

Politechnika Wrocławska  
Wydział Elektroniki

---

PROJEKT INŻYNIERSKI  
System do ankietowania na stronie WWW

---

*Autor:*

MARCIN WINCEK 196099

Promotor:

dr inż. Roman Ptak

10.12.2014 r.

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>4</b>
1.1	Wprowadzenie . . . . .	4
1.2	Cel i zakres projektu . . . . .	4
<b>2</b>	<b>Analiza wymagań systemowych</b>	<b>5</b>
2.1	Przypadki użycia aplikacji . . . . .	5
2.1.1	Autor ankiety . . . . .	5
2.1.2	Respondent . . . . .	6
2.2	Wymagania funkcjonalne . . . . .	6
2.3	Wymagania niefunkcjonalne . . . . .	7
2.4	Przegląd innych rozwiązań . . . . .	8
2.4.1	Google Docs . . . . .	8
2.4.2	Ankietka.pl . . . . .	9
2.4.3	SurveyMonkey.com . . . . .	10
<b>3</b>	<b>Architektura systemu</b>	<b>12</b>
3.1	Organizacja aplikacji . . . . .	12
3.2	Struktura bazy danych . . . . .	14
<b>4</b>	<b>Implementacja</b>	<b>16</b>
4.1	Wykorzystane technologie . . . . .	16
4.1.1	Node.js . . . . .	16
4.1.2	MongoDB . . . . .	17
4.1.3	Mongoose . . . . .	18
4.1.4	Angular.JS . . . . .	19
4.1.5	Inne technologie . . . . .	20
4.2	Diagram klas . . . . .	20
4.3	API . . . . .	21
4.3.1	/api/surveys . . . . .	24
4.3.2	/api/surveys/{surveyId} . . . . .	24
4.3.3	/api/surveys/{surveyId}/activate . . . . .	25
4.3.4	/api/answers . . . . .	25
4.3.5	/api/answers/{surveyId} . . . . .	25
4.3.6	/api/code/{surveyCode} . . . . .	26
4.4	Komunikacja serwera HTTP z bazą danych . . . . .	26
4.4.1	MongoSurveyProvider . . . . .	27

4.4.2	MongoAnswerProvider . . . . .	28
4.4.3	MongoUserProvider . . . . .	30
<b>5</b>	<b>Testowanie</b>	<b>32</b>
5.1	Tworzenie nowej ankiety . . . . .	32
5.2	Edycja pytań ankiety . . . . .	33
5.3	Analiza zebranych wyników . . . . .	34
5.4	Kontrola jednokrotnego głosowania . . . . .	35
<b>6</b>	<b>Wdrożenie</b>	<b>36</b>
6.1	Node.js . . . . .	36
6.1.1	Windows . . . . .	36
6.1.2	Linux . . . . .	36
6.2	Nginx . . . . .	37
6.2.1	Windows . . . . .	37
6.2.2	Linux . . . . .	37
6.2.3	Konfiguracja serwera . . . . .	37
6.3	MongoDB . . . . .	39
6.3.1	Windows . . . . .	39
6.3.2	Linux . . . . .	41
6.4	Instalacja zależności projektowych . . . . .	42
6.5	Uruchomienie aplikacji . . . . .	42
<b>7</b>	<b>Instrukcja użytkownika</b>	<b>44</b>
7.1	Rejestracja w serwisie . . . . .	44
7.2	Tworzenie i edycja ankiety . . . . .	44
7.2.1	Udostępnianie ankiety . . . . .	46
7.2.2	Analiza wyników . . . . .	47
<b>8</b>	<b>Podsumowanie i wnioski</b>	<b>48</b>
8.1	Perspektywy rozwoju aplikacji . . . . .	48
8.2	Opis zawartości płyty CD . . . . .	48
	<b>Literatura</b>	<b>49</b>

# 1 Wstęp

## 1.1 Wprowadzenie

Wiele dziedzin przemysłu wykorzystuje badania opinii publicznej za pomocą ankiet celu podniesienia jakości swoich usług. Ankietowanie nie musi dotyczyć tylko przemysłu, może dotyczyć wielu innych dziedzin życia. Współczesne technologie informatyczne pozwalają zautomatyzować proces ankietowania przez zbieranie odpowiedzi do ankiet w bazie danych oraz automatyczną analizę danych. Pozwala to na znaczne usprawnienie tego procesu, a co za tym idzie - zyski finansowe, mniej wysiłku wymaganego od ankietującego, czy też większy komfort dla ankietowanego.

## 1.2 Cel i zakres projektu

Celem projektu jest stworzenie aplikacji internetowej pełniącej rolę systemu do ankietowania. Aplikacja będzie dostępna dla użytkowników w formie strony internetowej. System powinien spełniać podstawowe wymagania, tj.:

- tworzenie ankiet,
- zbieranie odpowiedzi od respondentów,
- analiza zebranych odpowiedzi.

Aplikacja dostarczy użytkownikom możliwości tworzenia i edycji ankiet oraz analizy ich wyników. System będzie generował dla każdej ankiety osobną stronę dla respondentów, a hiperłącze do tej strony będzie udostępniał dla użytkownika tworzącego ankietę. Projekt nie zakłada automatycznego rozsyłania linka do respondentów.

Na stronie udostępnianej respondentom wyświetlane będą pytania ankiety. Respondenci będą mieli możliwość udzielenia odpowiedzi na wszystkie pytania oraz wysłanie ich do systemu. Projekt zakłada, że każdy respondent będzie mógł wypełnić ankietę tylko jeden raz.

## 2 Analiza wymagań systemowych

### 2.1 Przypadki użycia aplikacji

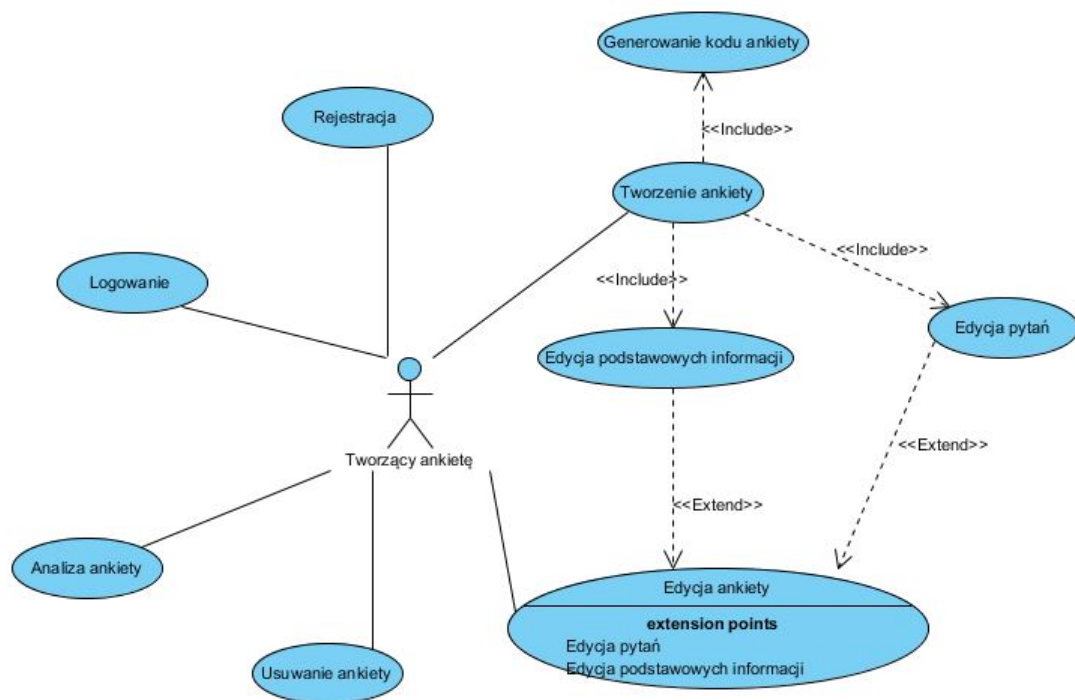
W systemie wyszczególniono dwa typy użytkowników:

- użytkownik tworzący ankietę,
- respondent.

Dla każdego z typów określono typowe przypadki użycia aplikacji.

#### 2.1.1 Autor ankiety

Rysunek 1. przedstawia przypadki użycia aplikacji przez autora ankiety.

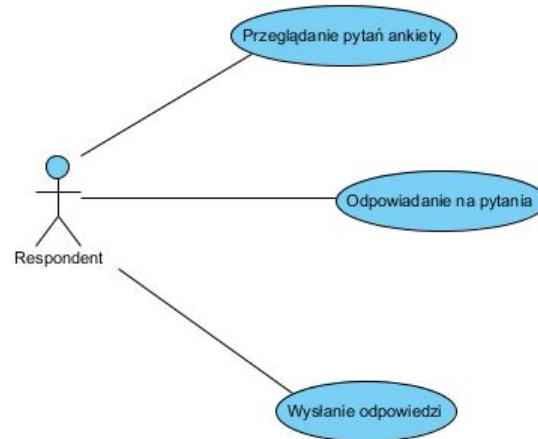


Rysunek 1: Diagram przypadków użycia dla twórcy ankiety

Użytkownik chcąc skorzystać z systemu musi wcześniej zalogować się do niego, a jeśli nie posiada jeszcze własnego konta, musi takowe stworzyć. Po zalogowaniu się do serwisu może korzystać ze wszystkich funkcjonalności. Najbardziej rozbudowanymi z nich jest tworzenie nowej ankiety oraz edycja już stworzonych. Użytkownik ma również możliwość usunięcia swojej ankiety z systemu.

### 2.1.2 Respondent

Rysunek 2. przedstawia przypadki użycia aplikacji przez respondenta.



Rysunek 2: Diagram przypadków użycia dla respondenta

Użytkownik udzielający odpowiedzi w ankiecie ma mniej możliwości w systemie. Jego zadaniem jest udzielenie odpowiedzi na pytania i wysłanie tych odpowiedzi na serwer.

## 2.2 Wymagania funkcjonalne

Analiza przypadków użycia pozwala wyselekcjonować funkcjonalności, jakie system powinien oferować użytkownikom. Wymagania funkcjonalne postawione przed systemem zostały przedstawione w formie tabeli. Kolejność zawartych w niej funkcjonalności jest chronologicznym odzwierciedleniem kolejności ich implementacji. O krytyczności danej funkcjonalności decyduje jej niezbędność w systemie. W przedstawionej tabeli charakterystyka krytyczności została opisana w skali czterostopniowej: wysoka, średnia, niska oraz dodatkowa. Wysokiej krytyczności są funkcjonalności będące absolutnym filarem działania systemu, zaś elementy dodatkowe nie wnoszą niczego do funkcjonalności, lecz pozwalają na wykonanie tych samych czynności w inny, bardziej komfortowy sposób. Tabela 1. przedstawia dokładną analizę wymagań funkcjonalnych.

Tabela 1: Spis wymagań funkcjonalnych

Opis	Krytyczność
Rejestracja kont użytkowników w systemie	Wysoka
Logowanie na konta użytkowników	Wysoka
Zmiana danych osobowych przez użytkowników	Dodatkowa
Tworzenie ankiety	Wysoka
Usuwanie ankiety	Średnia
Edycja pytań	Wysoka
Edycja podstawowych informacji ankiety	Średnia
Generowanie kodu ankiety	Wysoka
Możliwość oddawania głosów w ankiecie	Wysoka
Generowanie wykresów z wyników ankiety	Wysoka
Eksportowanie wyników ankiety do zewnętrznych plików	Dodatkowa
Pobieranie wyników ankiety w czasie rzeczywistym	Niska
Automatyczne rozsyłanie linków do ankiety na wskazane adresy email	Dodatkowa

## 2.3 Wymagania нефункциональные

Aplikacja została przetestowana i działała poprawnie na następujących przeglądarkach internetowych:

- Google Chrome 37,
- Mozilla Firefox 31,
- Opera 22,
- Internet Explorer 9.

Działanie aplikacji na niższych wersjach tych przeglądarek może (ale nie musi) być niezgodne z wymaganiami. Autor aplikacji nie gwarantuje poprawnego działania na przeglądarkach nie wymienionych powyżej. Wyjątkiem tutaj jest Internet Explorer, który od wersji 8 w dół nie jest wspierany.

Aplikacja powinna zapewniać bezpieczeństwo przetwarzanych danych, a szczególnie danych osobowych. W tym celu należy zabezpieczyć połączenie między klientem a serwerem za pomocą odpowiedniego protokołu (SSL).

Przeglądarka internetowa, za pomocą której użytkownik korzysta z aplikacji musi mieć włączoną obsługę JavaScript.

Aplikacja serwerowa powinna dać się uruchomić na maszynie z systemem operacyjnym Windows lub Linux (dystrybucje Ubuntu lub Debian).

## 2.4 Przegląd innych rozwiązań

W Internecie można znaleźć wiele rozwiązań podobnych do tworzonego systemu. Wiele z nich, oprócz podstawowych funkcjonalności, zapewnia również ciekawe dodatki, takie jak automatyczne powiadomienia dla autora o nowych odpowiedziach, wiele dodatkowych typów pytań, możliwość skorzystania z predefiniowanych pytań itp. Niestety, takie dodatkowe funkcjonalności są często dostępne po zapłaceniu dodatkowych opłat. Poniżej wymienione zostały przykłady aplikacji do ankietowania.

### 2.4.1 Google Docs

Google Docs to prawdopodobnie najpopularniejsze narzędzie wykorzystywane do przeprowadzania ankiet. Jest to rozwiązanie w pełni darmowe, do korzystania potrzebne jest tylko konto w serwisie Gmail. System pozwala w łatwy sposób stworzyć ankietę i udostępnić ją dla respondentów. Serwis Google Docs daje możliwość stworzenia ankiety składającej się z wielu pytań, istnieje również wiele możliwych typów pytań (dokładnie 9). Tworzenie przykładowej ankiety przedstawione zostało na Rysunku 3.

Sporą wadą tej aplikacji jest fakt, że po przeprowadzeniu ankiety wyniki dostarczane są w postaci surowego arkusza kalkulacyjnego. Dalsze kroki analizy ankiety, takie jak wykresy itp., trzeba przeprowadzić samemu.

#### Zalety

- Rozwiązanie darmowe
- Łatwość stworzenia ankiety
- Wiele typów pytań

#### Wady

- Słaba analiza wyników ankiety.



Page 1 of 1

## Ocena produktu

Form Description

Question Title: Jak podoba Ci się produkt?

Help Text:

Question Type: Multiple choice ☐ Go to page based on answer

Options:

- ☐ Super
- ☐ Dobry
- ☐ Słaby
- ☐ Click to add option

or Add "Other"

Advanced settings

Done ☒ Required question

Rysunek 3: Okno tworzenia ankiety w Google Docs

### 2.4.2 Ankietka.pl

Ankietka.pl jest serwisem polskim, prawdopodobnie najpopularniejszym w naszym kraju. W przeciwieństwie do Google Docs jest tylko częściowo darmowy. W darmowej wersji serwisu napotkać można znaczące ograniczenia, np. limit respondentów per ankietę (30). Dodatkowo korzystając z darmowej wersji nasza ankietę będzie umieszczona w publicznie dostępnej Bazie ankiet, co może być niepożądane.

Decydując się jednak na płatną wersję (najtańsze rozwiązanie - 39 zł/msc) mamy do dyspozycji wiele przydatnych funkcji, np. eksport wyników do zewnętrznych plików, ukrycie ankiety w Bazie ankiet, brak reklam w serwisie.

Rysunek 4. przedstawia przykładową analizę odpowiedzi do ankiety.

### Zalety

- Wiele typów pytań
- Rozwinięta analiza odpowiedzi (wykresy)
- Możliwość eksportu do pliku

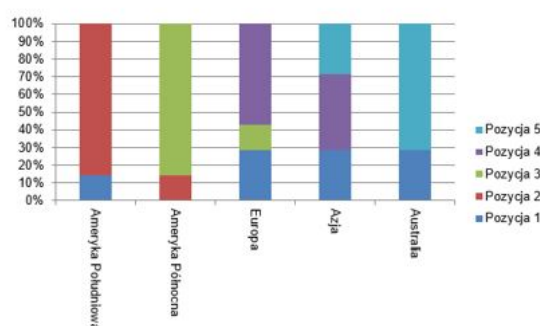
## Wady

- Ograniczenia w darmowej wersji
- Zapisywanie ankiety do bazy ankiet
- Pełna funkcjonalność po zapłaceniu (min. 39 zł/msc)

### 9. Bardzo proszę o uszeregowanie wymienionych poniżej krajów od tych, które najchętniej byś zwiedził/a: [pytanie rankingowe]

Odpowiedź / Pozycja	1	2	3	4	5	Średnia	Punkty
Ameryka Południowa	14,29%	85,71%	0,00%	0,00%	0,00%	1.86	0
Ameryka Północna	0,00%	14,29%	85,71%	0,00%	0,00%	2.86	0
Europa	28,57%	0,00%	14,29%	57,14%	0,00%	3	0
Azja	28,57%	0,00%	0,00%	42,86%	28,57%	3.43	0
Australia	28,57%	0,00%	0,00%	0,00%	71,43%	3.86	0

Wypełnienia: 7



Rysunek 4: Przykładowe okno analizy wyników ankiety w serwisie Ankietka.pl

### 2.4.3 SurveyMonkey.com

Jako ostatni omówiony zostanie serwis SurveyMonkey.com. W porównaniu z wcześniej przedstawionymi rozwiązaniami, SurveyMonkey.com jest najbardziej rozbudowanym, profesjonalnym narzędziem, powszechnie używanym przez wiele światowych firm (Facebook, Samsung, Virgin). Jak można się spodziewać, za wysoką jakość trzeba zapłacić. Podobnie jak na stronie Ankietka.pl istnieje darmowa wersja serwisu, jednak ograniczenia są bardzo duże: limit pytań w ankiecie - 10, limit odpowiedzi - 100. W płatnej wersji (najtańsza wersja - 75 zł/msc) otrzymujemy wiele użytecznych funkcji, opisanych niżej. Dokładna oferta serwisu została przedstawiona na Rysunku 5.

## Zalety

- Kompleksowe, profesjonalne narzędzie
- Bezpieczeństwo rozwiązania (SSL)
- Konfigurowalny wygląd ankiety
- Analiza tekstu dla otwartych pytań

## Wady

- Ograniczenia w bezpłatnej wersji
- Większość funkcjonalności dostępna w płatnej części serwisu (minimum 75 zł/msc)

<b>BASIC</b> Free	<b>SELECT</b> zł 75 per month <small>SAVE with an annual plan</small>	<b>GOLD</b> zł 899 per year <small>Most Popular</small>	<b>PLATINUM</b> zł 2,599 per year
<a href="#">Sign Up »</a>	<a href="#">Sign Up »</a>	<a href="#">Sign Up »</a>	<a href="#">Sign Up »</a>
<b>Features include:</b>	<b>BASIC features +</b>	<b>SELECT features +</b>	<b>GOLD features +</b>
10 questions per survey 100 responses per survey	Unlimited questions 1,000 responses per month* <small>* zł 0.50 per additional response</small>	Unlimited questions Unlimited responses	Unlimited questions Unlimited responses
Easy-to-use web-based survey tool	Custom survey design & URLs	Custom redirect after survey is completed	Complete brand control with Research.net
Collect data via <b>weblink, email, Facebook</b> , or <b>embed</b> on your site or blog	Enhanced security (SSL/HTTPS) included	<b>Advanced logic features:</b> <ul style="list-style-type: none"><li>• Random assignment for A/B testing</li><li>• Question &amp; answer piping</li><li>• Question randomization or flipping</li></ul>	<ul style="list-style-type: none"><li>• Your own research.net survey URLs</li><li>• You control how your survey looks including adding your logo &amp; brand colors</li><li>• You decide where your respondents go after they complete your survey</li></ul>
Real-time results	Skip-logic & other advanced features	Text analysis for open responses	Expert phone support to answer any of your questions
24x7 email customer support	Excel export & printable PDF	SPSS integration	
<a href="#">See all features...</a>	<a href="#">See all features...</a>	<a href="#">See all features...</a>	<a href="#">See all features...</a>

Rysunek 5: Oferta serwisu SurveyMonkey.com

## 3 Architektura systemu

### 3.1 Organizacja aplikacji

Aplikację tworzą trzy oddzielne części:

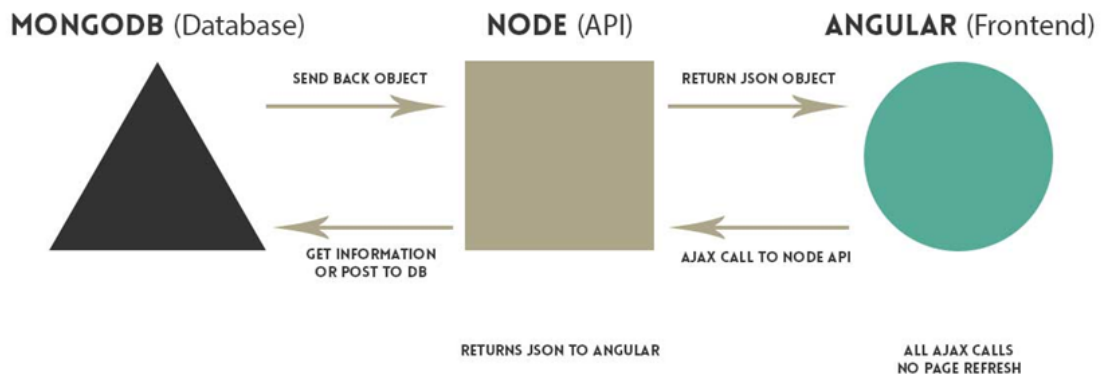
- serwer HTTP (node.js),
- baza danych (mongoDB),
- warstwa kliencka (AngularJS).

Tego typu architektura określana jest mianem MEAN (**M**ongoDB, **E**xpress.js, **A**ngular.js, **N**ode.js). Serwer HTTP udostępnia REST'owe API, będące podstawowym kanałem komunikacji między klientem a serwerem. API zostało dokładniej opisane w punkcie 4.3.

W przeciwieństwie do tradycyjnego podejścia do tworzenia aplikacji internetowych, w projekcie zastosowano koncepcję Single Page Application. Polega ona na tym, że przy przechodzeniu między poszczególnymi widokami nie jest wymagane pobranie całej zawartości strony (HTML + arkusze stylów + skrypty), ale jedynie części widoku. Część kliencka aplikacji (Angular) wykonuje zapytanie typu AJAX do serwera i w odpowiedzi otrzymuje fragment strony HTML, który to fragment zostaje umieszczony w zdefiniowanym przez programistę miejscu. Wszelkie dane dynamiczne (np. dane o odpowiedziach do ankiety) są pobierane z serwera w ten sam sposób. [4]

Dane wykorzystywane w aplikacji przesyłane są pomiędzy poszczególnymi jej częściami w formacie dokumentów JSON.

Rysunek 6. przedstawia architekturę projektowanego systemu.



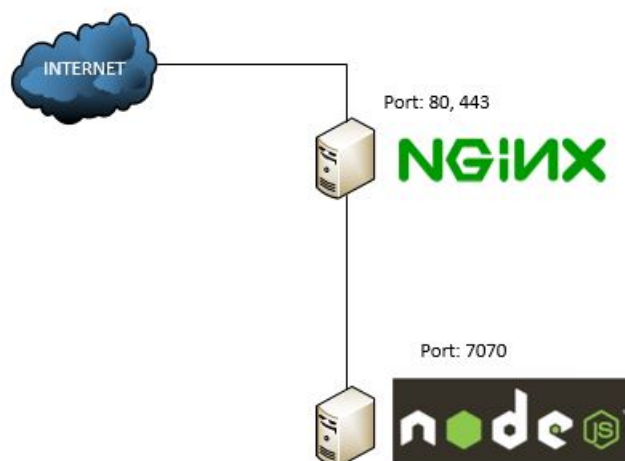
Rysunek 6: Architektura aplikacji

Źródło: <https://scotch.io/wp-content/uploads/2013/11/mean.jpg>

Funkcję persystencji pełni w aplikacji baza danych MongoDB. Jest to nierelacyjna baza danych, w której dane są przechowywane w postaci dokumentów JSON, co ujednolica strukturę danych w całym projekcie. Więcej informacji o MongoDB znajduje się w punkcie 4.1.2.

Schodząc na nieco niższy poziom architektury, trzeba opisać strukturę serwera HTTP. Node.js sam w sobie nie jest widoczny publicznie na porcie 80. W rzeczywistości wykorzystany został serwer Nginx jako reverse proxy: wszystkie zapytania na port 80 kieruje najpierw na port 443, gdzie dokładane są zabezpieczenia, a następnie na port 7070, na którym pracuje serwer Node.js. [5]

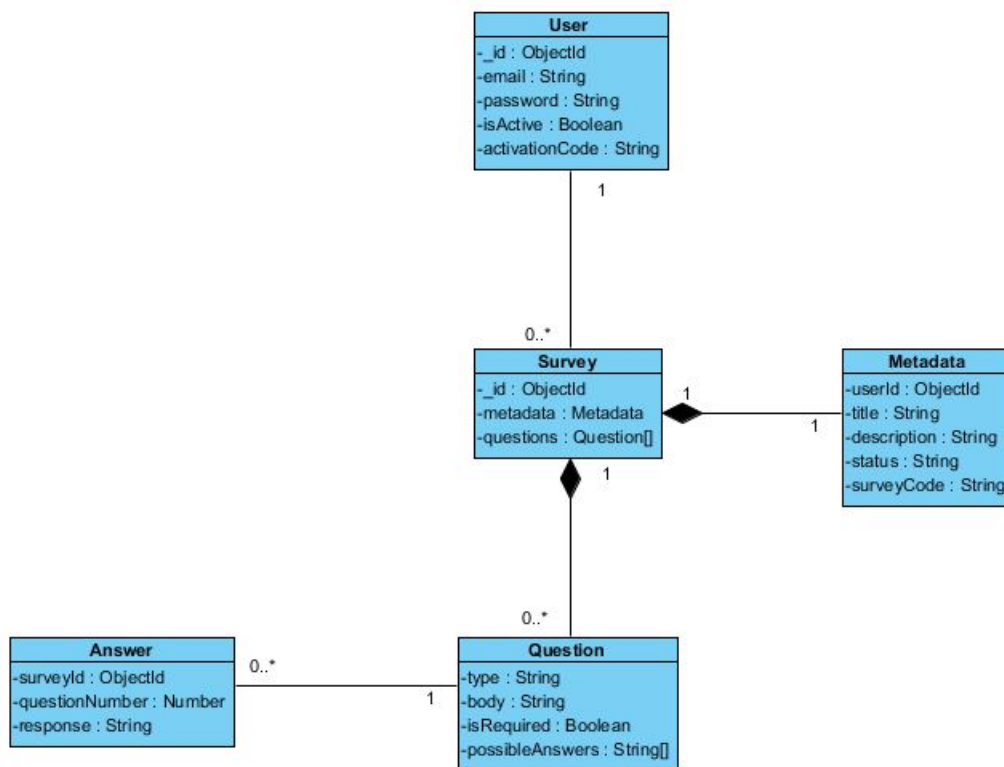
Rysunek 7. przedstawia architekturę systemu wzbogaconą o serwer Nginx.



Rysunek 7: Zastosowanie Nginx jako reverse proxy

### 3.2 Struktura bazy danych

Baza MongoDB jest bazą nierelacyjną, która pozwala na zagnieżdżanie obiektów. Jest to bardzo wygodne, ponieważ użytkownik ma możliwość przechowywania powiązanych ze sobą danych w jednej kolekcji, co przyspiesza proces wydobywania danych z bazy - nie ma konieczności tworzenia skomplikowanych często JOIN'ów. Struktura bazy danych przedstawiona została na Rysunku 8. [7] [12]



Rysunek 8: Struktura bazy danych

## 4 Implementacja

### 4.1 Wykorzystane technologie

#### 4.1.1 Node.js

Platforma Node.js jest staję się coraz bardziej popularnym narzędziem do tworzenia aplikacji serwerowych. W przeciwieństwie do tradycyjnego podejścia (np. Apache), w Node.js programista sam tworzy serwer HTTP. Wbrew pozorom, stworzenie takiego serwera zajmuje jedną linię kodu.

Node.js to biblioteka umożliwiająca wykonywanie skryptów JavaScript dzięki użyciu silnika V8 firmy Google, na której oparty został Google Chrome. Jest rozwiązaniem multiplatformowym, opartym na architekturze sterowanej zdarzeniami. Node.js operuje na jednym wątku, jednak dzięki zastosowaniu nieblokującej obsługi wejścia/wyjścia, pozwala na obsługę dziesiątek tysięcy równoległych połączeń do serwera. Takie asynchroniczne podejście wymaga, aby każda operacja wykorzystująca strumień we/wy używała funkcji zwrotnej (ang. *callback*), która wykonuje się po otrzymaniu danych. [9] [11]

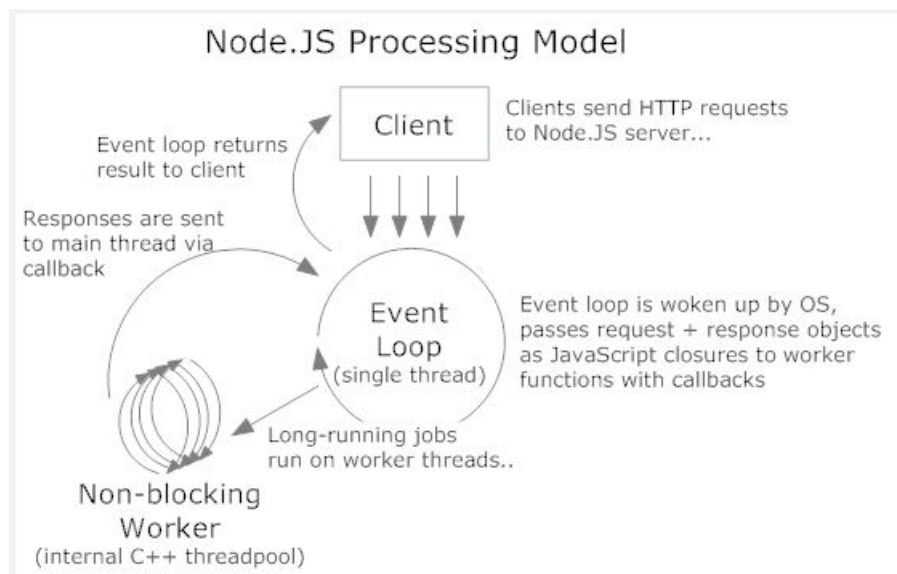
Listing 1. pokazuje sposób, w jaki w Node.js obsługuje się operacje asynchroniczne. Jako drugi parametr funkcji przekazywana jest funkcja zwrotna (ang. *callback*), która zostaje wywołana kiedy przychodzi odpowiedź funkcji `getSurveyById`.

Listing 1: Przykład użycia funkcji zwrotnej w pobraniu danych z bazy danych.

```
Surveys.getSurveyById($stateParams.surveyId,
  function(survey) {
    /**
     * zrob cos
     */
  });
```

Działanie Node.js opiera się na tzw. pętli zdarzeń. Wszystkie zdarzenia zachodzące podczas działania programu obsługiwane są w tej pętli w kolejności, w jakiej się w niej znajdują. Takie podejście nie blokuje działania programu, więc mimo faktu, że Node.js działa na jednym wątku, możliwe wysłanie odpowiedzi do wielu klientów jednocześnie. Rysunek 9. pokazuje schemat działania pętli zdarzeń Node.js. [3]





Rysunek 9: Schemat działania pętli zdarzeń w Node.js

Źródło: <http://sekurak.pl/wp-content/uploads/2013/08/nodejseventloop.png>

#### 4.1.2 MongoDB

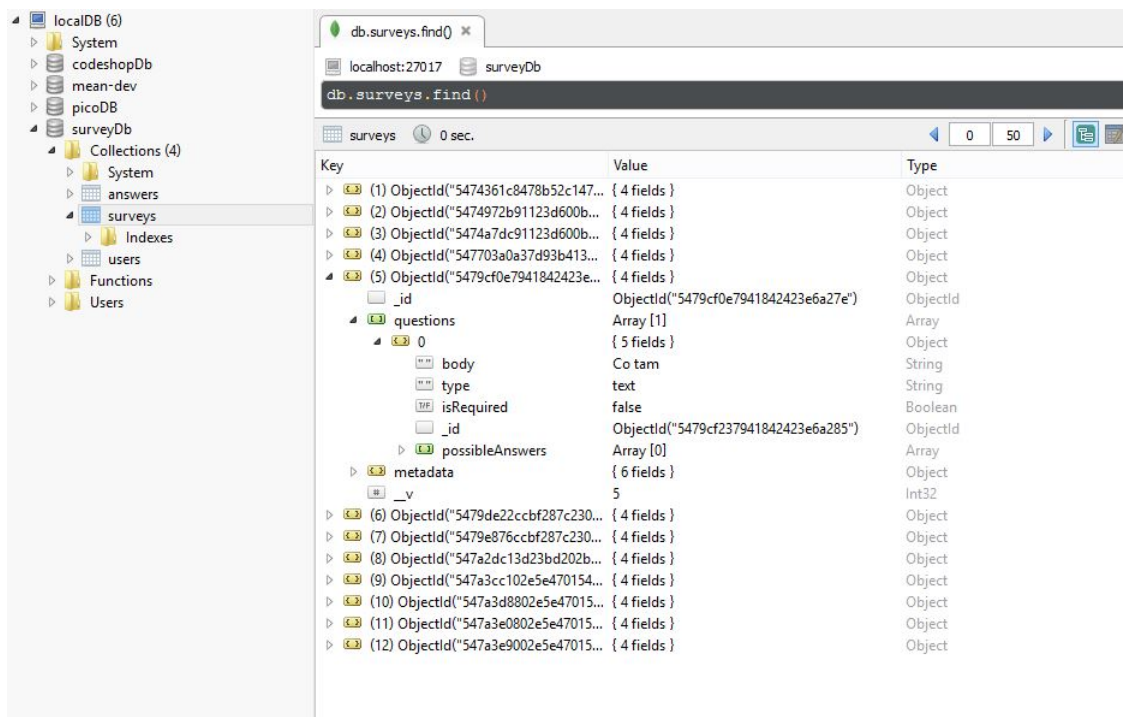
MongoDB to nierelacyjny system zarządzania bazą danych napisany w języku C++. Charakteryzuje się dużą skalowalnością, wydajnością oraz brakiem ściśle zdefiniowanej struktury obsługiwanych baz danych. Zamiast tego, dane składowane są jako dokumenty w stylu JSON, co umożliwia aplikacjom bardziej naturalne ich przetwarzanie, przy zachowaniu możliwości tworzenia hierarchii oraz indeksowania. [7]

Dane w MongoDB przechowywane są w formie dokumentów JSON, co pozwala na zagnieżdżanie danych. Programista ma możliwość umieszczenia powiązanych ze sobą danych w jednej kolekcji, co zmniejsza czas dostępu do tych danych. W relacyjnej bazie danych podczas pobierania danych często zachodzi potrzeba zwracania się do różnych tabel za pomocą JOIN'ów, w MongoDB jest to wyeliminowane. [12]

Podobnie jak w relacyjnych bazach danych, w MongoDB istnieje możliwość indeksowania pól dokumentu.

Baza MongoDB posiada dobrą obsługę klastrowania i replikacji danych, co obecnie jest bardzo często wykorzystywane przy przetwarzaniu rozległych zbiorów danych.

Na rysunku 10. pokazano sposób przechowywania danych w MongoDB, przy użyciu graficznego interfejsu Robomongo.



Rysunek 10: Struktura danych w bazie MongoDB

#### 4.1.3 Mongoose

Biblioteka Mongoose pozwala na ustandaryzowanie dostępu do bazy danych MongoDB. Generalnie, do MongoDB użytkownik może zapisać jakiekolwiek dane, nie istnieje żaden natywny sposób ograniczania lub walidacji wprowadzanych danych. Z tego powodu powstała właśnie biblioteka Mongoose, w której programista może zdefiniować strukturę kolekcji i przy zapisywaniu danych do bazy przeprowadzana jest ich walidacja. [8]

Listing 2. pokazuje sposób deklarowania struktury pojedynczej kolekcji bazy MongoDB z użyciem biblioteki Mongoose.

Listing 2: Przykład definicji struktury kolekcji w bibliotece Mongoose

```
var surveySchema = mongoose.Schema({
  metadata: {
    userId: {
      type: mongoose.Schema.Types.ObjectId,
      required: true
    },
    title: {
      type: String,
```

```

        required: true
    },
    description: String,
    status: {
        type: String,
        enum: ['draft', 'inProgress', 'finished']
    },
    answerCount: Number,
    surveyCode: {
        type: String,
        unique: true
    }
},
questions: [questionSchema]
});

```

#### 4.1.4 Angular.JS

Angular.JS jest frameworkiem frontendowym umożliwiającym tworzenie aplikacji w stylu Single Page Application. Angular dzięki zastosowaniu wzorca MVC pozwala w efektywny sposób oddzielić logikę aplikacji od widoku HTML. Najważniejszą funkcją Angulara jest dwukierunkowe wiązanie danych, która redukuje ilość kodu napisanego w trakcie uwalniania backendu serwera z odpowiedzialności za szablony. Szablony są stworzone w prostym HTMLu zgodnie z danymi zawartymi w zakresie (scope) zdefiniowanym przez model. Serwis \$scope w Angular wyłapuje zmiany w modelu i modyfikuje HTML w widoku poprzez kontroler. Podobnie, wszelkie zmiany w widoku widać w modelu. To pozwala ominąć potrzebę aktywnego manipulowania DOMu i ułatwia samodzielne i szybkie tworzenie aplikacji internetowych. Angular wyłapuje zmiany w modelach przez porównanie wartości z wartościami zgromadzonymi we wcześniejszym procesie dirty-checking. [1] [2]

#### 4.1.5 Inne technologie

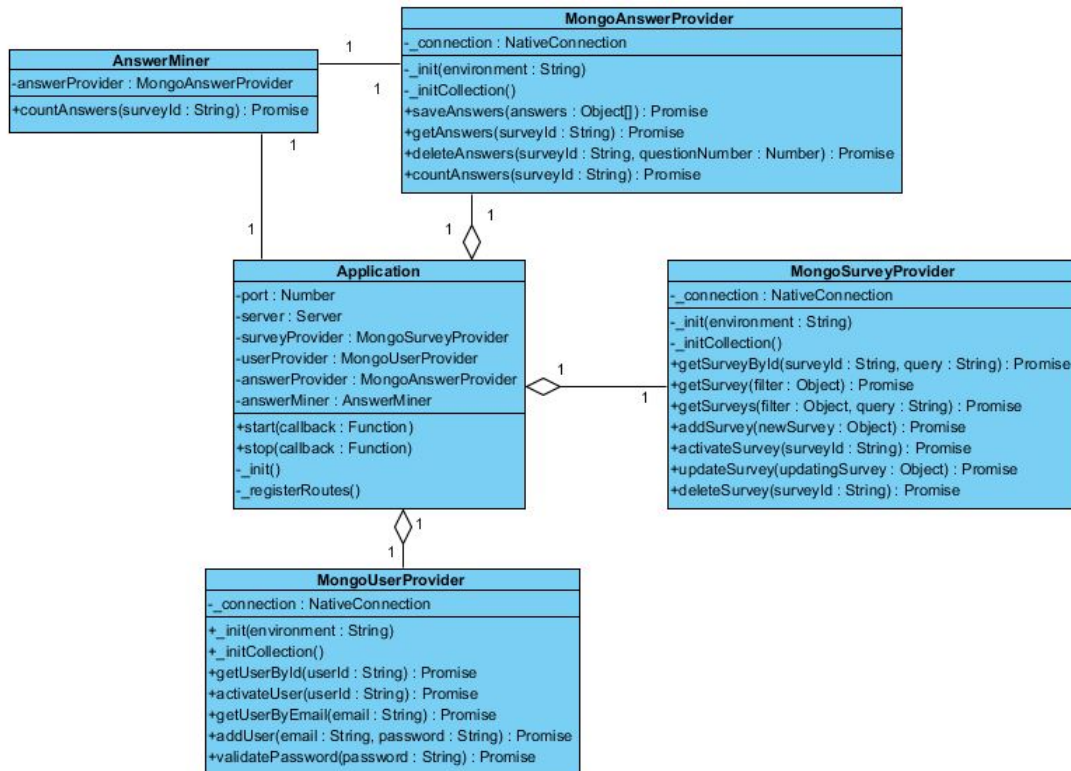
Oprócz wymienionych wyżej technologii w projekcie użytych zostało więcej dodatkowych bibliotek przydatnych przy realizacji pomniejszych zadań. Każda z niżej wymienionych technologii jest dostępna na licencji MIT.

- Express.js - lekki framework dla Node.js ułatwiający stworzenie rozbudowanej aplikacji internetowej.
- Twitter Bootstrap 3.0 - framework frontendowy zawierający gotowe arkusze stylów, ułatwiający szybkie zbudowanie aplikacji o atrakcyjnym interfejsie.
- Chart.JS - Biblioteka do rysowania wykresów na stronie HTML. Wykorzystuje znaczniki `<canvas>` wprowadzone w HTML5.
- passport.js - Moduł odpowiedzialny za obsługę uwierzytelniania i autoryzacji użytkowników systemu oraz za obsługę sesji HTTP.
- Lodash.js - Biblioteka „ogólnego użytku”, dostarcza wielu przydatnych funkcji, jak wydajna obsługa kolekcji.
- Nodemailer - Moduł odpowiedzialny za wysyłanie emaili do użytkowników. W aplikacji wykorzystywany przy wysyłaniu linków aktywacyjnych do nowo powstałych kont.

## 4.2 Diagram klas

Język JavaScript, w jakim wykonana została zarówno część serwerowa, jak i kliencka, nie wymaga od programisty wykorzystywania klas i obiektów. Jednak w celu lepszej skalowalności zaleca się, aby w większych aplikacjach używać podejścia obiektowego. W obecnym etapie rozwoju języka JavaScript nie istnieje niestety kontrola typów, co znacznie utrudnia pracę, ale w standardzie EcmaScript 6 planowane jest jej wprowadzenie.

W części serwerowej aplikacji zdefiniowano kilka klas, których struktura widoczna jest na Rysunku 11.



Rysunek 11: Diagram klas części serwerowej

Centralną klasą, odpowiedzialną za odpowiedzi na zapytania HTTP klientów, jest klasa `Application`. W metodzie `_registerRoutes()` inicjowana jest tablica routingu, czyli jakie adresy url są widoczne dla klientów i jak są one obsługiwane. Klasa po wywołaniu funkcji `start()` tworzy serwer HTTP na zdefiniowanym w konstruktorze porcie, na który przyjmowane są zapytania od klientów.

Klasy `MongoSurveyProvider`, `MongoAnswerProvider`, `MongoUserProvider` odpowiedzialne są za kontakt aplikacji z serwerem bazy danych. Są one swoistym wrapperem na bibliotekę `Mongoose` i dostarczają funkcji, za pomocą których programista może pobrać z bazy danych interesujące go informacje.

Klasa `AnswerMiner` jest pomocniczą klasą, która jest odpowiedzialna za wstępne przetworzenie danych o wynikach ankiet do struktury wymaganej przez bibliotekę `Chart.js` odpowiadającą za rysowanie wykresów.

### 4.3 API

W celu umożliwienia komunikacji między stroną kliencką a serwerem, na serwerze zdefiniowano REST'owe API, do którego można wysyłać zapytania typu HTTP. API dostarcza możliwości pobierania oraz manipulowania danymi z bazy danych.

Definiuje również ścieżki dostępu do fragmentów stron HTML, które później są dynamicznie renderowane przez Angulara. Na Listingu 3. pokazano sposób definiowania tablicy routingu w serwerze Node.js.

Listing 3: Definicja routingu w Node.js

```
router.route('/api/surveys')
  .get(hasAccess, surveys.getSurveys)
  .post(hasAccess, surveys.addSurvey);

router.route('/api/surveys/:surveyId')
  .get(hasAccess, surveys.getSurveyById)
  .put(hasAccess, surveys.updateSurvey)
  .delete(hasAccess, surveys.deleteSurvey);

router.route('/api/surveys/:surveyId/activate')
  .post(hasAccess, surveys.activateSurvey);

router.route('/api/answers')
  .post(respond.saveAnswer);

router.route('/api/answers/:surveyId')
  .get(respond.getAnswers)
  .post(respond.deleteAnswers);

router.route('/api/code/:surveyCode')
  .get(surveys.getSurveyByCode);

router.route('/')
  .get(views.index);

router.route('/login')
  .get(auth.loginView)
  .post(passport.authenticate('local-login'));

router.route('/signup')
  .get(auth.signupView)
```

```

    .post(passport.authenticate('local-signup'));

router.route('/activation')
    .get(isLoggedIn, auth.activateUser);

router.route('/profile')
    .get(isActive, views.profile);

router.route('/survey/:surveyCode')
    .get(respond.surveyView);

router.route('/partials/:filename')
    .get(views.partials);

router.route('/partials/directives/:filename')
    .get(isLoggedIn, views.directive);

router.route('/logout')
    .get(auth.logout);

app.use('/', router);

```

Obiekt typu Router definiuje konkretną ścieżkę HTTP, a następnie przypisuje jej obsługiwane funkcje (GET, POST, PUT, DELETE). Jeśli w argumencie przekazanych jest więcej funkcji, wykonywane są one jedna po drugiej. W tym przypadku w wielu miejscach sprawdzane jest najpierw, czy użytkownik wykonujący zapytanie do serwera jest zalogowany (funkcja `hasAccess`), a potem dopiero wykonuje właściwą część zapytania.

Na szczególną uwagę zasługują ścieżki zaczynające się przedrostkiem `/partials`. Wysyłają one do klienta fragmenty widoków HTML, które następnie AngularJS wstawia w odpowiednie miejsca na stronie.

Dane z serwera są wysyłane do klienta w postaci dokumentów JSON.

#### 4.3.1 /api/surveys

Metody:

- **GET**

Zwraca dane o ankietach

- **POST**

Dodaje do bazy danych nową ankietę.

Parametry:

- title {String} - tytuł ankiety
- description {String} - opis ankiety

#### 4.3.2 /api/surveys/{surveyId}

Metody:

- **GET**

Zwraca dane o ankiecie o podanym identyfikatorze.

Parametry:

- surveyId {String} - identyfikator ankiety

- **PUT**

Modyfikuje ankietę o podanym ID.

Parametry:

- surveyId {String} - identyfikator ankiety
- survey {Object} - zmodyfikowana ankietka

- **DELETE**

Usuwa ankietę o podanym ID.

Parametry:

- surveyId {String} - identyfikator ankiety



#### 4.3.3 /api/surveys/{surveyId}/activate

Metody:

- **POST**

Zmienia status ankiety o podanym ID na 'inProgress'.

Parametry:

- surveyId {String} - identyfikator ankiety

#### 4.3.4 /api/answers

Metody:

- **POST**

Zapisuje pojedynczy zbiór odpowiedzi do podanej ankiety.

Parametry:

- answers {String} - zbiór odpowiedzi do konkretnj ankiety. Zawiera w sobie ID ankiety

#### 4.3.5 /api/answers/{surveyId}

Metody:

- **GET**

Pobiera informacje o odpowiedziach w ankiecie o podanym ID.

Parametry:

- surveyId {String} - identyfikator ankiety

- **POST**

Usuwa odpowiedzi na konkretne pytanie z podanej ankiety.

Parametry:

- surveyId {String} - ID ankiety
- questionNumber {Number} - numer pytania

#### 4.3.6 /api/code/{surveyCode}

Metody:

- **GET**

Pobiera informacje o ankiecie o podanym kodzie ankiety.

Parametry:

- surveyCode {String} - kod ankiety

### 4.4 Komunikacja serwera HTTP z bazą danych

Poniżej opisane zostały klasy pełniące funkcje dostępu do bazy danych.

#### 4.4.1 MongoSurveyProvider

Klasa odpowiedzialna za dostęp do danych z kolekcji **surveys**.

Metody:

Tabela 2: Opis funkcji klasy MongoSurveyProvider

Typ zwracany	Opis
q.Promise	<b>getSurveyById</b> ({String} surveyId, {String} query) Pobiera z bazy danych ankietę i przekazuje ją do funkcji zwrotnej Parametry: surveyId - identyfikator ankiety query - ciąg znaków informujący jakie dane ankiety wyciągnąć
q.Promise	<b>getSurvey</b> ({Object} filter, {String} query) Pobiera z bazy danych tablicę ankiet spełniających odpowiednie kryteria filtrowania Parametry: filter - kryteria filtrowania query - ciąg znaków informujący jakie dane ankiety wyciągnąć
q.Promise	<b>addSurvey</b> ({Object} newSurvey) Dodaje do bazy danych nową ankietę. Parametry: newSurvey - ankietą zapisywana do bazy
q.Promise	<b>activateSurvey</b> ({String} surveyId) Aktywuje ankietę o podanym ID. Parametry: surveyId - identyfikator ankiety
q.Promise	<b>updateSurvey</b> ({String} surveyId, {Object} updatingSurvey) Aktualizuje dane ankiety o podanym ID. Parametry: surveyId - identyfikator ankiety updatingSurvey - zaktualizowana ankietą
q.Promise	<b>deleteSurvey</b> ({String} surveyId) Usuwa ankietę o podanym ID. Parametry: surveyId - identyfikator ankiety

Na listingu 4. pokazano przykład użycia metody `getSurvey` w celu pobrania z bazy i wysłania do użytkownika ankiety o podanym kodzie.

Listing 4: Przykład użycia metody `getSurvey`

```
surveyProvider.getSurvey({  
  "metadata.surveyCode": req.params.surveyCode  
}).then(function(result) {  
  MessageSender.sendJsonObject(res, result);  
});
```

#### 4.4.2 MongoAnswerProvider

Klasa odpowiedzialna za dostęp do kolekcji `answers`.

Metody:

Tabela 3: Opis funkcji klasy `MongoAnswerProvider`

Typ zwracany	Opis
q.Promise	<b>saveAnswers</b> ({Object[]} answers) Dodaje do bazy danych odpowiedzi do konkretnej ankiety. Parametry: answers - tablica obiektów opisujących odpowiedzi do ankiety
q.Promise	<b>getAnswers</b> ({String} surveyId) Pobiera z bazy danych odpowiedzi do konkretnej ankiety Parametry: surveyId - identyfikator ankiety
q.Promise	<b>deleteAnswers</b> ({String} surveyId, {Number} questionNumber) Usuwa z bazy danych odpowiedzi dotyczące zadanego pytania w ankiecie. Parametry: surveyId - identyfikator ankiety questionNumber - numer pytania
q.Promise	<b>countAnswers</b> ({String} surveyId) Oblicza ile odpowiedzi padło w danej ankiecie na dane pytanie. Parametry: surveyId - identyfikator ankiety

Na listingu 5. pokazano przykład użycia metody `saveAnswers`, w którym do bazy danych zapisywane są odpowiedzi do ankiety przesłane przez respondenta.

Listing 5: Przykład użycia metody `saveAnswers`

```
answerProvider.saveAnswers(answers)
  .then(function (result) {
    MessageSender.sendJsonObject(res, result);
  }, function (err) {
    MessageSender.sendDatabaseError(res, err);
  });
```

#### 4.4.3 MongoAuthProvider

Klasa odpowiedzialna za dostęp do kolekcji **users**

Metody:

Tabela 4: Opis funkcji klasy MongoAuthProvider

Typ zwracany	Opis
q.Promise	<b>getUserById</b> ({String} userId, {String} query) Pobiera dane o użytkowniku o podanym identyfikatorze. Parametry: userId - ID użytkownika query - ciąg znaków informujący jakie dane użytkownika wyciągnąć
q.Promise	<b>getUserByEmail</b> ({String} email) Pobiera z bazy danych dane użytkownika o podanym emailu. Parametry: email - adres email użytkownika
q.Promise	<b>activateUser</b> ({String} activationCode) Aktywuje konto użytkownika. Parametry: activationCode - kod aktywacyjny użytkownika
q.Promise	<b>addUser</b> ({String} email, {String} password) Dodaje do bazy danych nowego użytkownika. Parametry: email - adres email nowego użytkownika password - hasło nowego użytkownika
Boolean	<b>validatePassword</b> ({String} password) Sprawdza poprawność wprowadzonego hasła. Parametry: password - hasło użytkownika

Na listingu 6. pokazano przykład użycia metody **getUserByEmail** podczas operacji rejestrowania użytkownika w serwisie.

Listing 6: Przykład użycia metody `getUserByEmail`

```
userProvider.getUserByEmail(email)
  .then(function(user) {
    if(user) {
      return done(null, false,
        req.flash('signupMessage',
          Message.EMAIL_TAKEN.en));
    } else {
      userProvider.addUser(email, password,
        function(err, newUser) {
          if (err)
            throw err;
          return done(null, newUser);
        });
    }
  }, function(err) {
    return done(err);
  });
```

## 5 Testowanie

W trakcie pisania aplikacji częściowo wprowadzona została koncepcja TDD (ang. *Test Driven Development*). Napisane zostały testy jednostkowe dla funkcji dostępu do bazy danych oraz testy REST’owego API. Zdefiniowano również szereg scenariuszy testowych dla testów funkcjonalnych.

Poniżej przedstawiono tylko niektóre testy, które zostały przeprowadzone na aplikacji.

### 5.1 Tworzenie nowej ankiety

**Cel testu:** testowanie poprawnego tworzenia ankiety.

**Scenariusz (kroki testowe):**

Akcje użytkownika	Odpowiedź systemu
1. Przejście na stronę logowania.	2. Wyświetlenie strony logowania.
3. Wpisanie poprawnych danych logowania.	4. Przekierowanie na stronę użytkownika
5. Naciśnięcie przycisku „Create survey”	6. Wyświetlenie strony do wpisania danych ankiety
7. Wpisanie danych ankiety i zatwierdzenie.	8. Przejście na stronę z podstawowymi danymi ankiety.

**Ocena testu:** Pozytywna



## 5.2 Edycja pytań ankiety

**Cel testu:** testowanie poprawnego edytowania pytań.

**Scenariusz (kroki testowe):**

Akcje użytkownika	Odpowiedź systemu
1. Przejście na stronę logowania.	2. Wyświetlenie strony logowania.
3. Wpisanie poprawnych danych logowania.	4. Przekierowanie na stronę użytkownika
5. Wybranie jednej z ankiet widocznych na liście ankiet	6. Przejście na stronę z podstawowymi danymi ankiety.
7. Przejście do zakładki „Edit questions”.	8. Wyświetlenie widoku edycji pytań.
9. Wciśnięcie przycisku „Add new question”.	10. Dodanie na stronie nowego pytania.
11. Modyfikacja danych pytania.	12. Zapisanie zmodyfikowanego pytania.

**Ocena testu:** Pozytywna

### 5.3 Analiza zebranych wyników

**Cel testu:** testowanie poprawnego zliczania odpowiedzi.

**Scenariusz (kroki testowe):**

Akcje użytkownika	Odpowiedź systemu
1. Przejście na stronę logowania. 3. Wpisanie poprawnych danych logowania.  5. Wybranie jednej z ankiet widocznych na liście ankiet 7. Przejście do zakładki „Sharing”.  9. Wciśnięcie przycisku „Activate survey”.  11. Przejście w drugiej przeglądarce pod wygenerowany adres. 13. Oddanie głosu na konkretne pytanie i wysłanie odpowiedzi. 15. W panelu autora ankiety przejście do zakładki „Analysis”. 17. Naciśnięcie przycisku „Get answers”.	2. Wyświetlenie strony logowania. 4. Przekierowanie na stronę użytkownika 6. Przejście na stronę z podstawowymi danymi ankiety. 8. Wyświetlenie widoku udostępniania ankiety. 10. Wygenerowanie kodu ankiety. 12. Wyświetlenie widoku ankiety z możliwością odpowiedzi. 14. Wyświetlenie komunikatu o poprawnym zagłosowaniu. 16. Wyświetlenie widoku analizy wyników. 18. Wyświetlenie wykresu pokazującego jedną odpowiedź oddaną w drugiej przeglądarce.

**Ocena testu:** Pozytywna

## 5.4 Kontrola jednokrotnego głosowania

**Cel testu:** testowanie oddawania tylko jednego głosu przez respondenta.

**Scenariusz (kroki testowe):**

Akcje użytkownika	Odpowiedź systemu
1. Przejście na stronę logowania. 3. Wpisanie poprawnych danych logowania.  5. Wybranie jednej z ankiet widocznych na liście ankiet 7. Przejście do zakładki „Sharing”.  9. Wciśnięcie przycisku „Activate survey”.  11. Przejście w drugiej przeglądarce pod wygenerowany adres. 13. Oddanie głosu na konkretne pytanie i wysłanie odpowiedzi. 15. Zamknięcie przeglądarki i ponowne jej otworzenie, przejście pod wygenerowany adres.	2. Wyświetlenie strony logowania. 4. Przekierowanie na stronę użytkownika 6. Przejście na stronę z podstawowymi danymi ankiety. 8. Wyświetlenie widoku udostępniania ankiety. 10. Wygenerowanie kodu ankiety. 12. Wyświetlenie widoku ankiety z możliwością odpowiedzi. 14. Wyświetlenie komunikatu o poprawnym zagłosowaniu. Wyświetlenie komunikatu o poprawnym zagłosowaniu.

**Ocena testu:** Pozytywna

## 6 Wdrożenie

### 6.1 Node.js

Pierwszym krokiem do wdrożenia aplikacji Code-Shop jest zainstalowanie platformy *Node.js*. Aplikacja została stworzona w wersji *v0.10.33* platformy. Node.js jest biblioteką wieloplatformową i może być zainstalowany w większości popularnych systemów operacyjnych.

Rysunek 12. pokazuje dostępne sposoby pobierania Node.js.

		
<b>Windows Installer</b>	<b>Macintosh Installer</b>	<b>Source Code</b>
node-v0.10.33-x86.msi	node-v0.10.33.pkg	node-v0.10.33.tar.gz

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.exe)	32-bit	64-bit
Mac OS X Installer (.pkg)	Universal	
Mac OS X Binaries (.tar.gz)	32-bit	64-bit
Linux Binaries (.tar.gz)	32-bit	64-bit
SunOS Binaries (.tar.gz)	32-bit	64-bit
Source Code	node-v0.10.33.tar.gz	

Rysunek 12: Strona internetowa Node.js umożliwiająca pobranie środowiska

#### 6.1.1 Windows

Aby zainstalować *Node.js* na systemie operacyjnym *Windows* wystarczy przejść na stronę <http://nodejs.org/download/>, skąd można pobrać przeznaczone dla tego systemu instalatory (.msi lub .exe). Następnie należy postępować zgodnie z instrukcjami tegoż instalatora, co spowoduje zainstalowanie środowiska w systemie operacyjnym. [10]

#### 6.1.2 Linux

Poniżej przedstawiono instrukcję instalacji dla dwóch dystrybucji Linuxa: Debian i Ubuntu. Jest poprawna również dla dystrybucji opartych na tych dwóch, takich

jak Linux Mint, Linux Mint Debian Edition, elementaryOS.

Pierwszym krokiem jest skonfigurowanie repozytoriów Node.js:

```
curl -sL https://deb.nodesource.com/setup | bash -
```

Następnym, ostatnim krokiem jest zainstalowanie środowiska za pomocą menadżera pakietów *apt-get*:

```
apt-get install -y nodejs
```

Powyższe kroki odnoszą się do systemu Debian. Na Ubuntu proces instalacji jest praktycznie identyczny, należy tylko dodać komendę **sudo**. [10]

## 6.2 Nginx

### 6.2.1 Windows

W celu zainstalowania serwera na Windowsie, należy pobrać ze strony <http://nginx.org/en/download.html> plik .zip zawierający serwer Nginx. Następnie należy zmodyfikować konfigurację serwera w sposób opisany w punkcie 6.2.3. Aby uruchomić serwer jako serwis, należy przejść w konsoli do katalogu, gdzie został wypakowany, a następnie wpisać komendę:

```
start nginx
```

### 6.2.2 Linux

Aby zainstalować serwer Nginx na Linuksie (Debian/Ubuntu) należy zaktualizować repozytoria pakietów komendą

```
apt-get update
```

a następnie zainstalować komendą

```
apt-get install nginx
```

Po tym kroku należy zmodyfikować konfigurację serwera (domyślnie znajduje się w pliku `/etc/nginx/conf/nginx.conf`) w sposób opisany w punkcie 6.2.3, po czym zrestartować serwer komendą

```
service nginx restart
```

### 6.2.3 Konfiguracja serwera

W pliku konfiguracyjnym serwera Nginx (`nginx.conf`) w bloku **http** należy wprowadzić następujące dane:

```

server {
    listen 80;
    server_name {nazwa_serwera};
    rewrite ^ https://$server_name$request_uri;
}

server {
    listen 443;
    ssl on;
    server_name {nazwa_serwera};

    ssl_certificate {sciezka do certyfikatu SSL};
    ssl_certificate_key {sciezka do klucza SSL};

    location / {
        proxy_pass http://localhost:7070;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
    }
}

```

Przykładowa konfiguracja została przedstawiona poniżej.

```

server {
    listen 80;
    server_name example.com;
    rewrite ^ https://$server_name$request_uri;
}

server {
    listen 443;
    ssl on;
    server_name example.com;

    ssl_certificate /etc/ssl/server.crt;
    ssl_certificate_key /etc/ssl/server.key;

    location / {
        proxy_pass http://localhost:7070;
        proxy_set_header Upgrade $http_upgrade;
    }
}

```

```
proxy_set_header Connection 'upgrade';  
proxy_set_header Host $host;  
}
```

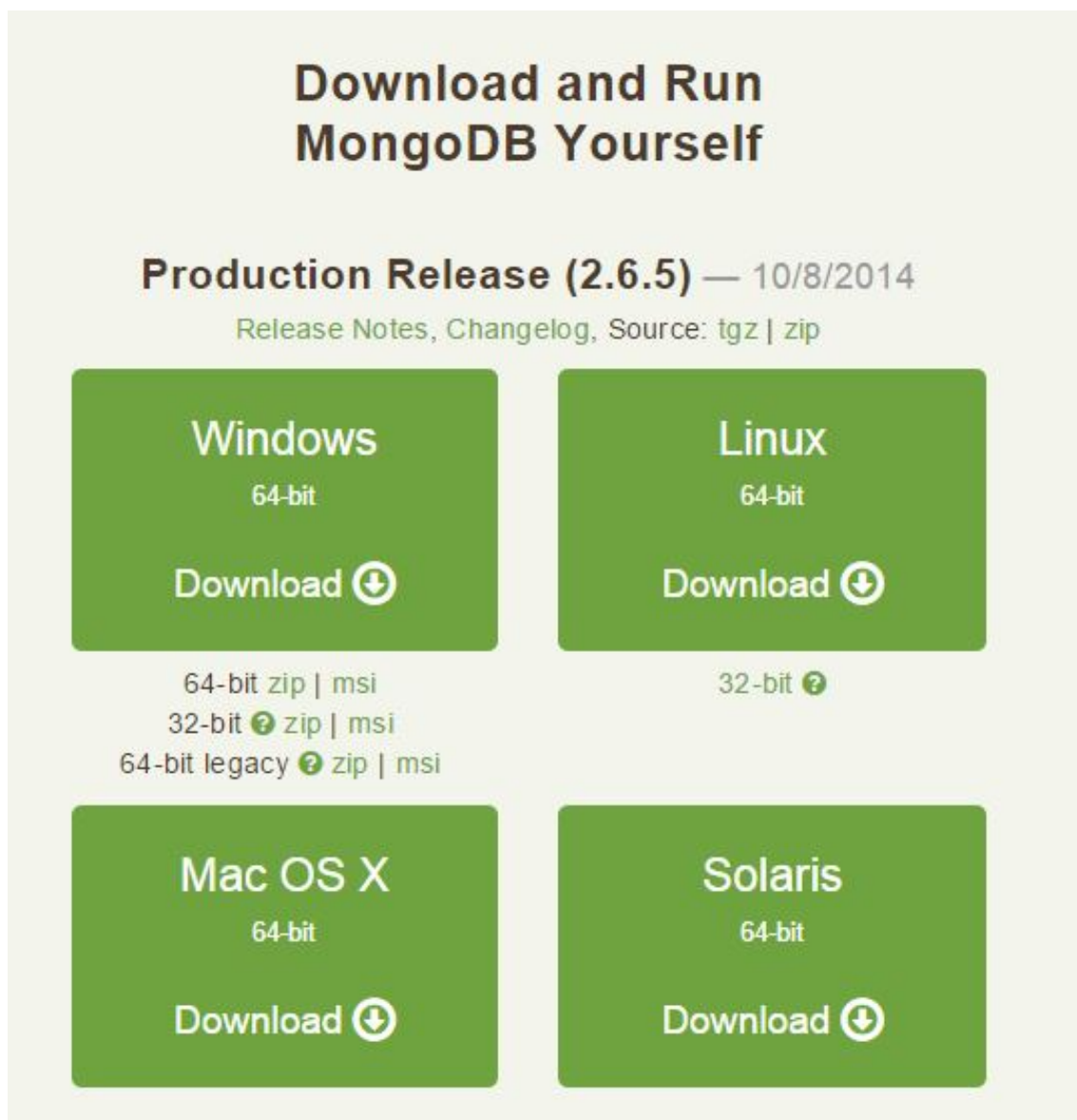
[5]

## 6.3 MongoDB

### 6.3.1 Windows

Baza danych MongoDB dla systemu Windows przeznaczona jest dla różnych architektur, więc w pierwszej kolejności należy zidentyfikować jaki typ systemu jest zainstalowany na danym komputerze. Następnie należy przejść na stronę pobierania MongoDB (<http://www.mongodb.org/downloads>) i pobrać odpowiednią wersję bazy (2.6.4). Kolejnym krokiem jest samo zainstalowanie MongoDB, które polega na postępowaniu według instrukcji instalatora. [12]

Rysunek 13. pokazuje sposoby pobierania MongoDB.



Rysunek 13: Strona internetowa MongoDB umożliwiająca pobranie bazy danych

Po zainstalowaniu bazy danych należy uruchomić ją jako serwis. W tym celu należy uruchomić wiersz poleceń systemu Windows jako administrator, a następnie postępować wg następujących kroków:

1. Stwórz katalogi na pliki bazy danych oraz na logi.

```
mkdir c:\data\db  
mkdir c:\data\log
```



2. Stwórz plik konfiguracyjny za pomocą następujących poleceń:

```
echo logpath=c:\data\log\mongod.log> "C:\Program  
Files\MongoDB 2.6 Standard\mongod.cfg"  
echo dbpath=c:\data\db>> "C:\Program Files\MongoDB  
2.6 Standard\mongod.cfg"
```

3. Utwórz serwis MongoDB.

```
sc.exe create MongoDB binPath= "\"C:\Program  
Files\MongoDB 2.6 Standard\bin\mongod.exe\""  
--service --config=\"C:\Program Files\MongoDB 2.6  
Standard\mongod.cfg\" DisplayName= "MongoDB 2.6  
Standard" start= "auto"
```

Jeśli komenda zakończy się sukcesem powinien pojawić się następujący komunikat:

```
[SC] CreateService SUCCESS
```

4. Uruchom serwis MongoDB

```
net start MongoDB
```

Po wykonaniu powyższego procesu w systemie Windows pracuje serwis MongoDB, z którego będzie korzystać aplikacja.

### 6.3.2 Linux

Aby zainstalować MongoDB na systemie typu Linux należy:

1. Zaimportować klucz publiczny przez menadżer pakietów systemu:

```
sudo apt-key adv --keyserver  
hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
```

2. Stworzyć plik .list dla MongoDB:

```
echo 'deb  
http://downloads-distro.mongodb.org/repo/ubuntu-upstart  
dist 10gen' | sudo tee  
/etc/apt/sources.list.d/mongodb.list
```

3. Zaktualizować lokalną bazę pakietów:

```
sudo apt-get update
```

4. Zainstalować pakiety MongoDB:

```
sudo apt-get install -y mongodb-org
```

5. Wystartować bazę danych jako serwis:

```
sudo service mongod start
```

Po wykonaniu tego kroku wystartowany został serwis MongoDB. Poniższa została przetestowana na dystrybucjach Ubuntu oraz Debian. [12]

## 6.4 Instalacja zależności projektowych

Aby uruchomić projekt należy najpierw zainstalować wszystkie niezbędne biblioteki, z których aplikacja będzie korzystać. Po przejściu do głównego katalogu, gdzie wypakowany został projekt, należy uruchomić następującą komendę:

```
npm install
```

Komenda ta instaluje wszystkie biblioteki zapisane w pliku `package.json`. Menadżer pakietów `npm` został zainstalowany razem z Node.js. Instalacja zewnętrznych bibliotek wygląda tak samo na wszystkich systemach operacyjnych.

## 6.5 Uruchomienie aplikacji

Po przejściu przez wszystkie poprzednie kroki użytkownik jest gotowy do uruchomienia aplikacji. Na początku należy ustawić zmienną systemową `NODE_ENV` na wartość `production`. Najprostszą wersją jest uruchomienia aplikacji jest wpisanie w terminalu następującej komendy:

```
node app.js
```

będąc w katalogu głównym projektu. Niestety po zamknięciu terminala proces zostanie zamknięty. Aby uruchomić proces jako serwis najlepiej użyć biblioteki `forever`. Instalacja biblioteki odbywa się za pomocą poznanego wcześniej menadżera pakietów `npm`. Aby zainstalować bibliotekę `forever` należy w terminalu wprowadzić następującą komendę:

```
npm install forever -g
```

Opcja `-g` informuje, że biblioteka ma być zainstalowana globalnie.

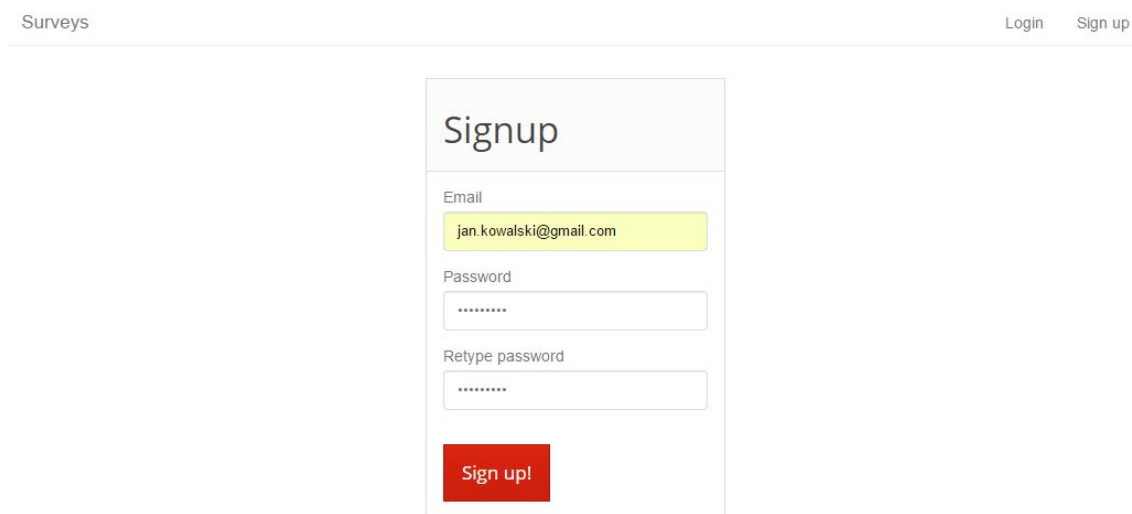
Po zainstalowaniu `forevera`, będąc w katalogu głównym projektu wprowadzić należy komendę:

```
forever start app.js
```

## 7 Instrukcja użytkownika

### 7.1 Rejestracja w serwisie

Aby zarejestrować się w systemie należy w panelu rejestracji podać swój adres email oraz wpisać hasło do konta. Na podany adres wysłany zostanie link aktywacyjny, który należy kliknąć. Po zrobieniu tego użytkownik przeniesiony zostanie do strony logowania, gdzie za pomocą swoich danych będzie mógł wejść do serwisu. Widok ekranu rejestracji przedstawiony został na Rysunku 14.



Surveys Login Sign up

**Signup**

Email  
jan.kowalski@gmail.com

Password  
\*\*\*\*\*

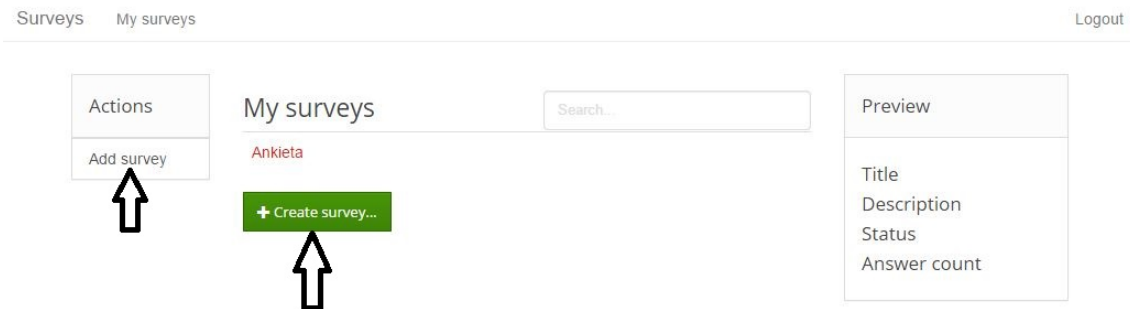
Retype password  
\*\*\*\*\*

**Sign up!**

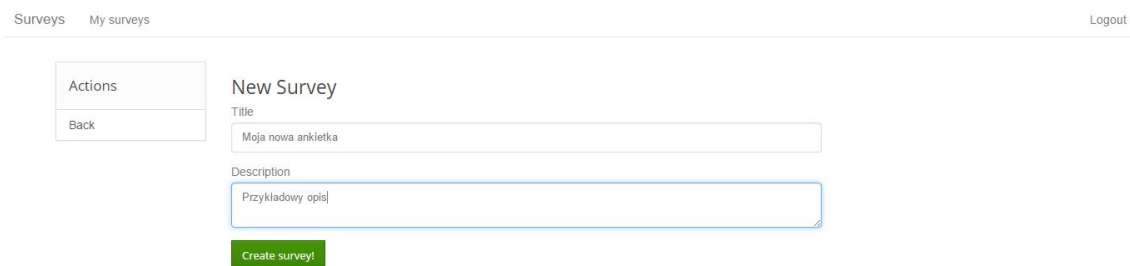
Rysunek 14: Strona rejestracji użytkownika

### 7.2 Tworzenie i edycja ankiety

Po zalogowaniu użytkownik ma możliwość stworzenia nowej ankiety oraz edycji już istniejących. Schemat działania jest praktycznie taki sam, więc został opisany w jednym punkcie. Aby stworzyć nową ankietę należy w głównym widoku nacisnąć przycisk „Create survey”, po czym wpisać tytuł i opis ankiety. Kroki te pokazane są kolejno na Rysunkach 15. i 16.



Rysunek 15: Tworzenie nowej ankiety



Rysunek 16: Tworzenie nowej ankiety

Następnie można przejść do edycji pytań. W celu dodania nowego pytania należy nacisnąć przycisk „Add new question”. Aby zmienić treść pytania należy na nie kliknąć i wpisać nowy tekst, po czym zatwierdzić przyciskiem. Można zmienić typ pytania, zdefiniować, czy pytanie jest wymagane, oraz dodać możliwe odpowiedzi. Po jakiegokolwiek edycji pytania, dane ankiety są automatycznie zapisywane w bazie danych.

Widok edycji pytań przedstawiony został na Rysunku 17.

Actions

Basic info

Edit description

**Edit questions**

Sharing

Analysis

### Questions

Jaki kolor lubisz

Question type: oneChoice

Is required? ☐

Possible answers:

- ☐ Czerwony
- ☐ Zielony
- ☐ Niebieski

Add answer

#2 New question

Add new question

Rysunek 17: Edycja pytań

### 7.2.1 Udostępnianie ankiety

Aby udostępnić ankietę dla respondentów należy wygenerować link do ankiety. W tym celu należy w widoku ankiety przejść do zakładki „Sharing”, a następnie kliknąć przycisk „Activate survey”. W polu tekstowym pojawi się link do ankiety, który można rozesłać do respondentów.

Ekran udostępniania ankiety pokazany został na Rysunku 18.

Surveys My surveys

Logout

Actions

Basic info

Edit description

Edit questions

**Sharing**

Analysis

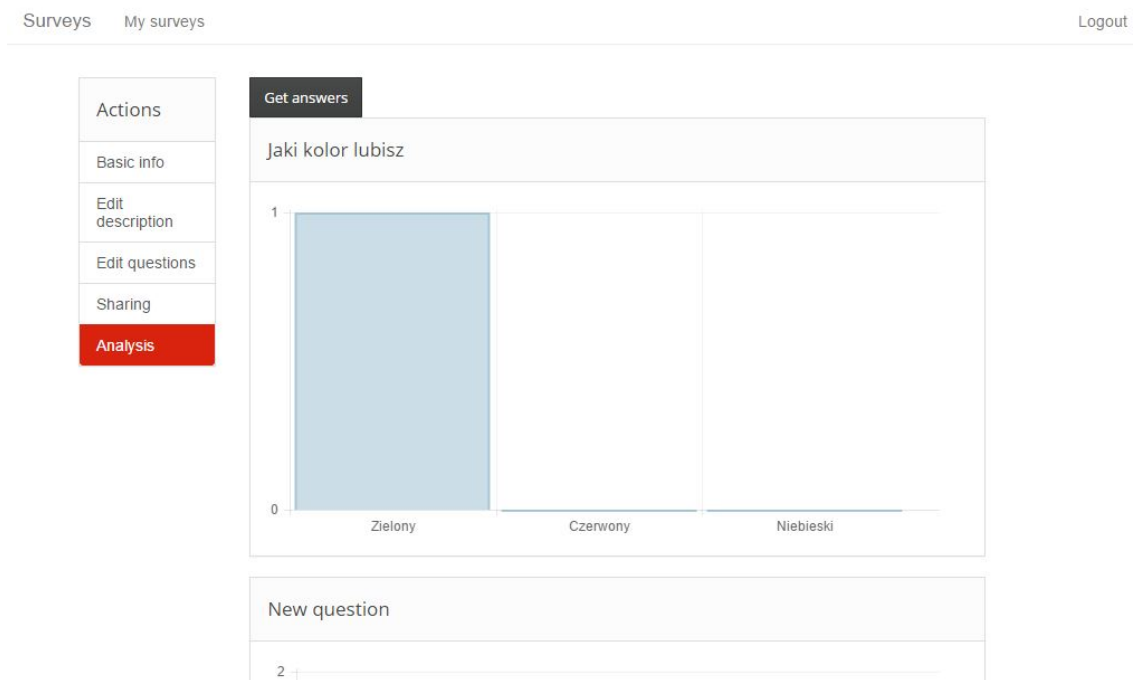
<https://91.196.50.18/survey/bPV9y1P4GB>

Rysunek 18: Udostępnianie ankiety

### 7.2.2 Analiza wyników

W celu przeprowadzenia analizy wyników ankiety należy w widoku ankiety przejść do zakładki „Analysis”. Następnie należy nacisnąć przycisk „Get answers”. Po wykonaniu tej czynności powinny wygenerować się wykresy przedstawiające wyniki ankiety.

Przykładowa analiza wyników pokazana została na rysunku 19.



Rysunek 19: Analiza ankiety

## 8 Podsumowanie i wnioski

Udało się zrealizować wszystkie niezbędne do działania systemu funkcjonalności. Funkcjonalności dodatkowe zostały pominięte w podstawowej wersji serwisu, ale planowana jest ich implementacja w przyszłych wersjach aplikacji.

Wybrane technologie, mimo stosunkowo młodego wieku, dobrze sprawdziły się w implementacji projektu. Szczególnie baza MongoDB, ze względu na swoją prostotę, znacznie ułatwiła stworzenie struktury danych przetwarzanych przez aplikację.

Zastosowanie koncepcji Single Page Application zwiększyło komfort korzystania z serwisu. Jest to rozwiązanie znacznie szybsze od standardowego podejścia. Obserwując dzisiejszy rynek technologii internetowych można zauważyć znaczny wzrost popularności tego typu rozwiązań, przede wszystkim dzięki powstawaniu nowoczesnych frameworków do tego przeznaczonych, np. użytemu w projekcie Angularowi.

### 8.1 Perspektywy rozwoju aplikacji

W kolejnych wersjach serwisu planowane jest zaimplementowanie dodatkowych funkcjonalności, przede wszystkim:

- Pobieranie wyników ankiety w czasie rzeczywistym dzięki zastosowaniu technologii WebSockets,
- Eksportowanie wyników ankiety do zewnętrznych plików (np. xlsx, csv).

Oprócz tego planowane jest rozwinięcie już istniejących funkcjonalności:

- Wprowadzenie większej liczby typów pytań (pytania macierzowe, skalowe).
- Zwiększenie możliwości analizy wyników ankiety (np. tabelki).
- Lepsza analiza odpowiedzi na pytania otwarte.

### 8.2 Opis zawartości płyty CD

Płyta CD dołączona do dokumentacji zawiera kod źródłowy projektu oraz elektroniczną wersję dokumentacji w formacie PDF. Kod źródłowy aplikacji umieszczony został w folderze „surveys”, a dokumentacja w folderze „dokumentacja”.



## Literatura

1. *AngularJS*, <http://pl.wikipedia.org/wiki/AngularJS>, 10.12.2014
2. *AngularJS API Docs*, <https://docs.angularjs.org/api>, 10.12.2014
3. *Bezpieczeństwo aplikacji internetowych bazujących na platformie Node.js*, <http://sekurak.pl/bezpiec>  
aplikacji-internetowych-opartych-o-platforme-node-js-czesc-1, 10.12.2014
4. Ethan Brown, *Web development with Node and Express*, O'Reilly Media 2014
5. *Configuring Nginx and SSL with Node.js*, <http://www.sitepoint.com/configuring-nginx-ssl-node-js/>, 10.12.2014
6. *Express – api reference*, <http://expressjs.com/4x/api.html>, 10.12.2014
7. *MongoDB*, <http://en.wikipedia.org/wiki/MongoDB>, 10.12.2014
8. *Mongoose API v3.8.19*, <http://mongoosejs.com/docs/api.html>, 10.12.2014
9. *Node.js*, <http://en.wikipedia.org/wiki/Node.js>, 10.12.2014
10. *Node.js v0.10.33 Manual & Documentation*, <http://nodejs.org/api/>, 10.12.2014
11. Guillermo Rauch, *Podręcznik Node.js*, Smashing Magazine 2014
12. *The MongoDB 2.6 Manual*, <http://docs.mongodb.org/manual/>, 10.12.2014

## Spis rysunków

1	Diagram przypadków użycia dla tworzącego ankietę . . . . .	5
2	Diagram przypadków użycia dla respondenta . . . . .	6
3	Okno tworzenia ankiety w Google Docs . . . . .	9
4	Przykładowe okno analizy wyników ankiety w serwisie Ankietka.pl . .	10
5	Oferta serwisu SurveyMonkey.com . . . . .	11
6	Architektura aplikacji . . . . .	13
7	Zastosowanie Nginx jako reverse proxy . . . . .	14
8	Struktura bazy danych . . . . .	15
9	Schemat działania pętli zdarzeń w Node.js . . . . .	17
10	Struktura danych w bazie MongoDB . . . . .	18
11	Diagram klas części serwerowej . . . . .	21
12	Strona internetowa Node.js umożliwiająca pobranie środowiska . . . .	36
13	Strona internetowa MongoDB umożliwiająca pobranie bazy danych .	40
14	Strona rejestracji użytkownika . . . . .	44
15	Tworzenie nowej ankiety . . . . .	45
16	Tworzenie nowej ankiety . . . . .	45
17	Edycja pytań . . . . .	46
18	Udostępnianie ankiety . . . . .	46
19	Analiza ankiety . . . . .	47

## Spis listingów

1	Przykład użycia funkcji zwrotnej w pobraniu danych z bazy danych. .	16
2	Przykład definicji struktury kolekcji w bibliotece Mongoose . . . . .	18
3	Definicja routingu w Node.js . . . . .	22
4	Przykład użycia metody <code>getSurvey</code> . . . . .	28
5	Przykład użycia metody <code>saveAnswers</code> . . . . .	29
6	Przykład użycia metody <code>getUserByEmail</code> . . . . .	31