POLITECHNIKA LUBELSKA

Wydział Elektrotechniki i Informatyki

Kierunek Informatyka



PRACA INŻYNIERSKA

**Aplikacja do zarządzania gabinetem stomatologicznym**

**DENTAL CLINIC ADMINISTRATION APPLICATION PROGRAM**

|  |  |
| --- | --- |
| Dyplomanci:  **Adrian Perec** nr albumu:  **Maciej Sulenta** nr albumu:  **Kamil Szalast** nr albumu: **83274** | Promotor:  **dr Mariusz Dzieńkowski** |

Lublin 2022

**Spis treści**

Streszczenie 5

Abstract 6

1. Wstęp 7

2. Cel i zakres pracy 9

2.1. Cel pracy 9

2.2. Zakres pracy 9

2.3. Podział pracy 10

2.4. Słownik pojęć 10

3. Analiza potrzeb rynku 11

3.1. Sytuacja zastana oraz identyfikacja potrzeb 11

3.2. Przegląd systemów do rezerwacji wizyt stomatologicznych 12

3.3. Ograniczenia dostępnych systemów oraz uzasadnienie utworzenia dedykowanego systemu 14

4. Technologie i narzędzia wykorzystane do budowy systemu 16

4.1. Uzasadnienie wyboru architektury internetowej 16

4.2. Stos technologiczny 16

4.2.1. Środowisko wytwórcze 17

4.2.2. Baza danych 17

4.2.3. Strona serwera 18

4.2.4. Strona klienta 21

5. Projekt systemu zarządzania wizytami gabinetu stomatologicznego 27

5.1. Procesy biznesowe 27

5.1.1. Obiekty biznesowe 27

5.1.2. Aktorzy biznesowi 27

5.1.3. Procesy biznesowe 28

5.2. Specyfikacja wymagań 29

5.2.1. Wymagania funkcjonalne 29

5.2.2. Wymagania niefunkcjonalne 30

5.3. Analiza wymagań 31

5.3.1. Statyczne elementy aplikacji 31

5.3.2. Panel lekarza 32

5.3.3. Panel pacjenta 32

5.3.4. Panel administratora 32

5.4. Projekt struktury bazy danych 33

5.4.1. Struktura bazy danych 34

5.4.2. Role i uprawnienia użytkowników 37

6. Implementacja 39

6.1. Baza danych 39

6.2. Serwer aplikacji 39

6.3. Aplikacja internetowa 39

6.3.1. Funkcjonalność aplikacji dostępna dla niezalogowanego użytkownika 39

6.3.2. Panel lekarza 39

6.3.3. Panel pacjenta 39

6.3.4. Panel administratora 39

7. Testowanie 40

7.1. Testy zabezpieczeń 40

7.2. Testy funkcjonalne 40

7.3. Testy niefunkcjonalne 40

8. Podsumowanie 41

Literatura 42

# Streszczenie

# Abstract

# Wstęp

W dobie internetu oraz swobodnego dostępu do sieci, ludzie coraz częściej sięgają po nowoczesne rozwiązania z sektora informatycznego, które to pośredniczą w zamawianiu różnego rodzaju usług. Przykładem może być aplikacja mobilna firmy Bolt, która odniosła zauważalny sukces w obszarze przewozów taksówkowych. Przedsiębiorstwo to jest obecnie warte ponad miliard dolarów, a z jego usług korzysta ponad 25 milionów użytkowników na świecie [strona 1]. Wobec tego stwierdzić można, iż odpowiednio zaprojektowany i zaimplementowany system informatyczny jest w stanie zastąpić nawet tak trywialną czynność jak zamówienie prywatnego przejazdu do określonego miejsca docelowego.

Przedmiotem niniejszego opracowania jest z kolei aplikacja internetowa pozwalająca umówić wizytę stomatologiczną. Proces rejestracji pacjenta na wizytę jest o wiele bardziej złożony i wymaga przepływu zdecydowanie większej ilości informacji, niż uprzednio wspomniana rezerwacja przejazdu taksówkowego. Mianowicie podczas rejestracji na wizytę należy podać swoje dane osobowe, telefon kontaktowy, oraz ewentualne objawy przewidywanej choroby. Podczas telefonicznej rezerwacji pacjent będzie miał do wyboru jeden spośród kilku zaproponowanych przez rejestratorkę (lub rejestratora) określony termin wizyty, który jednocześnie będzie musiał dopasować do już wcześniej zaplanowanych przez siebie zajęć i obowiązków życia codziennego. Taki sposób zamawiania wizyty stomatologicznej jest oczywiście mniej wygodny, niż realizacja tej samej czynności z pomocą systemu informatycznego, którego przewagą będzie m. in. klarowne dostarczenie informacji o dostępnych terminach wizyt i rodzajach usług świadczonych przez poszczególnych specjalistów.

Dość istotnym elementem funkcjonowania gabinetów lekarskich (w tym stomatologicznych) jest konieczność prowadzenia dokumentacji każdego pacjenta. Każdy kuracjusz powinien posiadać folder, który zawiera dane dotyczące jego stanu zdrowia, występujących chorób oraz udzielonych mu świadczeń zdrowotnych. W dokumentacji takiej powinny znajdować się także wyniki badań, zdjęcia rentgenowskie itd. [strona 2]. W wielu placówkach medycznych takie archiwum prowadzone jest wciąż w formie papierowej, co niesie ze sobą pewne ograniczenia. Po pierwsze obieg informacji w zakładzie medycznym korzystającym z takiej formy bazy danych jest ograniczony, ponieważ każdy pracownik musi posiadać fizyczną kopię (lub oryginał) danych medycznych pacjenta. Z kolei kuracjusze, którzy potrzebują dostępu do swojej dokumentacji muszą pojawić się osobiście w placówce, aby otrzymać ją do wglądu lub ewentualnie uzyskać odpłatną kopię.

Zmiana formy prowadzenia dokumentacji medycznej z papierowej na cyfrową niesie więc za sobą wymierne korzyści dla wszystkich pracowników placówki oraz jej pacjentów. Należy jednak pamiętać, iż taki system powinien przede wszystkim zapewniać ochronę danych osobowych zarówno lekarzy jak i kuracjuszy. System informatyczny użytkowany jednocześnie przez pacjentów oraz pracowników placówki medycznej powinien posiadać interfejs graficzny, który będzie estetyczny oraz intuicyjny w obsłudze. Należy mieć na uwadze fakt, że stomatolodzy to w dużej mierze osoby w podeszłym wieku, które nie zawsze są w stanie biegle poruszać się w systemach komputerowych. Wobec tego autorzy przewidzieli w projektowanym systemie rolę administratora, który będzie mieć dostęp do funkcjonalności przypisanych lekarzowi.

Pierwszy rozdział niniejszej pracy inżynierskiej zawiera wstęp, w którym przedstawiono uzasadnienie doboru tematu niniejszego opracowania, a także przedstawiono korzyści płynące z potencjalnego zastosowania systemu informatycznego zaprojektowanego przez autorów pracy.

W drugim rozdziale przedstawiono cel oraz zakres pracy wraz z krótkim przedstawieniem podstawowych funkcjonalności projektowanej aplikacji internetowej. W dwóch ostatnich podrozdziałach rozdziału drugiego przedstawiono podział pracy pomiędzy jej autorów oraz wyjaśniono podstawowe pojęcia związane z projektowanym systemem informatycznym.

TUTAJ OPISAĆ ROZDZIAŁY TRZECI I DALSZE

# Cel i zakres pracy

W niniejszym rozdziale przedstawiony został cel oraz zakres pracy inżynierskiej. Dodatkowo przedstawiono także podział prac między autorów niniejszego opracowania i opisano podstawowe pojęcia stosowane do opisu projektowanego systemu informatycznego.

## Cel pracy

Celem niniejszego opracowania było zaprojektowanie oraz zaimplementowanie aplikacji służącej do zarządzania gabinetem stomatologicznym. Wykonany przez autorów niniejszej pracy inżynierskiej serwis internetowy zapewniałby odpowiednią funkcjonalność i ergonomiczność ustalania wizyt lekarskich. Aplikacja przeznaczona jest do użytku przez pacjentów rejestrujących się na wizyty, lekarzy świadczących określony zakres usług medycznych, oraz sekretariat czy też administrację potencjalnej placówki medycznej.

## Zakres pracy

Zakres opracowania składa się z części studialnej, w której to omówiony został kod źródłowy aplikacji, a w szczególności zastosowane przez autorów pracy inżynierskiej szkielety aplikacyjne. W części tej przedstawiono poszczególne rozwiązania programistyczne zastosowane w projekcie odpowiedzialne zarówno za stronę wizualną jak i logiczną projektowanego serwisu internetowego.

Kolejny element opracowania stanowi kod źródłowy zaimplementowanej aplikacji internetowej. Utworzony system składa się z części statycznej, odpowiedzialnej za dostarczanie podstawowych informacji o gabinecie stomatologicznym. Osoba przeglądająca stronę internetową będzie miała dostęp do sekcji „O nas”, zawierającej informacje o lekarzach specjalistach, a także o sprzęcie medycznym dostępnym w gabinetach dentystycznych. Na stronie pojawi się także rubryka „Oferta” w której to potencjalny pacjent będzie miał możliwość zapoznania się z spektrum świadczonych usług. Pacjent, który chciałby skorzystać z usług placówki miałby także dostęp do wykazu cen poszczególnych zabiegów w sekcji o nazwie „Cennik”. W celu zapewnienia odpowiednich informacji kontaktowych, na stronie umieszczone zostaną podstawowe dane takie jak: adres placówki (wraz z lokalizacją na mapie), godziny otwarcia, telefon kontaktowy oraz adres elektronicznej skrzynki pocztowej. Dodatkowo pacjent będzie miał możliwość skorzystać z formularza kontaktowego w celu uzyskania dodatkowych informacji.

Główną funkcjonalność aplikacji internetowej stanowił będzie system rezerwacji wizyt stomatologicznych. W ramach tworzonego oprogramowania rozróżniono 3 typy użytkowników: pacjent, lekarz, administrator systemu. Z poziomu strony głównej witryny użytkownik będzie miał możliwość zarezerwowania wizyty na konkretny zabieg u wybranego lekarza na dowolny, dostępny w grafiku stomatologa termin. Podczas opisanego powyżej procesu, po wypełnieniu danych osobowych potencjalny pacjent będzie miał możliwość założenia konta w systemie. Tak utworzone konto będzie dawało użytkownikowi dostęp do panelu pacjenta wyposażonego w możliwość rezerwacji nowej wizyty, anulowania przyszłej wizyty, zmiany danych osobowych oraz usunięcia konta.

Lekarze korzystający z systemu będą dostawali dane logowania (login i hasło) od administratora systemu, który to będzie zakładał konta dla zatrudnionych stomatologów. Medycy po zalogowaniu do systemu będą mieli możliwość odwoływania nadchodzących świadczeń zdrowotnych, a dodatkowo udostępniona im zostanie historia przeprowadzona przez nich badań lekarskich. Każdy lekarz będzie miał także możliwość edytowania swoich danych oraz usunięcia konta.

Użytkownik określony jako administrator z poziomu systemu będzie upoważniony do tworzenia wizyt, ich edytowania i anulowania. Dodatkowo będzie miał on możliwość dodawania użytkowników typu pacjent oraz lekarz do systemu, a także zmiany ich danych.

## Podział pracy

Tabela 2.1 zawiera podział prac pomiędzy autorów niniejszego opracowania.

Tabela 2.1. Podział pracy

|  |  |
| --- | --- |
| Dyplomant | Numer oraz nazwa rozdziału |
| Adrian Perec |  |
| Maciej Sulenta |  |
| Kamil Szalast |  |

## Słownik pojęć

Na potrzeby niniejszej pracy inżynierskiej zdefiniowane zostały podstawowe pojęcia określające rolę poszczególnych użytkowników projektowanego systemu informatycznego:

Pacjent – jest to osoba, która korzysta z usług placówki medycznej, która mogłaby użytkować projektowaną aplikację internetową.

Lekarz – reprezentuje wykwalifikowanego w danej specjalizacji pracownika gabinetu stomatologicznego.

Administrator – użytkownik mający pełną kontrolę nad najistotniejszymi elementami systemu. W praktyce jest to osoba zajmująca się telefoniczną rejestracją pacjentów na wizyty, a także zarządzaniem istniejącymi wizytami oraz grafikami lekarzy.

# Analiza potrzeb rynku

Jak wspomniano we wstępie niniejszego opracowania, coraz więcej firm świadczących różnej maści usługi decyduje się na przekształcenie swojego dotychczasowego sposobu prowadzenia grafiku wykonywanych prac. Usługodawcy zaopatrzając się w informatyczny system rezerwacji świadczeń (w tym przypadku są to oczywiście wizyty stomatologiczne) są w stanie zaoszczędzić sobie oraz swoim pracownikom wysiłku związanego z prowadzeniem papierowej dokumentacji.

## Sytuacja zastana oraz identyfikacja potrzeb

Spośród analizowanych systemów zapewniających internetową rezerwację wizyt stomatologicznych najczęściej spotykanym jest aplikacja internetowa. Tego typu system ma jedną niepodważalną zaletę: nie wymaga on instalacji, ponieważ jest dostępny bezpośrednio w przeglądarce internetowej, bez znaczenia czy użytkownik korzysta z wersji na komputery stacjonarne czy z wersji przeznaczonej na telefony komórkowe.

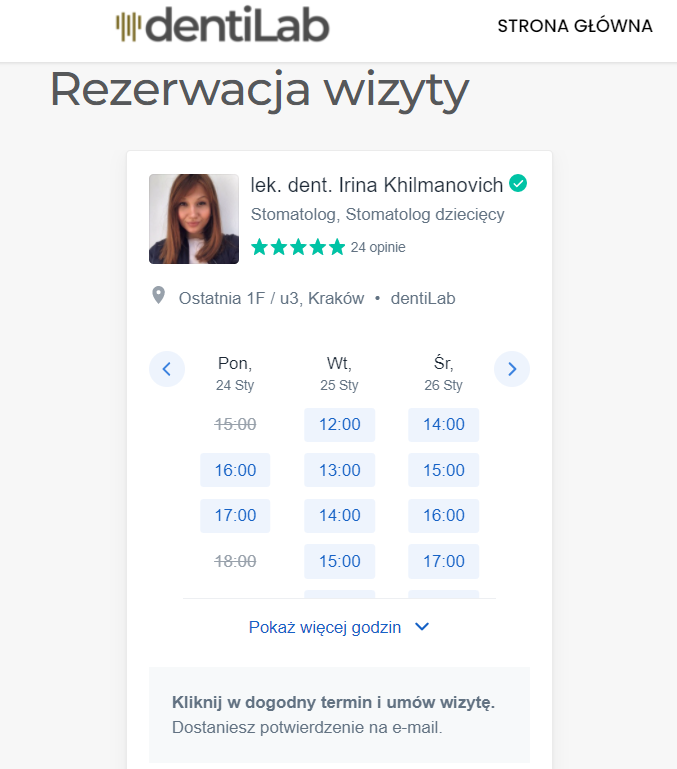
Znaczny odsetek analizowanych przez autorów pracy inżynierskiej gabinetów stomatologicznych korzysta z zewnętrznego systemu rezerwacji jakim jest strona internetowa pod adresem https://www.znanylekarz.pl/ [strona 3]. Rozwiązanie tego typu jest korzystne dla placówek medycznych, ponieważ może zapewnić szereg usprawnień:

* brak konieczności prowadzenia ewidencji wizyt w tradycyjnej formie papierowej, która jak wiadomo niesie ze sobą znaczne ograniczenia,
* możliwość zredukowania liczby pracowników potrzebnych do zarządzania placówką medyczną,
* zapewnienie stałym klientom placówki dostępu do historii leczenia oraz zaleceń sporządzonych przez lekarza prowadzącego leczenie,
* możliwość oparcia kampanii marketingowej placówki medycznej o nowoczesny system rezerwacji, który z pewnością przyciągnie młodych klientów.

Niestety wciąż wiele gabinetów nie korzysta z przedstawionego powyżej systemu informatycznego, w zamian zamieszczając na swoich stronach internetowych proste formularze kontaktowe [strona 4]. Takie rozwiązanie wymaga od potencjalnego pacjenta podania swoich danych osobowych i kontaktowych, aby w dalszej kolejności rejestratorka zatrudniona przez gabinet stomatologiczny mogła skontaktować się z nim i ustalić termin wizyty. Nie jest to zbyt ergonomiczne rozwiązanie, gdyż klient placówki medycznej może zamiast tego po prostu sam skontaktować się z rejestracją i od razu umówić konkretny termin interesującego go zabiegu.

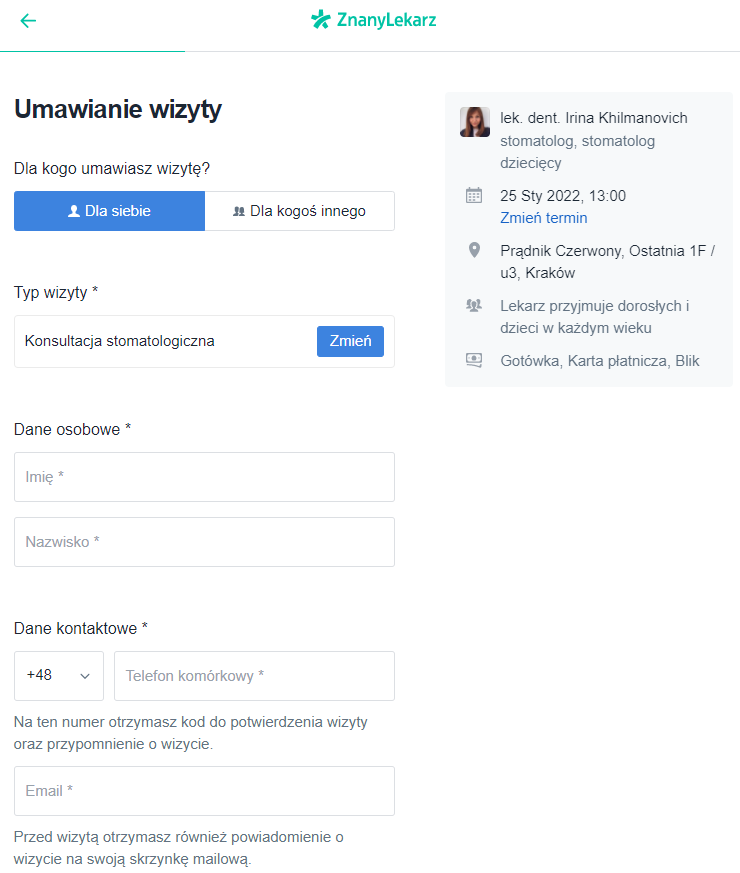
## Przegląd systemów do rezerwacji wizyt stomatologicznych

Wspomniany w poprzednim rozdziale zewnętrzny system rezerwacji wizyt jest najczęściej wybieranym rozwiązaniem przez gabinety stomatologiczne. Na niektórych stronach [strona 5]autorzy zastosowali gotowy element (Rys. 3.1.) służący do przekierowania użytkownika na witrynę z formularzem rejestracji. Gdy użytkownik kliknie w interesującą go godzinę wizyty, zostanie oddelegowany do formularza rejestracji znajdującego się w zewnętrznym serwisie (Rys. 3.2.).



Rys. 3.1. Zdjęcie ekranu przedstawiające element utworzony przez autorów serwisu [strona 3] zastosowany na stronie internetowej gabinetu stomatologicznego

Takie rozwiązanie zapewnia sprawną rezerwację konkretnego terminu wizyty, aczkolwiek przekierowanie użytkownika witryny do innej podstrony może wzbudzić pewną dozę nieufności. Dodatkowym atutem tego konkretnego rozwiązania jest przejrzystość graficznego interfejsu grafika lekarza z dostępnymi terminami. Sam formularz rejestracji zamieszczony w zewnętrznym serwisie jest klarowny i wymaga podania jedynie niezbędnych informacji, a co najważniejsze, przez cały czas użytkownik (w prawym górnym roku formularza) ma wgląd w najważniejsze dane na temat rezerwowanej przez siebie wizyty.



Rys. 3.2. Zdjęcie ekranu formularza rejestracyjnego znajdującego się w popularnym serwisie [strona 3]

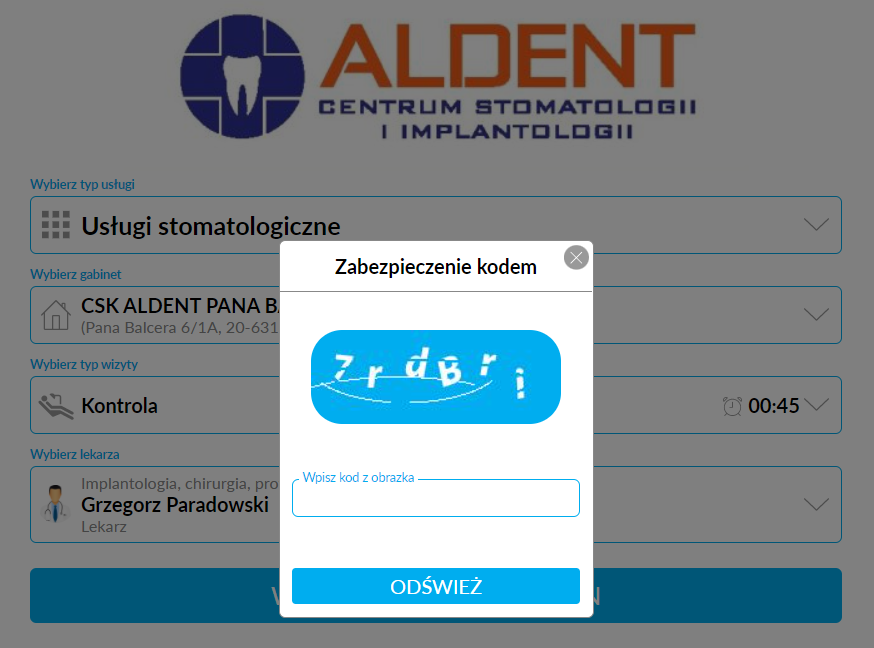
Kolejnym, najbardziej zbliżonym do zaproponowanego w ramach tematu pracy inżynierskiej rozwiązaniem jest własny system informatyczny odpowiadający za rezerwację wizyt w placówce medycznej. Za przykład posłuży tutaj witryna internetowa https://aldent.lublin.pl/, która korzysta z oprogramowania firmy o nazwie „FELG DENT” (Rys. 3.3.).



Rys. 3.3. Zdjęcie ekranu panelu służącego do wyszukania oraz późniejszej rezerwacji terminu wizyty

Przedstawiony na Rys. 3.3. system różni się od poprzednio omawianego kilkoma istotnymi cechami. Po pierwsze użytkownik po kliknięciu przycisku „Umów się na wizytę” (znajdującego się na stronie głównej kliniki) zostaje przekierowany do nowej podstrony, jednakże tam wciąż widnieje logo gabinetu stomatologicznego. Pozwala to zachować pewną iluzję, że pozostajemy na tej samej witrynie internetowej i nie wprowadza wspomnianego wcześniej poczucia nieufności.

Po kliknięciu przycisku „Wyszukaj wolny termin” zatwierdzającego wypełniony formularz, przed jego wysłaniem system informatyczny przeprowadza dodatkową weryfikację w formie testu CAPTCHA (Rys. 3.4) (ang. *Completely Automated Public Turing test to tell Computers and Humans Apart*)[książka o captcha]. Test ten ma za zadanie zablokować przesyłanie formularza dla nieautentycznych użytkowników, którymi najczęściej jest złośliwe oprogramowanie. Wobec tego musi być on łatwy do rozwiązania przez człowieka, natomiast trudny do odszyfrowania przez programy komputerowe – najczęściej ta metoda autoryzacji przyjmuje formę zniekształconego napisu który należy wpisać w odpowiednie pole. Jest to bardzo znacząca cecha systemu, ponieważ odpowiada ona za jego bezpieczeństwo.



Rys. 3.4. Zdjęcie ekranu przedstawiające test CAPTCHA

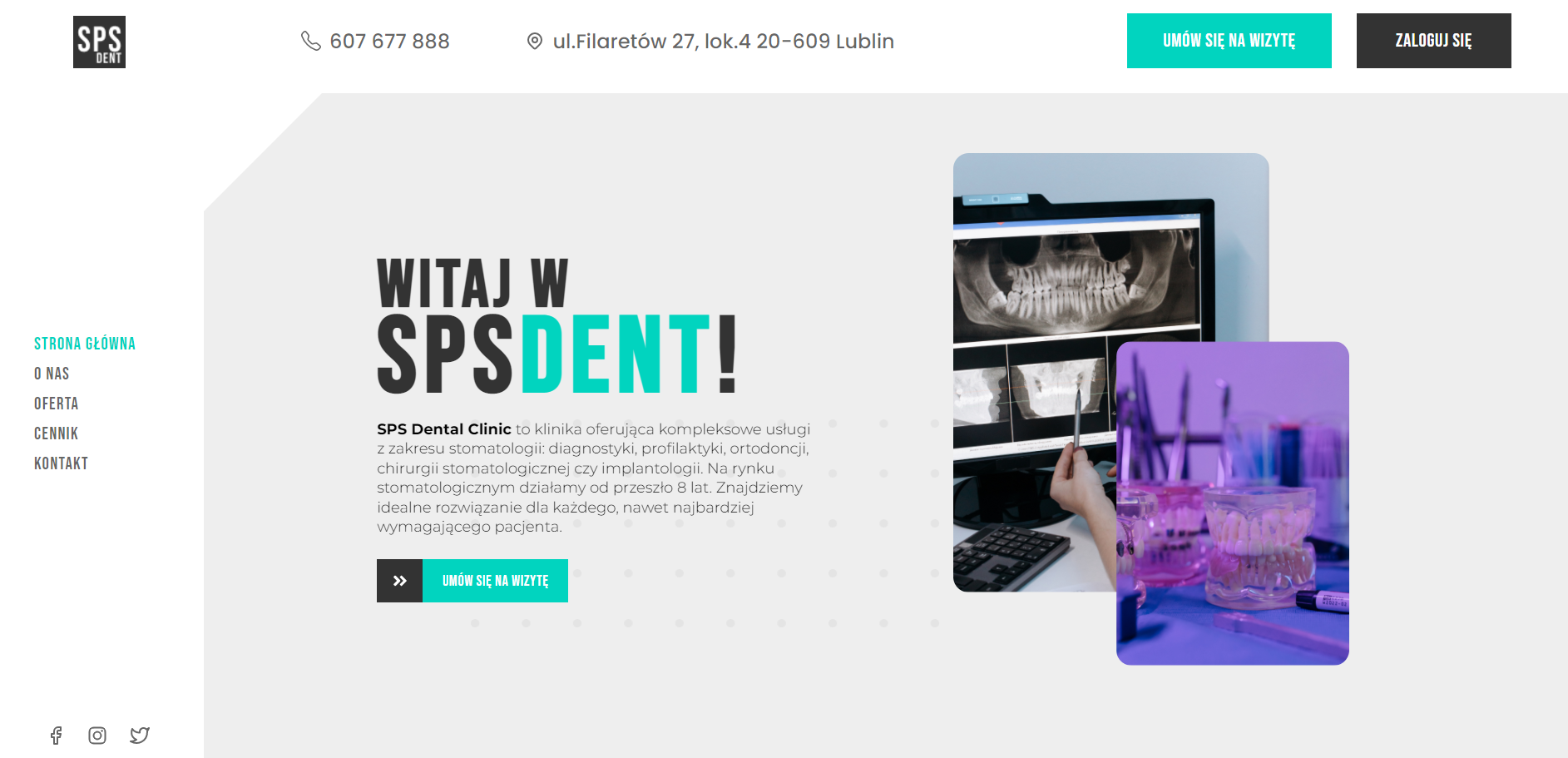
## Ograniczenia dostępnych systemów oraz uzasadnienie utworzenia dedykowanego systemu

Omówione w poprzednim podpunkcie bieżącego rozdziału aplikacje internetowe nie zostały zaopatrzone w pewną funkcjonalność, która według autorów niniejszego opracowania jest bardzo istotna. Chodzi tutaj o możliwość założenia konta w systemie wspomagającym zarządzanie gabinetem stomatologicznym oraz o jego funkcjonalność.

W przytoczonym powyżej przykładzie serwisu zewnętrznego o nazwie „ZnanyLekarz” pacjent ma co prawda możliwość założenia konta, jednakże konto to nie jest ściśle powiązane z jakąkolwiek placówką medyczną. Wobec tego pacjent nie ma dostępu do ewentualnych zaleceń lekarza ani do swojej historii leczenia.

Z kolei przykład serwisu stworzonego przez firmę „FELG DENT” nie posiada możliwości założenia konta w systemie. To natomiast oznacza, iż stali pacjenci kliniki będą zobligowani do każdorazowego wpisywania swoich danych osobowych i kontaktowych podczas rezerwacji wyżyty. Takie rozwiązanie z pewnością zostałoby uznane przez stałych użytkowników jako nieergonomiczne i niewygodne.

Autorzy niniejszego opracowania stworzyli aplikację internetową, która skupia się na dostarczeniu brakujących w wymienionych przykładach rozwiązań zarówno potencjalnym pacjentom jak i placówce medycznej, która docelowo korzystałaby z zaprojektowanego systemu. Oprogramowanie zaopatrzone zostało w przejrzysty formularz rejestracji, ergonomiczny panel służący do rezerwowania wizyty stomatologicznej, a także w odpowiednie panele dla każdego z typów użytkowników, tzn. lekarza, pacjenta oraz administratora. Twórcy oprogramowania zadbali także o klarowny i sugestywny graficzny interfejs witryny internetowej (Rys. 3.5.) korzystając z odpowiednich narzędzi omówionych w kolejnym rozdziale.



Rys. 3.5. Zdjęcie ekranu strony głównej witryny zaprojektowanej przez twórców niniejszego opracowania

# Technologie i narzędzia wykorzystane do budowy systemu

W bieżącym rozdziale omówione zostaną technologie oraz szkielety programistyczne zastosowane podczas implementacji aplikacji będącej tematem niniejszego opracowania.

## Uzasadnienie wyboru architektury internetowej

Wybierając temat pracy autorzy zdecydowali się na zaprojektowanie aplikacji internetowej, ponieważ system o takiej architekturze perfekcyjnie spełnia wszystkie wymagania funkcjonalne projektowanego serwisu.

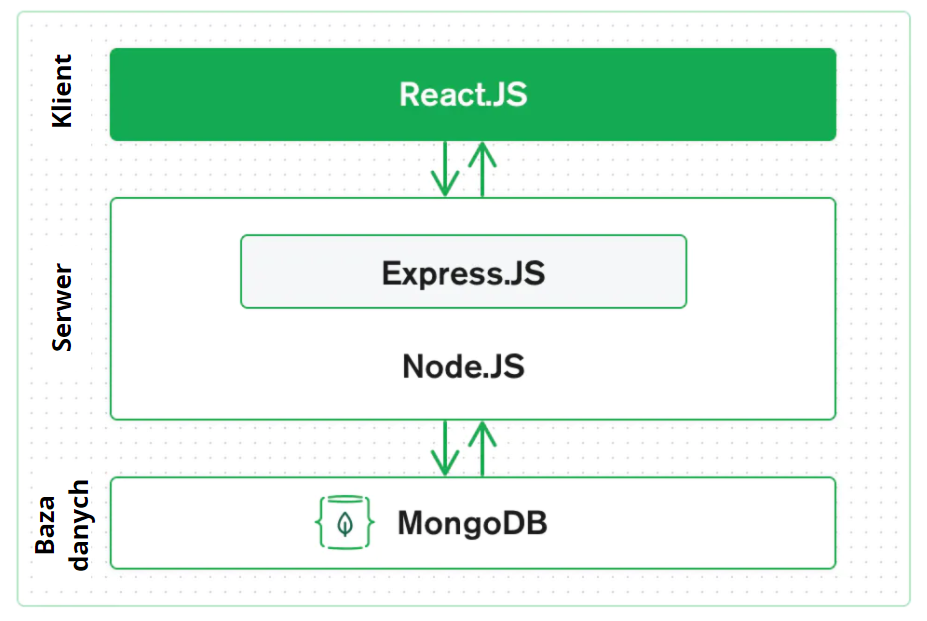
Najważniejszymi zaletami zastosowanej architektury jest jej niezależność od systemu operacyjnego oraz brak konieczności instalacji i konfiguracji systemu na urządzeniu. Tak zaprojektowany system może być obsługiwany za pośrednictwem dowolnego urządzenia, które posiada przeglądarkę internetową oraz dostęp do sieci.

## Stos technologiczny

W celu zaprojektowania responsywnej oraz interesującej graficznie strony internetowej autorzy niniejszego opracowania zastosowali odpowiednie szkielety programistyczne oraz biblioteki, opierając większość projektowanego przez siebie systemu o język programowania JavaScript. W nomenklaturze informatycznej stos technologiczny, który został zastosowany nosi nazwę MERN [strona 7]. Jest to skrótowiec utworzony na podstawie pierwszych liter następujących technologii:

* MongoDB – nierelacyjny system bazodanowy
* Express – szkielet programistyczny służący do budowania aplikacji sieciowych po stronie serwera o nazwie Node.JS
* React.JS – biblioteka języka JavaScript do tworzenia strony klienckiej systemu
* Node.JS – najbardziej popularna platforma serwerowa języka JavaScript

Autorzy niniejszego opracowania zdecydowali się na powyższy stos technologiczny ze względu na swoje zainteresowania technologiami internetowymi oraz przez zamiar znalezienia pierwszej pracy w branży informatycznej.



Rys. 4.1. Grafika przedstawiająca funkcjonowanie aplikacji internetowej o stosie technologicznym MERN

### Środowisko wytwórcze

**Visual Code Studio – darmowy edytor tekstowy**

W celu zaimplementowania systemu informatycznego autorzy wykorzystali edytor tekstowy o nazwie Visual Code Studio w wersji 1.63 wraz z niezbędnymi dodatkowymi rozszerzeniami w celu poprawienia ergonomiki sporządzania kodu źródłowego. Oprócz swojej podstawowej funkcji omawiany edytor tekstowy posiada także wsparcie dla zarządzania wersjami kodu źródłowego za pośrednictwem systemu kontroli wersji o nazwie GIT.[strona 6]

### Baza danych

W celu przechowywania danych wykorzystany został otwarty, nierelacyjny system bazodanowy o nazwie MongoDB. Podstawową cechą tego systemu jest brak konkretnie zdefiniowanej struktury baz danych utworzonych w tym systemie. W systemie MongoDB dane nie są przechowywane w serii rekordów oraz tabel (jak ma to miejsce w relacyjnych systemach baz danych), lecz w formie dokumentów i kolekcji [książka Nowoczesne aplikacje internetowe]. Pojedynczy dokument (który w pewnym sensie jest odpowiednikiem rekordu) przechowywany jest w formacie o nazwie BSON, co z kolei oznacza binarną postać danych formatu o nazwie JSON (ang. *JavaScript Object Notation*). Poniżej wymieniono zalety przechowywania danych w wymienionym wcześniej formacie BSON:

* zastosowanie dokumentów eliminuje konieczność stosowania złączeń jak w przypadku relacyjnych baz danych,
* pojedynczy dokument jest w naturalny sposób przetwarzany na obiekty w kodzie źródłowym programu.

Kolejną charakterystyczną cechą bazy danych w systemie MongoDB jest brak definiowania tabel oraz typów danych przechowywanych w pojedynczym dokumencie. Dzięki temu w każdym dokumencie można przechowywać dane różnego typu, a system bazodanowy nie będzie generował komunikatów o błędach.

Kolejną różnicą pomiędzy MongoDB a relacyjnymi bazami danych jest inne podejście do przygotowania struktury przechowywanych informacji. W systemach relacyjnych baz danych np. SQL istotne jest aby dane zostały jak najbardziej znormalizowane. Pozwala to zapobiec powielaniu informacji. W MongoDB głównym celem jest maksymalne ułatwienie korzystania z danych w projektowanej aplikacji, co z kolei ułatwia programistom przygotowanie zapytań do bazy danych.

W celu konfiguracji połączenia bazy danych z aplikacją internetową autorzy wykorzystali także szkielet programistyczny o nazwie „Express”. Jednakże mechanizmy funkcjonowania systemu od strony serwera zostaną omówione w kolejnym podrozdziale.

### Strona serwera

Za mechanizmy działające po stronie serwera zaprojektowanej aplikacji internetowej odpowiada język programowania JavaScript oraz szkielety programistyczne korzystające z niego.

**Język programowania JavaScript**

JavaScript określić można jako język programowania sieci. Wszechobecność tego języka w strukturze stron internetowych spowodowana jest faktem, iż większość przeglądarek internetowych dostępnych na urządzeniach różnego typu (komputerach stacjonarnych, telefonach komórkowych, tabletach itd.) posiada wbudowany silnik do obsługi kodu napisanego w JavaScript. JS bo tak można w skrócie określić wyżej wymieniony język, jest jednym z trzech podstawowych technologii służących do budowy stron internetowych, równolegle z HTML (ang. *HyperText Markup Language*) oraz CSS (ang. *Cascading Style Sheets*). HTML jest językiem znaczników, stosowanym do tworzenia dokumentów hipertekstowych, natomiast CSS to język służący do opisu formy w jakiej powinny być wyświetlane strony internetowe [JS-DefiniteGuide].

JavaScript jest językiem programowania odpowiedzialnym za zdefiniowanie odpowiednich zachowań stron internetowych. Poniżej wypunktowane zostały jego podstawowe właściwości:

* oparty na wielu paradygmatach
* bazuje na prototypach
* jego składnia jest dynamiczna
* zorientowany obiektowo
* stylu deklaratywnym i imperatywnym

Ten skryptowy język programowania, w którym funkcje pełnią jednocześnie rolę obiektów. Referencje do tych obiektów można natomiast przechowywać w zmiennych. Opisywany język programowania swoją popularność zyskał poprzez zapewnienie odpowiedniej funkcjonalności dynamicznych stron internetowych. Jednakże przeglądarki internetowe to nie jedyne środowisko, w którym możemy wykonać kod źródłowy napisany w JS [strona 8]. Przykładem może być zastosowane przez autorów niniejszego opracowania środowisko uruchomieniowe o nazwie Node.JS lub inne narzędzia programistyczne jak np. Apache CouchDB, Adobe Acrobat.

W tym miejscu należałoby także wspomnieć o standardach języka JavaScript, które jak wiadomo, w branży informatycznej zmieniają się dość dynamicznie. Taką ustandaryzowaną specyfikacją dla JS jest standard o nazwie ECMAScript. Wersją specyfikacji, która wprowadziła diametralne zmiany, jest wydana w 2015 roku wersja szósta nazywana skrótem ES6. Poniżej wypunktowane zostały główne zmiany, które zastosowano:

* słowa kluczowe *let* i *const*, które pozwalają zadeklarować odpowiednio zmienną w zakresie danego bloku kodu i stałą,
* definiowanie funkcji strzałkowych, czyli skróconego sposobu zapisu funkcji o następującej składni **(x, y) => {};**, gdzie x oraz y oznaczają argumenty funkcji, a w nawiasach klamrowych zawiera się ciało funkcji,
* zastosowanie klas jako szablonu do tworzenia obiektu w języku JavaScript,
* umożliwienie programistom korzystania z mechanizmu dziedziczenia, dzięki słowu kluczowemu *extends*,
* wprowadzenie dodatkowych funkcji pomocniczych w klasach odpowiadających za operacje wykonywane na tablicach oraz na łańcuchach znaków.

**Wieloplatformowe środowisko uruchomieniowe Node.JS**

Node.JS jest to wieloplatformowe oprogramowanie, które pozwala tworzyć programistom kod źródłowy w języku JavaScript pracujący po stronie serwera. [strona 9]. Narzędzie to jest w stanie funkcjonować poza przeglądarką internetową, bezpośrednio współpracując z systemem operacyjnym. Ta kooperacja pozwala narzędziu jakim jest Node.JS na udostępnienie swoim programom interfejsu programowania aplikacji (ang. *API – application programming interface*) systemu operacyjnego. Wśród udostępnionych funkcjonalności znajdują się:

* dostęp do bibliotek systemowych
* dostęp do sytemu plików
* dostęp do uruchomionych procesów, a co najważniejsze do uruchomionych serwerów HTTP (ang. *Hypertext Transfer Protocol*)

Autorzy niniejszego opracowania zdecydowali się użyć dokładnie tej technologii po stronie serwera ze względu na jej kluczowe zalety.

Po pierwsze, kod źródłowy aplikacji zapewniającej funkcjonalność systemu po stronie serwera sporządza się w JavaScript. Takie podejście znacznie ułatwia pracę nad projektem, gdyż zarówno po stronie klienta jak i serwera wykorzystujemy ten sam język programowania. Z kolei sam JavaScript jest stosunkowo nowym językiem programowania i zapewnia liczne ulepszenia, w porównaniu do innych działających po stronie serwera takich jak np. PHP.

Po drugie Node.JS charakteryzuje się wysoką efektywnością działania. Twórcy tej technologii zaprojektowali ją tak, by optymalizowała wydajność i skalowalność aplikacji internetowych.

Kolejnym atutem Node.JS jest jego dostępność na różne systemy operacyjne. Środowisko to można zainstalować na systemach takich jak:

* Microsoft Windows,
* Linux,
* macOS,
* WebOS,
* i wiele innych.

Dodatkowo należy wspomnieć o obszernym wsparciu, które obejmuje zapewnienie odpowiedniej infrastruktury oraz dokumentacji dla omawianego środowiska uruchomieniowego, dostarczanym przez dostawców usług sieciowych.

Omawiając Node.JS warto także wspomnieć o menedżerze pakietów tegoż środowiska (ang. NPM – *Node Package Manager*). Technologia ta gwarantuje dostęp do wielu pakietów z których można korzystać budując aplikację internetową i jest wykorzystywana do automatyzacji narzędzi współtworzących system od strony serwera.

**Szkielet programistyczny Express**

Szkielet aplikacji internetowych Express zapewnia obsługę następujących mechanizmów potrzebnych dla części systemu po stronie serwera:

* konfiguracja podstawowych ustawień aplikacji internetowej (np. port na którym działa aplikacja),
* definiowanie funkcji mających za zadanie obsługę żądań HTTP i HTTPS o różnych metodach, skierowanych do różnych adresów URL,
* zarządzanie sesjami przeglądarki internetowej oraz obsługą błędów,
* serializacja i deserializacja danych w formacie JSON.

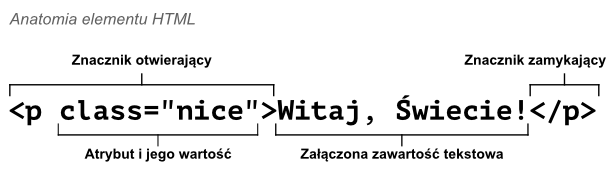
Należy także zaznaczyć iż szkielet programistyczny Express cechuje się bardziej przejrzystą strukturą kodu niż samo środowisko Node.JS, którą można zauważyć podczas implementacji następujących zadań:

* rejestracja danych,
* uwierzytelnianie danych,
* autoryzacja danych.

### Strona klienta

**HTML**

HTML jest to hipertekstowy język znaczników (ang. *HyperText Markup Language*), służący do tworzenia dokumentów hipertekstowych, które z kolei są interpretowane przez przeglądarki internetowe. HTML służy do opisania struktury witryn internetowych za pomocą segmentów nazywanych elementami HTML. [Jon Duckett HTML i CSS] Większość z tych elementów składa się z dwóch znaczników: otwierającego i zamykającego. Taki pojedynczy człon nadaje odpowiednie znaczenie semantyczne poszczególnym fragmentom strony internetowej. Dodatkowo każdy element HTML pełni funkcję pojemnika, wewnątrz którego umieścić można inne znaczniki HTML. Znaczniki HTML można rozszerzać o atrybuty, których wartości dostarczają dodatkowych informacji, dzięki którym twórcy kodu są w stanie zmienić sposób interpretacji znacznika przez przeglądarkę internetową (Rys. 4.2.). [strona 10]



Rys. 4.2. Anatomia elementu HTML

Warto byłoby nadmienić, iż aktualna wersja języka HTML to HTML5. Nad standardami tego hipertekstowego języka znaczników czuwa organizacja W3C (ang. *World Wide Web Consortium*), która w swych szeregach skupia wiele przedsiębiorstw prężnie działających w branży informatycznej m. in Adobe, Amazon, Apple, Microsoft. Konsorcjum W3C ustanowiło także podstawowe idee oraz cele dotyczące Internetu:

* uniwersalny dostęp dla wszystkich użytkowników, niezależnie od oprogramowania jakim dysponują oraz ich języka rodzimego,
* decentralizacja, która nie pozwala zagarnąć Internetu jednej firmie i ograniczyć go konkretną technologią,
* ewolucja – standardy W3C są tworzone z myślą o ich przyszłym rozwoju, gdyż ciągle zwiększa się ilość informacji w Internecie i dotychczasowe metody jej wyszukiwania i skalowania mogą przestać wystarczać,
* kompatybilność, która zapewnia możliwość uruchomienia nawet pierwszej stworzonej na świecie strony internetowej na najnowszych przeglądarkach internetowych.

**CSS**

Kaskadowe arkusze stylów (ang. *Cascading Style Sheets*) stanowią język, który pozwala tworzyć reguły określające wygląd zawartości elementów umieszonych na stronie internetowej. Technologia ta została opracowania w 1996 r. przez wspomniane uprzednio konsorcjum W3C. Umieszczenie listy dyrektyw odpowiadających za ustalanie wyglądu strony internetowej w osobnym pliku przyniosło ze sobą szereg korzyści:

* zmniejszenie zawiłości dokumentu HTML,
* uprawnienie zmiany sposobu wyświetlania strony internetowej w zależności od rozmiaru okna przeglądarki internetowej,
* rozdzielenie warstwy danych i warstwy prezentacji,
* ułatwienie wprowadzania zmian w strukturze dokumentu (aplikacja internetowa jest łatwiejsza w utrzymaniu),
* możliwość stosowania zewnętrznych arkuszy CSS, dzięki czemu jeden arkusz stylu może służyć do definiowania wyglądu kilku dokumentów HTML.

Działanie CSS polega na kojarzeniu poszczególnych reguł z elementami HTML. Reguły te definiują sposób wyświetlania konkretnych elementów na stronie internetowej, a składają się z dwóch części: selektora oraz deklaracji.

Selektor to wyrażenie, które określa do jakich elementów HTML odnosi się dana reguła, natomiast deklaracje określają w jaki sposób powinny zostać wyświetlone elementy określone w selektorze(Rys. 4.3.).



Rys. 4.3. Elementy składowe reguły CSS

Deklaracja natomiast składa się z dwóch elementów: właściwości i wartości. Pierwsza z nich odnosi się do konkretnego aspektu wyglądu elementu, druga natomiast wskazuje konkretne ustawienie, które chcemy nadać danej właściwości (Rys. 4.4.).



Rys. 4.4. Elementy składowe deklaracji CSS

Niezmiernie istotną cechą arkuszy CSS jest kaskadowość. Kaskada według specyfikacji technicznej tej technologii [strona 11] to algorytm, który dostarcza przeglądarce internetowej informacje o tym, który konkretnie styl przypisać do danego elementu. Algorytm ten korzysta z następujących warstw określających hierarchię ważności poszczególnych reguł:

* Waga – kaskada sprawdza typ „konkurujących” stylów.
* Pochodzenie – sprawdzane jest miejsce, w którym dane reguły CSS zostały zdefiniowane. W kolejności od najważniejszego do najmniej ważnego:
  + strona internetowa,
  + użytkownik,
  + przeglądarka internetowa.
* Specyficzność – odwołuje się do wagi konkretnego selektora. W kolejności do najważniejszego do najmniej ważnego:
  + style zadeklarowane bezpośrednio w atrybucie elementy HTML,
  + style przypisane do elementu za pomocą identyfikatora,
  + selektor klasy/ pseudoklasy/ atrybut,
  + tag/ pseudoelement.
* Pozycja – czyli kolejność zdefiniowania reguły CSS w kodzie źródłowym.

Istotnym pojęciem odwołującym się do wyglądu strony internetowej jest także responsywność. Pojęcie to pochodzi ze skrótowca zdefiniowanego w języku angielskim RWD (ang. *Responsive Web Design*) i oznacza technikę projektowania strony w taki sposób, aby jej wygląd automatycznie dostosowywał się do rozmiaru okna przeglądarki, na której ta witryna jest wyświetlana. Arkusze stylów CSS także pomagają osiągnąć programistom odpowiednią responsywność projektowanej strony.

**JavaScript**

Język programowania JavaScript oraz jego cechy zostały omówione w poprzednim podrozdziale. Autorzy niniejszego opracowania, sporządzając kod źródłowy odpowiadający za stronę klienta posługiwali się głównie biblioteką programistyczną React.JS, która oczywiście bazuje na języku programowania JavaScript.

**React.JS**

Jest to jedna z najpopularniejszych bibliotek języka programowania JavaScript. React.JS wykorzystywany jest przez programistów do tworzenia graficznych interfejsów użytkownika aplikacji internetowych.

Oprawa graficzna strony internetowej zawiera dane, które przekazywane są do nadrzędnego komponentu biblioteki React.JS, który z kolei jest odpowiedzialny za odpowiednie przetworzenie dostarczonych mu informacji w postać dokumentu HTML. Natomiast taki dokument języka HTML interpretowany jest przez przeglądarkę internetową i wyświetlany użytkownikowi w postaci strony internetowej.[React and React Native book].

W uproszeniu można więc stwierdzić, iż React.JS to technologia służąca do graficznego przedstawienia treści zapisanej w postaci cyfrowej, co w nomenklaturze informatycznej oznacza renderowanie.

Ważną cechą biblioteki programistycznej React.JS jest prostota jej użytkowania. W porównaniu do skomplikowanych szkieletów programistycznych nie ma tu potrzebny z zaznajamianiem się z ogromem mechanizmów stanowiących daną technologię. Biblioteka React.JS składa się z dwóch głównych interfejsów programowania aplikacji (ang. *API – Application programming interface*):

* React DOM – interfejs ten służy do bieżącego przedstawiania treści aplikacji internetowej,
* komponenty tworzone za pomocą biblioteki React.JS – są to elementy przetwarzane przez pierwszy z wymienionych interfejsów.

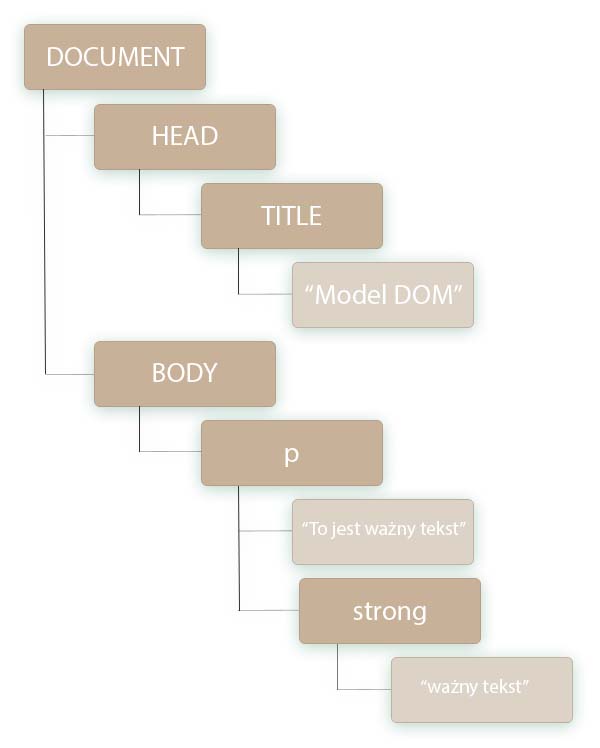
Komponenty obecne w kodzie źródłowym tworzonej aplikacji internetowej zawierają następujące składniki:

* dane pochodzące z określonych źródeł, , np. z bazy danych i przetwarzane przez komponent,
* metody zaimplementowane przez programistów, które reagują na zmiany zachodzące w danym komponencie,
* funkcje, które obsługują interakcję użytkownika aplikacji internetowej z danym elementem na stronie, który reprezentowany jest w kodzie źródłowym przez dany komponent,
* kod źródłowy w formacie JSX (ang. *JavaScript XML*).

Wspomniany wyżej JSX jest to format zapisu kodu HTML wewnątrz języka programowania JavaScript. Składnia ta świadczy o deklaratywnym charakterze biblioteki React.JS. Twórcy oprogramowania nie opisują szczegółowej sekwencji kroków, jaką powinien wykonać program, lecz warunki, jakie powinien spełniać projektowany system informatyczny. Taki sposób sporządzania kodu źródłowego jest bardzo wydajny i pozwala zapobiec wielu błędom, szczególnie jeżeli chodzi o tworzenie graficznego interfejsu użytkownika aplikacji.

Kolejnym, niezmiernie znaczącym elementem biblioteki React.JS jest wirtualny DOM (ang. *Document Object Model*). DOM czyli obiektowy model dokumentu to sposób przedstawienia dokumentów HTML oraz XML w postaci modelu obiektowego (Rys. 4.5.). DOM traktuje każdy znacznik HTML wraz z jego atrybutem i tekstem zawartym w nim, jako osobny obiekt, nazywany także węzłem. Taka forma przechowywania struktury strony internetowej pozwala programistom na odwołanie się do dowolnego węzła (obiektu) znajdującego się w DOM i zarządzanie nim.

Biblioteka React.JS przechowuje w pamięci reprezentację wszystkich elementów DOM, dzięki czemu za każdym razem, gdy dany komponent ma zostać zaktualizowany, React.JS porównuje aktualny stan (widoczny obecnie na stronie internetowej) i właściwości komponentu z jego przyszłą wersją, która dopiero zostanie wyświetlona użytkownikowi. Taki zabieg porównania ma na celu odpowiednią optymalizację wydajności mechanizmów aktualizujących zawartość witryny internetowej widzianej przez internautę.



Rys. 4.5. Graficzna reprezentacja modelu DOM [źródło: Document Object Model - DOM - Medianauka.pl]

# Projekt systemu zarządzania wizytami gabinetu stomatologicznego

## Procesy biznesowe

Pierwszym etapem projektowania dowolnego systemu komputerowego jest w wielu przypadkach określenie odpowiednich procesów, obiektów oraz aktorów biznesowych. Zdefiniowanie tych elementów biznesowych pozwala na lepsze zrozumienie zasad funkcjonowania projektowanej aplikacji zarówno osobom technicznym jak i nietechnicznym.

### Obiekty biznesowe

Obiektem biznesowym, który zawiera najwięcej informacji jest wizyta lekarska. Zawiera ona następujące dane:

* nazwa specjalizacji w ramach której przeprowadzana jest wizyta,
* opis wykonywanego zabiegu,
* dane osobowe lekarza przeprowadzającego wizytę,
* data i godzina przeprowadzenia wizyty,
* dane osobowe oraz kontaktowe pacjenta rezerwującego wizytę,
* wysokość opłaty za wykonany zabieg,
* status wizyty (zaplanowana lub zarchiwizowana).

Oczywiście powyższe dane pobierane są z informacji zawartych w innych obiektach biznesowych przedstawionych poniżej.

Użytkownik – obiekt ten powinien przechowywać dane osobowe oraz dane logowania zarejestrowanego użytkownika, niezależnie od jego roli w systemie.

Specjalizacja – obiekt zawierający nazwę specjalizacji oraz wszystkie usługi świadczone przez klinikę, będące w jej zakresie.

Rola w systemie – element ten określa rolę jaką pełni dany użytkownik w systemie.

Dyspozycyjność lekarza – obiekt, który powinien zawierać dane lekarza, jego specjalizację oraz zakres godzin, w których lekarz wyrazi swoją dyspozycyjność.

### Aktorzy biznesowi

W projektowanym systemie informatycznym autorzy aplikacji wyróżnili na etapie projektowania trzech aktorów biznesowych:

* pacjenta,
* lekarza,
* administratora.

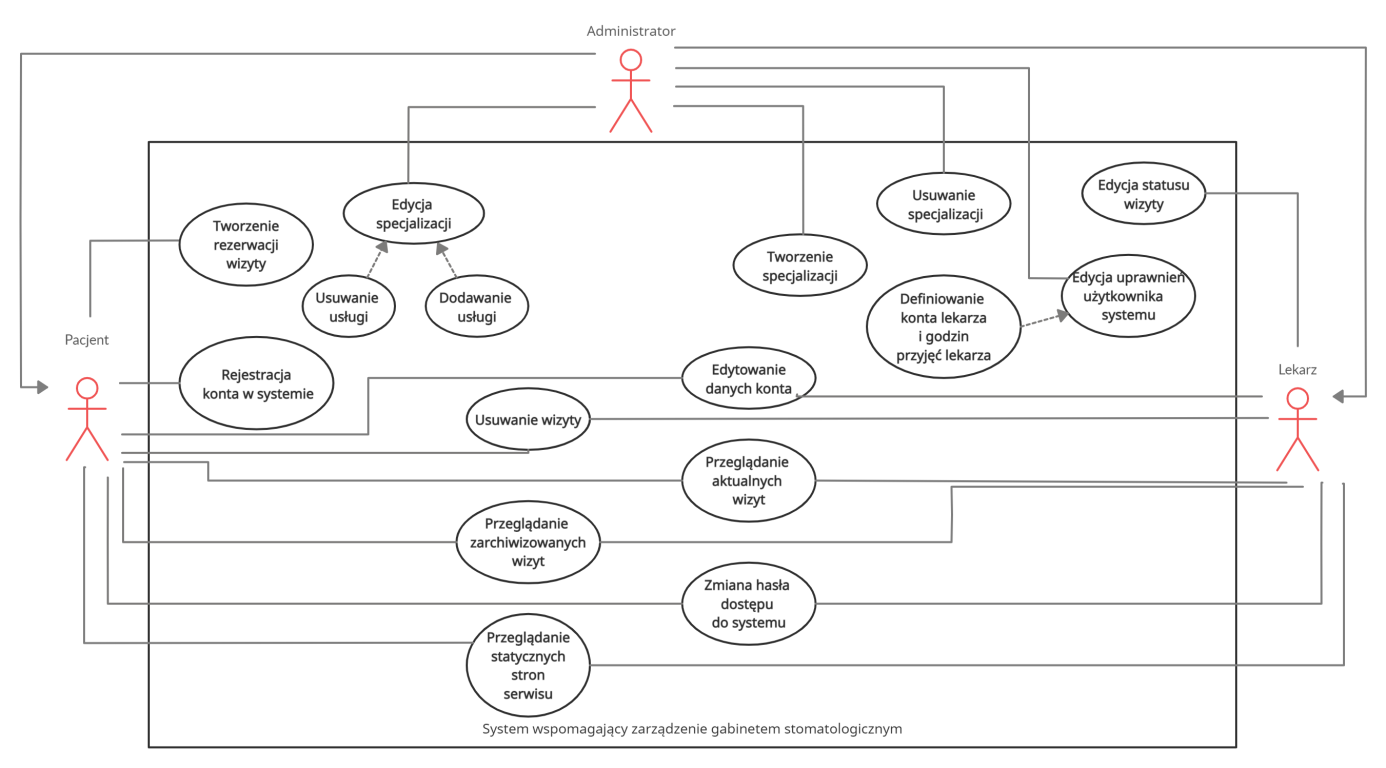
Pacjent to osoba korzystająca z systemu rezerwacji wizyt i nie należy on do personelu kliniki stomatologicznej, lecz wchodzi w interakcję z nią.

Lekarz natomiast świadczy specjalistyczne usługi na rzecz przychodni. Jednakże jak powszechnie wiadomo, wielu stomatologów przeprowadza wizyty w wielu gabinetach, więc nie można jednoznacznie stwierdzić czy jest on ściśle związany z daną placówką medyczną.

Administrator to personel pracujący na stanowiskach rejestratora bądź rejestratorki na recepcji kliniki stomatologicznej. Odpowiada on za prawidłowe deklarowanie dyspozycyjności danego lekarza w systemie, a także ma obowiązek rezerwować w systemie wizyty pacjentów.

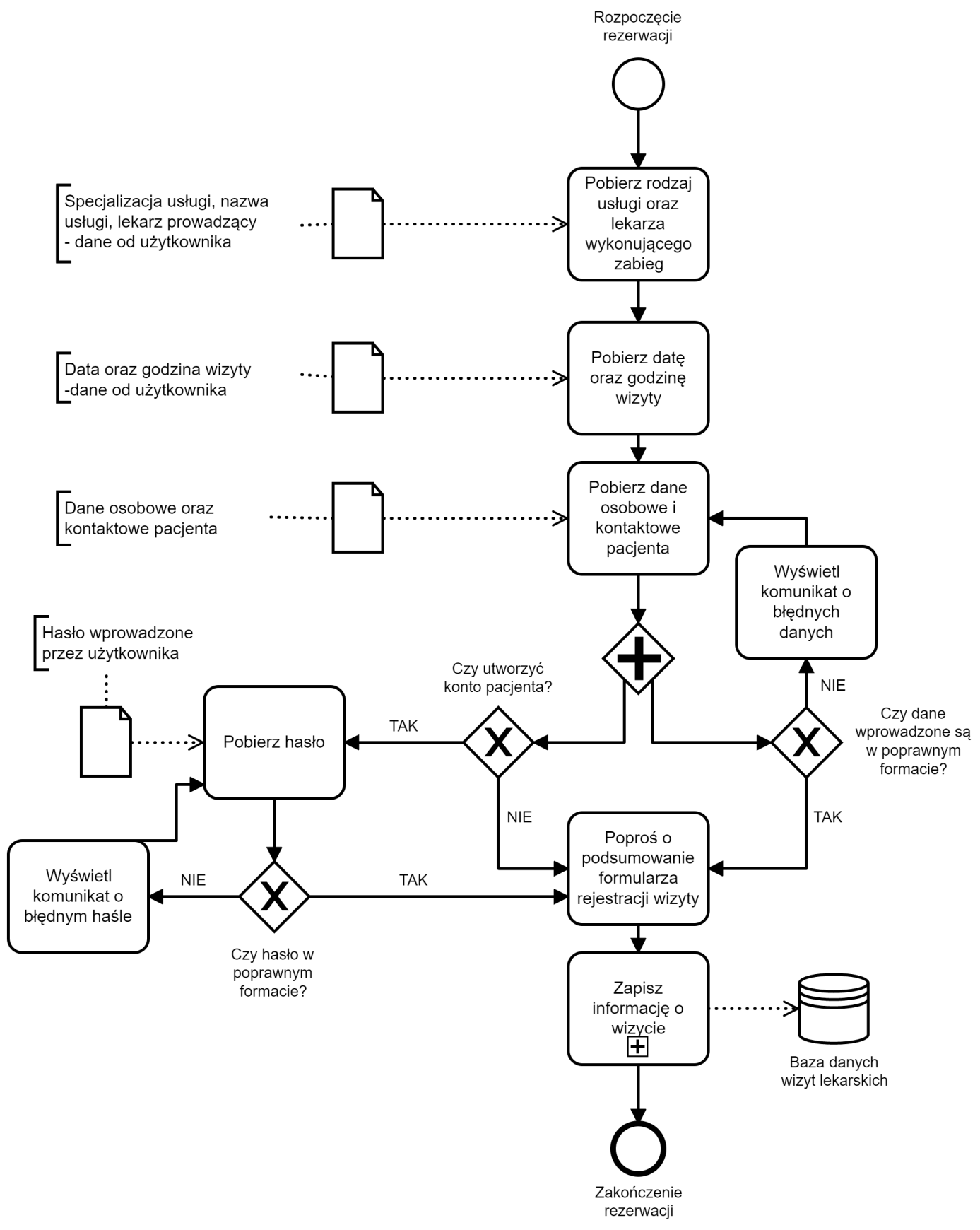
### Procesy biznesowe

Poniższy rysunek (Rys. 5.1.) przedstawia diagram przypadków użycia projektowanej aplikacji internetowej. Uwidocznione są na nim czynności realizowane przez użytkowników systemu o różnych uprawnieniach. Jak widać, najwięcej uprawnień posiada administrator, ponieważ ma możliwość korzystania z funkcjonalności dostępnej lekarzowi i pacjentowi, dodatkowo rozszerzonej o takie procesy jak np. edycja uprawnień użytkowników systemu.



Rys. 5.1. Diagram przypadków użycia

Spośród widocznych na powyższym diagramie, najistotniejszym procesem biznesowym z perspektywy funkcjonowania aplikacji internetowej jest tworzenie rezerwacji wizyty. Operacja ta może być wykonywana zarówno przez pacjenta (zalogowanego lub nie) jak i przez administratora. Graficzna prezentacja tego procesu została ukazana na diagramie BPMN (ang. *Business Process Model And Notation*) przedstawionym poniżej (Rys. 5.2.).



Rys. 5.2. Diagram BPMN przedstawiający proces rezerwacji wizyty

## Specyfikacja wymagań

W poniższym podrozdziale przedstawione zostały najważniejsze wymagania funkcjonalne oraz niefunkcjonalne projektowanego systemu.

### Wymagania funkcjonalne

Projektowana przez autorów niniejszego opracowania aplikacja internetowa stanowi system wspomagający pracę gabinetu stomatologicznego. Według założeń projektowych, najważniejszym i głównym elementem systemu powinien być moduł rezerwacji wizyty stomatologicznej. Tak utworzona wizyta powinna być określona za pomocą następujących informacji:

* rodzaj przeprowadzanego zabiegu,
* lekarz przeprowadzający zabieg,
* dane pacjenta,
* data i godzina wizyty,
* należność za przeprowadzoną usługę.

W obrębie projektowanego systemu wyróżnione zostały 3 rodzaje użytkowników, z których każdy posiada inne prawa:

* pacjent,
* lekarz,
* administrator.

Każdy zalogowany w systemie użytkownik powinien mieć także możliwość zmiany swoich danych lub usunięcia konta.

Natomiast zarówno użytkownicy niezalogowani jak i zalogowani, powinni mieć dostęp do statycznych modułów aplikacji internetowej, które dostarczały będą podstawowych informacji na temat funkcjonowania kliniki.

### Wymagania niefunkcjonalne

Wymienione poniżej wymagania niefunkcjonalne dotyczą projektowanego systemu jako całości, a nie poszczególnych cech systemu.

* Dostępność oraz niezawodność – projektowana aplikacja internetowa powinna być dostępna dla użytkowników 24 godziny na dobę, przez 365 dni w roku. Jednakże jak w przypadku każdego systemu informatycznego, należałoby przewidzieć odpowiednie prace konserwacyjne.
* Wydajność – poprzez zastosowanie biblioteki programistycznej React.JS witryna internetowa nie wymaga przeładowania dzięki czemu użytkownik uzyskuje natychmiastowy dostęp do poszczególnych funkcjonalności.
* Wsparcie techniczne – w obrębie systemu przewidziano rolę administratora, który może zmieniać dane innych użytkowników, jeżeli zajdzie taka potrzeba.
* Bezpieczeństwo – w celu uzyskania dostępu do swojego konta, użytkownik podczas tworzenia go zobowiązany jest podać hasło składające się z minimum ośmiu liter (w tym jednej wielkiej litery) oraz jednej cyfry i jednego znaku.
* Użyteczność – interfejs graficzny aplikacji został zaprojektowany w taki sposób, aby można było z niego korzystać na urządzeniach o różnej wielkości ekranu. Autorzy niniejszego opracowania starali się dobrać wielkość elementów interfejsu tak, aby był on ergonomiczny i użyteczny zarówno na ekranach telefonów komórkowych, laptopach i komputerach stacjonarnych. Istotnym był także dobór odpowiednich kolorów a przede wszystkim zapewnienie kontrastu pomiędzy tekstem a tłem poszczególnych elementów.

## Analiza wymagań

### Statyczne elementy aplikacji

Strona główna – użytkownik powinien mieć możliwość bezpośredniego przejścia do formularza rezerwacji wizyty stomatologicznej, może też się zalogować lub przejść do jednej z sekcji opisanych poniżej. Bardzo istotnym elementem obecnym na wielu stronach głównych jest też poziomy pasek znajdujący się u szczytu strony. Zawiera on podstawowe informacje takie jak numer telefonu do rejestracji oraz adres pod którym znajduje się gabinet kliniki.

Sekcja „o nas” – internauta powinien mieć dostęp do podstawowych informacji na temat placówki medycznej. Część ta podzielona zostanie docelowo na 3 panele:

* pierwszy zawierać będzie w większości elementy tekstowe, służące także optymalizację strony internetowej dla wyszukiwarek internetowych,
* panel „Nasz zespół” powinien przedstawiać poszczególnych specjalistów oferujących swe usługi w placówce medycznej oraz podstawowe informacje o nich,
* dział o nazwie „Gabinet” będzie zawierał galerię zdjęć przedstawiających przyrządy oraz pomieszczenia placówki medycznej, która korzystałaby z zaprojektowanej witryny internetowej.

Sekcja „oferta” będzie docelowo przedstawiać użytkownikowi rodzaj badań oraz usług świadczonych przez gabinet stomatologiczny. W celu zapewnienia odpowiedniej przejrzystości interfejsu, należałoby pogrupować poszczególne usługi według specjalizacji stomatologicznych, w zakresie których są wykonywane.

Sekcja „cennik” powinna przedstawiać nazwy poszczególnych zabiegów wraz z ich ceną. Jawne podanie cen za poszczególne usługi ma na celu zapewnienie potencjalnym klientom podstawowych informacji o kosztach leczenia, tak aby nie musieli oni w tym celu kontaktować się z rejestracją przychodni.

Sekcja „kontakt” będzie w przejrzysty sposób dostarczać informacji na temat godzin otwarcia placówki, adresu pod którym przychodnia świadczy swoje usługi, a także dane kontaktowe takie jak e-mail, numer telefonu komórkowego lub domowego. Dodatkowo dobrym rozwiązaniem jest umieszczenie w tej sekcji także formularza kontaktowego.

### Panel lekarza

Dostępny po zalogowaniu przez stomatologa panel użytkownika powinien zawierać następujące elementy.

* Grafik – w formie graficznego kalendarza, z którego stomatolog może wybrać konkretną datę, po czym na ekranie pojawi się lista wizyt zaplanowana na ten konkretny dzień. Z wyświetlonej listy przyszłych wizyt lekarz może wybrać dowolną i przenieść ją do archiwum (wizyt przeszłych) lub całkowicie usunąć wizytę z systemu.
* Historia wizyt – sekcja w której w formie listy (z możliwością sortowania według daty, godziny i rodzaju usługi) przechowywane są wizyty oznaczone jako zarchiwizowane. Lekarz powinien mieć możliwość usunąć taką wizytę lub przywrócić ją do wizyt bieżących.
* Ustawienia – panel w którym lekarz ma możliwość zmiany swoich danych takich jak numer telefonu kontaktowego, adresu e-mail, a nawet hasła po podaniu swojego poprzedniego hasła dostępu do systemu.

### Panel pacjenta

Użytkownik zalogowany w systemie jako pacjent, powinien mieć możliwość zarezerwowania wizyty na konkretny zabieg bez konieczności powtórnego wprowadzania swoich danych. W takim wypadku dane potrzebne do utworzenia wizyty powinny być pobierane z bazy danych.

Istotne jest także zapewnienie pacjentowi dostępu do informacji na temat jego aktualnych (nadchodzących) wizyt oraz do zabiegów ubiegłych (zarchiwizowanych). W sekcji „aktualne wizyty” pacjent ma możliwość usunięcia danej wizyty.

Każdy kuracjusz powinien także mieć możliwość zmiany swoich danych osobowych oraz zmiany hasła dostępu do systemu po uprzedniej weryfikacji hasła poprzedniego. W tej sekcji powinna znajdować się także najbliższa nadchodząca wizyta, zamówiona przez pacjenta.

### Panel administratora

Konto z uprawnieniami administratora projektowanego systemu powinno być obsługiwane przez osobę lub osoby pracujące na recepcji. Z poziomu tego panelu możliwe byłoby tworzenie wizyt stomatologicznych dla zarejestrowanych oraz niezarejestrowanych użytkowników witryny internetowej. Poprzez takie zaprojektowanie systemu wszystkie wyżyty odbywające się w klinice byłby by zawarte w bazie danych (nawet te rejestrowane drogą telefoniczną).

Kolejną funkcjonalnością modułu omawianego w bieżącym podrozdziale jest wgląd we wszystkie wizyty, zarówno przyszłe jak i zarchiwizowane, odbywające się w przychodni stomatologicznej. Administrator powinien mieć także możliwość zmiany statusu wizyty oraz ewentualnego usunięcia jej z systemu.

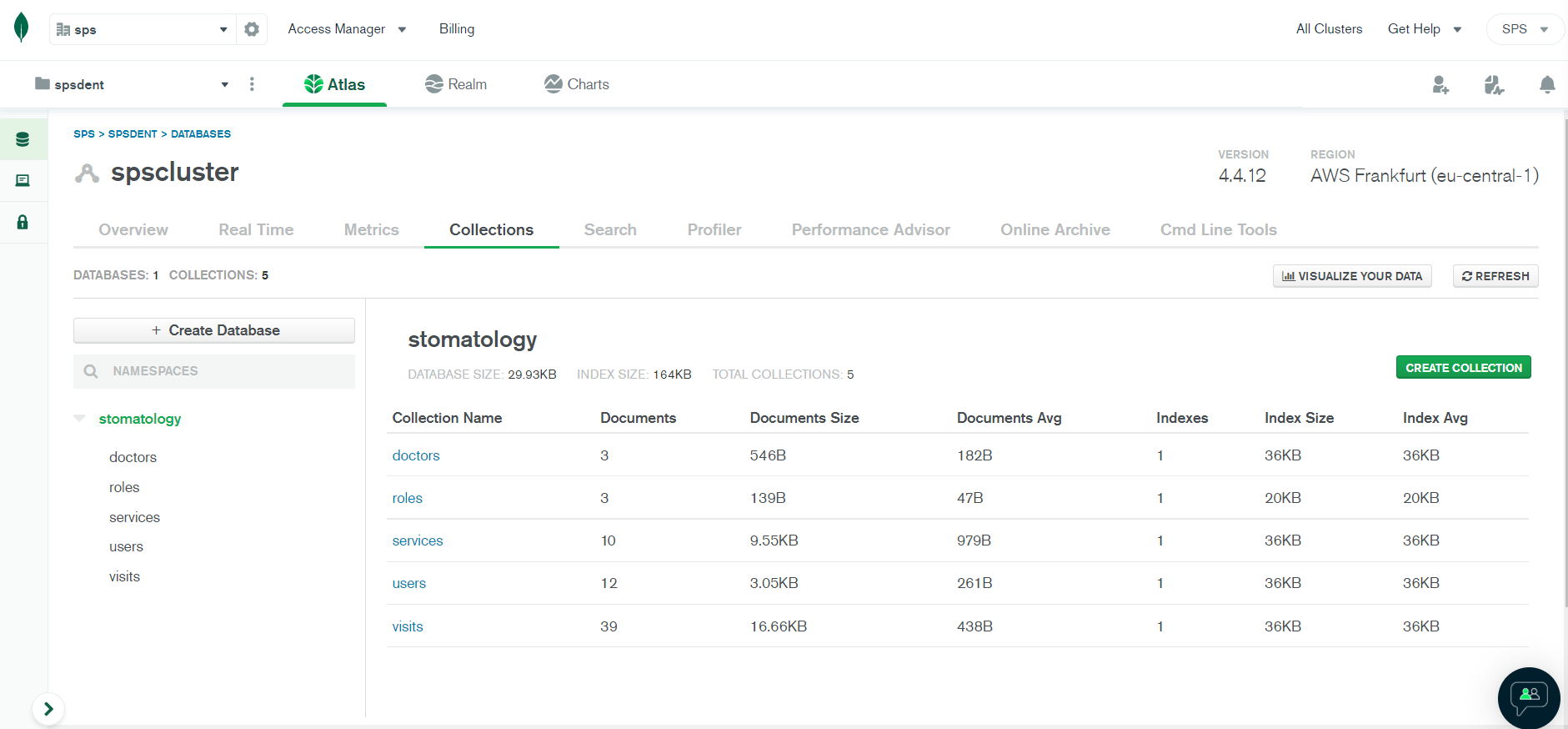
Rejestrator lub rejestratorka medyczna mająca dostęp do modułu administratora będzie także odpowiedzialna za tworzenie kont stomatologom oraz ustalanie każdemu z nich godzin przyjęć w gabinetach. Taki sposób funkcjonowania systemu odciąży lekarzy specjalistów z obowiązku zarządzania swoim grafikiem i pozwoli im w całości skupić się na wykonywanej pracy.

Dodatkowo w przypadku ewentualnego poszerzenia lub zawężenia zakresu usług świadczonych przez przychodnię stomatologiczną należałoby zapewnić także możliwość zarządzania usługami dostępnymi dla pacjentów korzystających z systemu.

## Projekt struktury bazy danych

Zastosowana baza danych w projekcie systemu będącego przedmiotem niniejszego opracowania oparta jest na modelu obiektowym, a nie relacyjnym. Wobec tego w strukturze bazodanowej brak tabel i powiązań między nimi. Zamiast tego w bazie danych znajdują się kolekcje, które przechowują dokumenty (obiekty) reprezentujące odpowiednie elementy systemu.

Na poniższym rysunku poniżej (Rys. 5.2.) przedstawiony został system zarządzania bazą danych o nazwie MongoDB Atlas. Jest to rozbudowane narzędzie oparte na technologii chmurowej, które pozwala programistom w pełni zarządzać obiektową bazą danych jaką jest MongoDB. Autorzy niniejszego opracowania zdecydowali się na użycie dokładnie tej technologii ze względu na przejrzystość graficznego interfejsu użytkownika.



Rys. 5.3.Zdjęcie ekranu przedstawiające główny panel systemu zarządzania bazą danych MongoDB Atlas

### Struktura bazy danych

Jak już wymieniono wcześniej, struktura obiektowej bazy danych opiera się na kolekcjach (zbiorach) obiektów. Każdy zbiór ma za zadanie przechowywać obiekty (nazywane też dokumentami) o tożsamej charakterystyce i właściwościach przechowywanych w formacie JSON (ang. *JavaScript Object Notation*). W projektowanej aplikacji internetowej przewidziano 6 rodzajów obiektów, wobec czego w bazie danych znajduje się 6 kolekcji opisanych w kolejnych akapitach.

„Doctors” – zbiór obiektów, które reprezentują dyspozycyjność danego stomatologa. Przechowują unikalny identyfikator lekarza, jego specjalizacji oraz godziny pracy w których wykonuje on zabiegi.



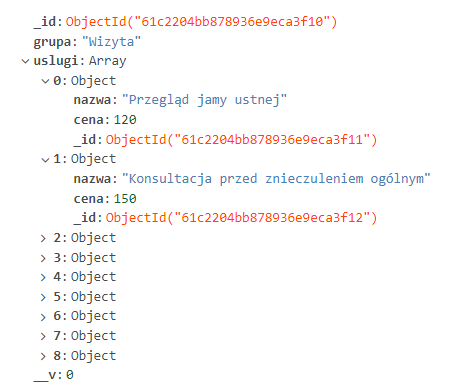
Rys. 5.4. Reprezentacja przykładowego obiektu kolekcji „Doctors”

„Roles” – zbiór zwierający dokładnie 3 elementy, które reprezentują role zdefiniowanych użytkowników systemu (pacjent, lekarz, administrator). Obiekty przechowywane w tej kolekcji zawierają unikalny identyfikator oraz nazwę określającą rolę, a więc i uprawnienia użytkownika systemie.



Rys. 5.5. Zawartość kolekcji „Roles”

„Services” – zbiór obiektów reprezentujących poszczególne specjalizacje lekarzy przyjmujących w przychodni. Każdy z obiektów posiada unikalny identyfikator, nazwę specjalności, którą przedstawiają, a także tablicę zawierającą konkretne usługi świadczone w ramach danej specjalizacji. Usługi te także reprezentowane są przez obiekty, z których każdy posiada identyfikator, nazwę oraz cenę danego świadczenia.



Rys. 5.6. Reprezentacja przykładowego dokumentu zbioru „Services”

„Users” to kolekcja, która przechowuje obiekty konkretnych użytkowników systemu. Poszczególne dokumenty (bo tak też można nazywać obiekty w zastosowanym systemie bazodanowym) zawierają dane osobowe oraz kontaktowe użytkownika takie jak:

* imię,
* nazwisko,
* adres e-mail,
* numer telefonu kontaktowego,
* dane adresowe takie jak miasto, ulica i kod pocztowy,
* hasło dostępu do serwisu w formie zaszyfrowanej,
* określenie roli użytkownika w systemie, w formie identyfikatora odnoszącego się do konkretnego obiektu w kolekcji „Roles”.



Rys. 5.7. Reprezentacja przykładowego elementu kolekcji „Users”

„Visits” – zbiór przechowujący dokumenty reprezentujące zarezerwowaną wizytę stomatologiczną. Każdy z obiektów znajdujących się w tejże kolekcji zawiera następujący zestaw informacji określających wizytę lekarską:

* specjalizacja,
* nazwa zabiegu,
* imię i nazwisko lekarza,
* data i godzina wykonywania zabiegu,
* dane osobowe i kontaktowe pacjenta,
* cena danej usługi,
* datę i czas utworzenia oraz ostatniej edycji obiektu reprezentującego wizytę.



Rys. 5.8. Przykładowy dokument kolekcji „Visits”

### Role i uprawnienia użytkowników

Zgodnie z zaprojektowaną bazą danych, w systemie znajdują się 3 rodzaje ról, które określają uprawnienia do korzystania z poszczególnych funkcjonalności systemu:

* administrator,
* lekarz,
* pacjent.

. Autorzy niniejszego opracowania dołożyli wszelkich starań aby sprostać wymaganiom opisanym w podrozdziałach 5.2.1. oraz 5.3. Oznacza to, iż najszerszy zakres uprawnień został nadany administratorowi, który przede wszystkim może tworzyć wizyty stomatologiczne dla zarejestrowanych bądź niezarejestrowanych w systemie pacjentów, edytować uprawnienia oraz dane użytkowników, edytować specjalizacje oraz świadczone w ich ramach usługi.

Użytkownik systemu posiadający uprawnienia pacjenta ma natomiast możliwość zarejestrowania wizyty lekarskiej oraz jej usunięcie.

Lekarze mają z kolei możliwość zmiany statusu wizyty (w przypadku projektowanej aplikacji internetowej jest to przeniesienie jej do archiwum) lub jej usunięcia. Podstawową funkcjonalnością panelu lekarza jest także dostęp do grafiku wizyt.

Użytkownicy niezarejestrowani w projektowanym systemie, podczas korzystania z witryny internetowej mają dostęp do statycznych elementów strony internetowej takich jak cennik, kontakt, oferta kliniki i strona główna. Oczywiście Ci użytkownicy mają także możliwość zarezerwowania wizyty lekarskiej oraz utworzenia konta pacjenta w systemie.

Wszyscy zarejestrowani w systemie użytkownicy, niezależnie od roli i zakresu uprawnień mają możliwość zmiany swoich danych osobowych oraz hasła. Są oni również w stanie całkowicie usunąć swoje konto z projektowanego systemu.

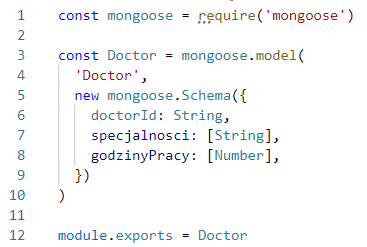
Użytkownicy o prawach pacjenta oraz lekarza po zalogowaniu się, zostaną przekierowani do panelu zarządzania swoim kontem, w którym to zawarta też będzie informacja o najbliższej zaplanowanej wizycie z ich udziałem.

# Implementacja

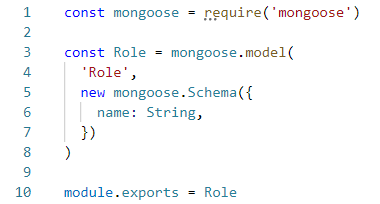
## Baza danych

Wykorzystana w projektowanej aplikacji internetowej baza danych Mongo DB jest obiektową bazą danych. Jak już wcześniej wspomniano, w tego typu magazynie informacji nie definiuje się konkretnych tabel, ponieważ dane przechowywane są w kolekcjach (zbiorach) w postaci dokumentów w formacie JSON. Szkielety, według których powinny być tworzone dokumenty przechowywane w bazie danych, zostały zaimplementowane w plikach odpowiadających nazwom kolekcji w folderze o nazwie *models*, co przedstawiają poniższe listingi (Listing 6.1., Listing 6.2., Listing 6.3., Listing 6.4., Listing 6.5.). Wobec powyższego stwierdzić można, iż struktura (model) obiektu przechowywanego w danej kolekcji jest definiowana w kodzie źródłowym aplikacji, a nie bezpośrednio w systemie bazodanowym.

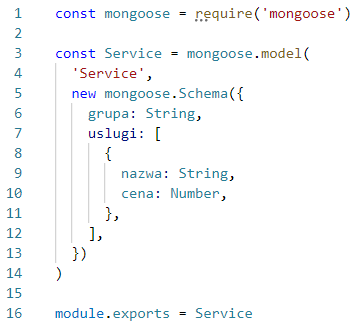
Listing 6.1. Implementacja dokumentu reprezentującego dyspozycyjność danego stomatologa



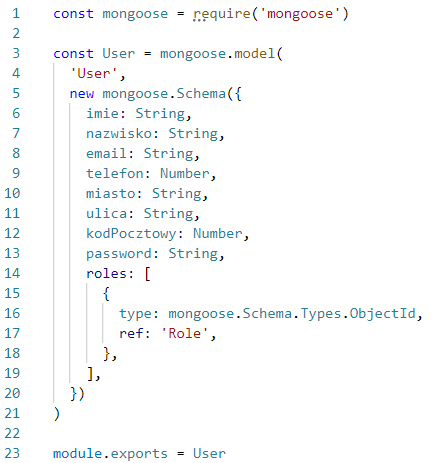
Listing 6.2. Implementacja dokumentu reprezentującego rolę danego użytkownika w systemie



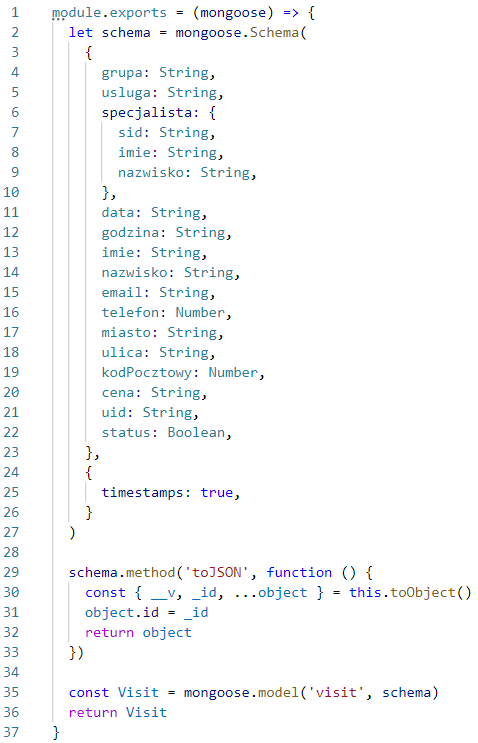
Listing 6.3. Implementacja obiektu reprezentującego specjalizację lekarską



Listing 6.4. Implementacja dokumentu przedstawiającego użytkownika zarejestrowanego w systemie



Listing 6.5. Implementacja obiektu reprezentującego wizytę lekarską utworzoną w systemie



Utworzenie reprezentacji bazy danych w kodzie źródłowym i zespojenie wszystkich utworzonych modeli w obiekt o nazwie *db* zostało przedstawione na poniższym listingu.

Listing 6.6. Zawartość pliku *index.js* w folderze *models*



## Serwer aplikacji

Jednym z podstawowych zadań serwera projektowanej aplikacji internetowej jest obsługa operacji wykonywanych na danych przechowywanych w bazie danych. Są to operacje określane w nomenklaturze informatycznej jako CRUD (ang. c*reate, read, update, delete*). Skrótowiec ten oznacza podstawowe operacje implementowane w aplikacjach korzystających z bazy danych:

* umieszczenie nowych informacji w bazie danych (ang. *create*),
* odczyt lub wyświetlenie danych z bazy (ang. *read*),
* edycja istniejących już informacji (ang. *update*),
* usuwanie istniejących danych z bazy (ang. *delete*).

Wszystkie wyżej wymienione operacje na danych zostały zaimplementowane w kodzie źródłowym jako odpowiednie metody w folderze o nazwie *controllers*. Te funkcje, które odpowiadają za przeprowadzanie odpowiednich operacji na obiektach znajdujących się w bazie danych przechowywane są w plikach, które określić można mianem kontrolerów.

Dla każdej kolekcji przechowywanej w bazie danych został utworzony osobny plik (kontroler) o nazwie korespondującej z nazwą kolekcji. Taki zabieg ma na celu zwiększenie czytelności kodu źródłowego aplikacji oraz zachowanie odpowiedniej struktury plików, z których składa się kod systemu informatycznego.

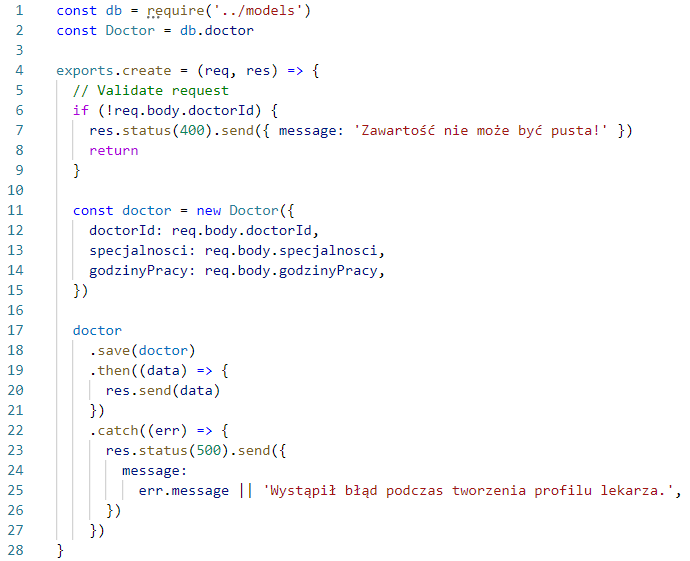
Na poniższych listingach (Listing 6.7., Listing 6.8., Listing 6.9., Listing 6.10., Listing 6.11) przedstawione zostały fragmenty kodu źródłowego odpowiedzialne za tworzenie, edycję, usuwanie lub odczytywanie informacji zawartych w bazie danych Mongo DB.

Listing 6.7. Implementacja kontrolera *role.js*



Powyższy listing ukazuje cały plik o nazwie *role.js*, w którym zdefiniowana została funkcja odczytująca obiekty z bazy danych reprezentujące trzy podstawowe role użytkowników w projektowanej aplikacji internetowej.

Listing 6.8. Fragment implementacji kontrolera o nazwie *doctor.js*



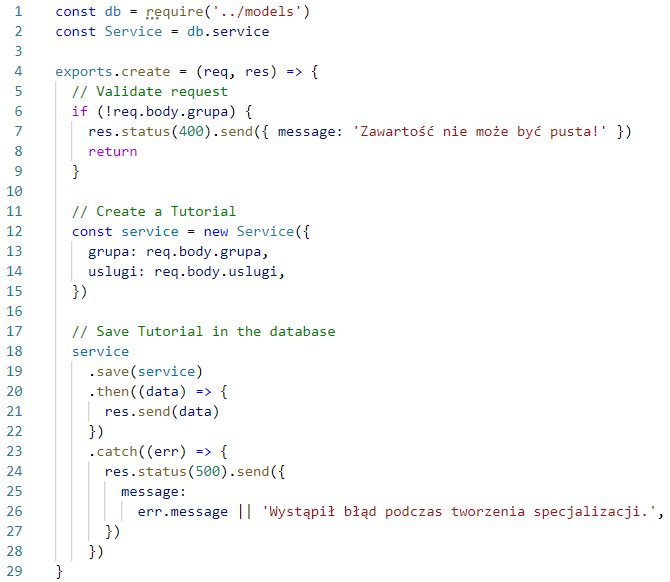
Powyższy listing przedstawia funkcję, która tworzy obiekt reprezentujący dyspozycyjności danego lekarza, a następnie zapisuje go w bazie danych (za pomocą funkcji save) w formacie dokumentu JSON.

Listing 6.9. Fragment implementacji kontrolera o nazwie *user.js*



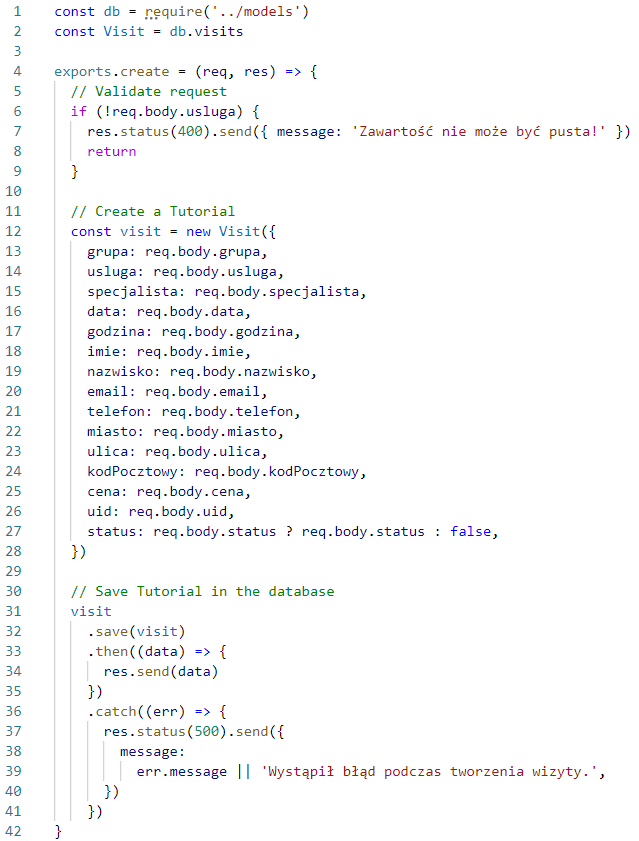
Powyższy fragment kontrolera o nazwie *user.js* zawiera implementację metody odpowiedzialnej za pobranie z bazy danych wszystkich użytkowników o konkretnym nazwisku, podanym jako parametr funkcji.

Listing 6.10. Fragment implementacji kontrolera o nazwie *service.js*



Przedstawiony powyżej listing zawiera implementację metody odpowiedzialnej za utworzenie i zapisanie w bazie danych nowej specjalizacji, wraz z usługami świadczonymi w jej zakresie.

Listing 6.11. Fragment implementacji pliku o nazwie *visit.js*

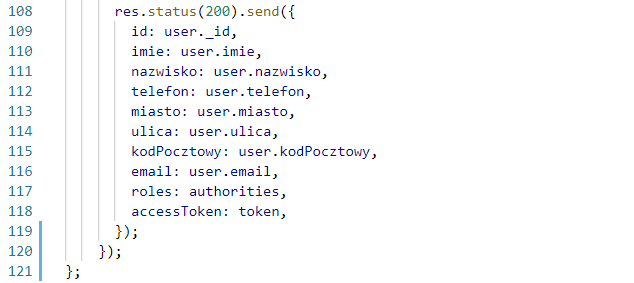
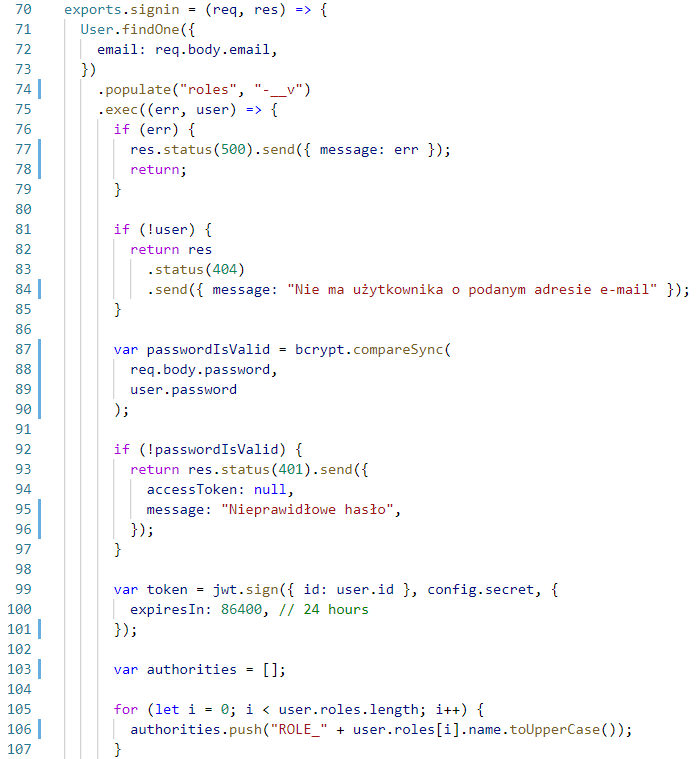


Ostatni z kontrolerów odpowiada za utworzenie obiektu reprezentującego wizytę lekarską oraz za zapisanie tejże wizyty do bazy danych Mongo DB.

W omawianym folderze zawierającym kontrolery znajduje się także plik o nazwie *auth.js* (Listing 6.12.), którego kod źródłowy odpowiada za obsługę autoryzacji następujących czynności:

* logowanie,
* rejestracja użytkownika,
* zmiana hasła dostępu do systemu.

Listing 6.12. Implementacja funkcji *singin* zawartej w pliku *auth.js*



NAPISAĆ O PLIKACH W FOLDERZE CONFIG, DO KONFIGURACJI POLACZENIA Z BAZA. POTEM NAPISAC O SAMYM SERVER.JS. NA KOŃCU NAPISAC O ROUTES.

## Aplikacja internetowa

### Funkcjonalność aplikacji dostępna dla niezalogowanego użytkownika

### Panel lekarza

### Panel pacjenta

### Panel administratora

# Testowanie

## Testy zabezpieczeń

## Testy funkcjonalne

## Testy niefunkcjonalne

# Podsumowanie

# Literatura