnieniu użytkownika w roli nauczyciela, jest wyświetlany panel nauczyciela (Rys. 4.8).



Rys. 4.8: Pusty panel nauczyciela w aplikacji *Find-A-Tutor*

Jeśli którykolwiek z uczniów dodał nowe ogłoszenie, to jest ono tutaj wyświetlane (Rys. 4.9). Użytkownik może wykonać dwie akcje na ogłosze-

The screenshot shows the same web application interface as before, but now with a table of announcements. The table has columns for 'CreatedAt', 'TakenAt', 'RelevantTo', 'Description', 'Subject', 'Assigned?', 'Paid?', 'Completed?', and 'Take Give'. There are two rows of data.

CreatedAt	TakenAt	RelevantTo	Description	Subject	Assigned?	Paid?	Completed?	Take	Give
04.01.2020	04.01.2020	31.01.2020	Potrzebuję korepetycji z zakresu geometrii	Mathematics	✗	✗	✗	Take	Give back
04.01.2020		15.02.2020	Potrzebne korepetycje z fizyki, rok I	Physics	✗	✗	✗	Take	Give back

Rys. 4.9: Panel nauczyciela w aplikacji *Find-A-Tutor*

niu: podjąć je lub oddać je do ogólnie dostępnej puli wolnych ogłoszeń.

Po naciśnięciu przycisku "Take", użytkownik zostaje przekierowany do podstrony ze szczegółami ogłoszenia (Rys. 4.10).

The screenshot shows a web form titled "Take an announcement" with a blue square icon. At the top right is a link "Go back to panel". The form contains the following fields and values:


- Created At:** 01/05/2020 12:20:05
- Taken At:** -
- Relevant To:** 15.02.2020
- Description:** Potrzebne korepetycje z fizyki, rok I studiów
- Subject:** Physics
- Assigned?** ☒
- Paid?** ☒
- Completed?** ☒
- Time** 2
- Price per hour** 50


At the bottom is a large blue button labeled "Take!".

Rys. 4.10: Podjęcie ogłoszenia przez nauczyciela w aplikacji *Find-A-Tutor*

W polu "Price per hour" użytkownik wpisuje stawkę godzinową. Po naciśnięciu przyciska "Take", dana ta jest walidowana, i przesłana do API, a ogłoszenie zostaje przypisane do danego konta nauczyciela.

Podjęcie może nastąpić tylko wtedy, kiedy ono jest wolne. Oddanie ogłoszenia następuje tylko wtedy, kiedy nie jest przypisany do niego nauczyciel i ogłoszenie nie zostało jeszcze opłacone.



Po powrocie do panelu użytkownika, widoczne jest w nim podjęte ogłoszenie wraz z oznaczeniem  w polu "Czy przypisane" (Rys. 4.11).

Find_A_Tutor.Frontend Home About Login Logout									
Tutor panel 									
CreatedAt	TakenAt	RelevantTo	Description	Subject	Assigned?	Paid?	Completed?	Take	Give
04.01.2020	04.01.2020	31.01.2020	Potrzebuję korepetycji z zakresu geometrii	Mathematics	✓	✗	✗	Take	Give back
04.01.2020		15.02.2020	Potrzebne korepetycje z fizyki, rok I	Physics	✗	✗	✗	Take	Give back

Rys. 4.11: Panel nauczyciela w aplikacji *Find-A-Tutor*

4.2.3 Płatność

Po ponownym zalogowaniu w roli ucznia widoczna jest zmiana w ogłoszeniu - zostało ono podjęte (Rysunek 4.12).


Find_A_Tutor.Frontend Home About Login Logout									
Student panel 								Create new announcement 	
CreatedAt	TakenAt	RelevantTo	Description	Subject	Assigned?	Paid?	Completed?	Details	
04.01.2020	04.01.2020	31.01.2020	Potrzebuję korepetycji z zakresu geometrii	Mathematics	✓	✗	✗	Details	
04.01.2020		15.02.2020	Potrzebne korepetycje z fizyki, rok I studiów	Physics	✗	✗	✗	Details	

Rys. 4.12: Panel ucznia w aplikacji *Find-A-Tutor*

Link "Details" przekierowuje użytkownika do szczegółów ogłoszenia, wraz z możliwością jego opłacenia (Rys. 4.13).

Kwota "Total price" jest wynikiem mnożenia "Time" i "Price per hour". W celu opłacenia, użytkownik wybiera sposób płatności, na przykład przy użyciu portfela elektronicznego PayPal.

Find_A_Tutor.Frontend Home About Login Logout

Details of announcement 

[Go back to panel](#)


Created At:
01/04/2020 12:02:30


Taken At:
01/04/2020 12:19:01


Relevant To:
31.01.2020

Description:
Potrzebuję korepetycji z zakresu geometrii

Subject:
Mathematics

Assigned?



Paid?



Completed?





Time
3


Price per hour
50

Total price
150

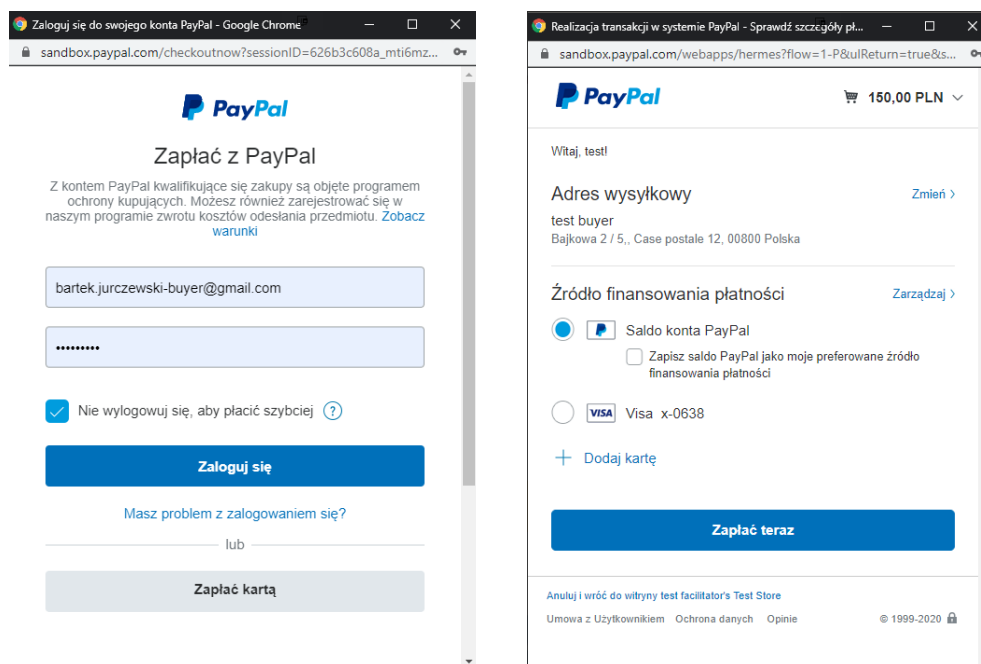






Powered by 

Rys. 4.13: Szczegóły ogłoszenia w aplikacji *Find-A-Tutor* w widoku ucznia



(a) Logowanie do serwisu PayPal (b) Potwierdzenie szczegółów płatności

Rys. 4.14: Płatność przy użyciu PayPal

Dokonanie płatności zostało pokazane na rysunku 4.14.

Po przetworzeniu płatności, status ogłoszenia zostaje zaktualizowany. Został on pokazany na rysunku 4.15.

4.3 Wylogowanie

Po naciśnięciu przycisku "Logout", na każdej z podstron, następuje zamknięcie sesji i wylogowanie z aplikacji.

Aplikacja Find-A-Tutor zapewnia wszystkie potrzebne funkcje dostarczając łatwy w obsłudze i wygodny portal do obsługi korepetycji.

Details of announcement

Created At:

01/04/2020 12:02:30

Taken At:

01/04/2020 12:19:01

Relevant To:

31.01.2020

Description:

Potrzebuję korepetycji z zakresu geometrii

Subject:

Mathematics

Assigned?



Paid?



Completed?



Time

3

Price per hour

50

Total price

150

Rys. 4.15: Szczegóły zapłaconego ogłoszenia w aplikacji *Find-A-Tutor* w widoku ucznia i studenta

5 Podsumowanie

Pierwszym etapem niniejszej pracy dyplomowej była analiza dwóch rynków: usług i płatności internetowych. Analiza rynku usług dowiodła, że dostępne rozwiązania, takie jak `www.olx.pl`, oferują tylko podstawowe funkcje. Przede wszystkim brakuje w nich płatności cyfrowych (w jakiegolwiek formie). Dodatkowo, pomogła ona dostrzec niszę – rynek korepetycji, który mimo podstawowego zagospodarowania, popełnia błędy całego rynku usług. W dalszej części pracy została dokonana analiza rynku płatności cyfrowych w celu dostarczenia jak najlepszego systemu dla użytkownika końcowego. W analizie zostały nie tylko porównane obecne rozwiązania (przelewy tradycyjne, pay-by-link, karta płatnicza i portfele elektroniczne), ale także spośród zwycięzcy na tym polu – portfeli elektronicznych – zostało przeprowadzone zestawienie obecnych liderów na tym rynku. Zwycięskim rozwiązaniem okazał się PayPal.

Celem pracy było dostarczenie aplikacji webowej która w łatwy sposób pozwoli na zarządzanie korepetycjami z punktu widzenia ucznia nauczyciela. Rozwiązanie takie zostało wykonane w sposób modularny, zgodnie z *Czystą Architekturą*. Aplikacja jest otwarta na modyfikacje w przyszłość i na wymianę poszczególnych elementów. Ponadto, struktura aplikacji pozwoli w przyszłości zastosować implementację innych systemów płatności. Wpływa to przede wszystkim na cykl życia aplikacji, która przez kolejne lata może nie być dalej rozwijana, ale same płatności portfelem elektronicznym będą aktualizowane. Od strony frontendowej, aplikacja *Find-A-Tutor* oferuje przejrzysty i czytelny interfejs który ułatwi użytkowanie portalu.

Zaimplementowany system PayPal, który zwyciężył w analizie rynkowej,

jest liderem płatności online. Inne firmy technologiczne, nawet te największe, takie jak Facebook lub Google są ze swoimi rozwiązaniami daleko za PayPal'em. Pomimo tego, że implementacja i testy nowych portfeli elektronicznych mogą być gotowe w niedalekiej przyszłości, to dogonienie konkurencji pod względem przekonania do siebie użytkowników i utwierdzenie ich w rewolucyjności ich rozwiązania może zająć lata.

W przyszłości rynek usług będzie z pewnością ewoluował wchłaniając kolejne gałęzie przemysłu. Będzie to oznaczało zwiększenie agregatów ogłoszeń (na przykład www.olx.pl) i powstanie zupełnie nowych aplikacji, takich jak *Find-A-Tutor*. Zaproponowana aplikacja może stanowić ciekawe rozwiązanie dla osób próbujących stworzyć autorską implementację serwisu ogłoszeniowego.

6 Indeks rysunków

2.1	Dodawanie ogłoszenia na serwisie OLX	11
2.2	Przykładowa płatność pay-by-link	14
2.3	Smart Payment Button	20
3.1	Kontrola kondycji aplikacji <i>Find-A-Tutor</i>	29
3.2	Mapowanie klas na struktury bazodanowe	31
3.3	Przykład podstawowej struktury JWT [21]	32
3.4	Przykład nagłówka tokenu dla standardu JWT	33
3.5	Przykład zawartości tokenu dla standardu JWT	33
3.6	Sposób obliczania podpisu w standardzie JWT	34
3.7	Działanie tokenu w praktyce [22]	35
3.8	Interfejs Swagger UI dla aplikacji <i>Find-A-Tutor</i>	36
3.9	Szczegóły zapytania GET	37
3.10	Czysta Architektura [15].	39
3.11	Schemat zapisu danych typu <i>obiekt</i> w formacie JSON	42
3.12	Rezultat wysłanego zapytania do WebAPI	43
3.13	Schemat wzorca Model-Widok-Kontroler [24]	44
3.14	Zasada organizacji wzorca projektowego MVC [25]	46
3.15	Struktura bazy danych dla aplikacji <i>Find-A-Tutor</i>	47
4.1	Strona startowa aplikacji <i>Find-A-Tutor</i>	49
4.2	Panel logowania do panelu użytkownika	50
4.3	Formularz rejestracji	51
4.4	Pusty panel ucznia w aplikacji <i>Find-A-Tutor</i>	51

4.5	Formularz do dodania nowego ogłoszenia w aplikacji <i>Find-A-Tutor</i>	52
4.6	Komunikat po dodaniu nowego ogłoszenia przez ucznia w aplikacji <i>Find-A-Tutor</i>	53
4.7	Panel ucznia po dodaniu nowego ogłoszenia w aplikacji <i>Find-A-Tutor</i>	53
4.8	Pusty panel nauczyciela w aplikacji <i>Find-A-Tutor</i>	54
4.9	Panel nauczyciela w aplikacji <i>Find-A-Tutor</i>	54
4.10	Podjęcie ogłoszenia przez nauczyciela w aplikacji <i>Find-A-Tutor</i>	55
4.11	Panel nauczyciela w aplikacji <i>Find-A-Tutor</i>	56
4.12	Panel ucznia w aplikacji <i>Find-A-Tutor</i>	56
4.13	Szczegóły ogłoszenia w aplikacji <i>Find-A-Tutor</i> w widoku ucznia	57
4.14	Płatność przy użyciu PayPal	58
4.15	Szczegóły zapłaconego ogłoszenia w aplikacji <i>Find-A-Tutor</i> w widoku ucznia i studenta	59

7 Indeks tabel

2.1	Porównanie portfeli elektronicznych	19
2.2	Porównanie serwisów dla korepetytorów	21

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-sklepov-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/xabari1/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/jwt-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