Wyższa Szkoła Technologii Informatycznych
w Katowicach

Wydział Informatyki
Kierunek Informatyka

Marcin Krawczyk

Nr albumu: 05451

Studia niestacjonarne

Opracowanie i implementacja aplikacji internetowej do zarządzania budżetem osobistym

Praca dyplomowa inżynierska
napisana pod kierunkiem
dr inż. Romana Simińskiego
w roku akademickim 2016/2017

Katowice 2018

Spis treści

1 Wstęp

1.1 Cel pracy

Celem pracy jest zaprojektowanie oraz realizacja systemu do zarządzania przychodami i wydatkami, budżetem osobistym, która pomoże nam odpowiednio dbać o przepływ naszej gotówki. Motywacją do stworzenia tego typu aplikacji była dla mnie chęć raz na zawsze zapanowania nad swoimi wydatkami i procesem zarabiania, wydawania i oszczędzania pieniędzy. Brak w pełni satysfakcjonującego mnie narzędzia dostępnego na rynku był ostatecznym czynnikiem dzięki któremu zdecydowałem, że chcę się pochylić nad tym problemem i zaprojektować aplikację w pełni wyczerpującą moje potrzeby. Jednocześnie to, że istnieje wiele pomniejszych aplikacji do zarządzania budżetem wskazuje, że na takie narzędzie jest zapotrzebowane. Problemem jest niepełność i słaby poziom merytoryczny dostępnych aplikacji, który jest szansą na powodzenia dla mojej implementacji tego problemu.

Główną platformą aplikacji będzie responsywna strona www dostosowująca się do każdego urządzenia. W aplikacji będą zaimplementowane następujące funkcje:

1. Manualne dodawanie wydatków i przychodów
   Wydatek będzie miał właściwości takie jak:
   - nazwa,
   - wartość (cena),
   - ilość sztuk
   - gdzie został zrealizowany
   - czym został zrealizowany (gotówka, karta, przelew, paypall),
   - kiedy,
   - kategoria wydatku.

Przychód będzie miał właściwości takie jak:

- nazwa,
- wartość,
- gdzie,
- przez kogo,
- czym został zrealizowany,
- kiedy,
- kategoria,

2. Posiadając wprowadzone wydatki i przychody będzie można w czasie rzeczywistym przedstawiać zależności między nimi
3. Aplikacja będzie podsumowywała te informacje na interaktywnych wykresach, bazując między innymi na kategoriach, i innych właściwościach
4. Aplikacja będzie miała wydzielony moduł do stworzenia tzw. **Budżetu Osobistego.**
    - Budżet osobisty to nasze oczekiwania, plany jak wyobrażamy sobie dany miesiąc

W Budżecie Osobistym będziemy deklarować nasze oczekiwania zarobkowe a także to ile na daną kategorie wydatków chcemy przeznaczyć pieniędzy, przykładowo:

- Kategoria: **Rozrywka**; deklarujemy 300zł – to znaczy, że w kategorii rozrywka będziemy mogli w danym okresie **Budżetu Osobistego** wydać 300zł łącznie. Mając ustawiony Budżet i wpisując regularnie nasze wydatki, będziemy widzieć jak w czasie rzeczywistym wygląda nasz stosunek tego ile zostało pieniędzy do wydania w tej kategorii, ile już wydaliśmy i ile czasu zostało do nowego okresu Budżetu Osobistego.

1.2 Zawartość pracy

Praca składa się z 9 rozdziałów. W niniejszym rozdziale znajduje się krótki wstęp oraz, opis zawartości pracy a także krótka informacja na temat języków, frameworków i technologii jakie wykorzystam w celu stworzenia oprogramowania. W drugim rozdziale zostaje przedstawiona analiza tematu, motywacji do podjęcia się rozwiązania problemu finansów. W rozdziale trzecim znajduje się analiza dostępnych aplikacji na rynku polskim i światowym.

1.3 Języki, frameworki, technologie:

W trakcie developmentu będę korzystał z systemu kontroli wersji GIT, serwera VPS, pracował w środowisku IDE – **Visual Studio Code**, korzystał z systemu do zarządzania zadaniami **Redmine**. Będę także korzystał z metod wirtualizacji opartych na **Dockerze** używając **docker-compose**.

Jeśli chodzi o zastosowane języki programowania i technologie z nimi związane, wykorzystam:

1. **Frontend:** Angular, HTML5, CSS3 (SCSS), ChartJS
   Backend: NestJS który używa ExpressJS + TypeScript. Jako bazę danych użyję MongoDB z wykorzystaniem mongoose.

2 Charakterystyka/analiza problemu

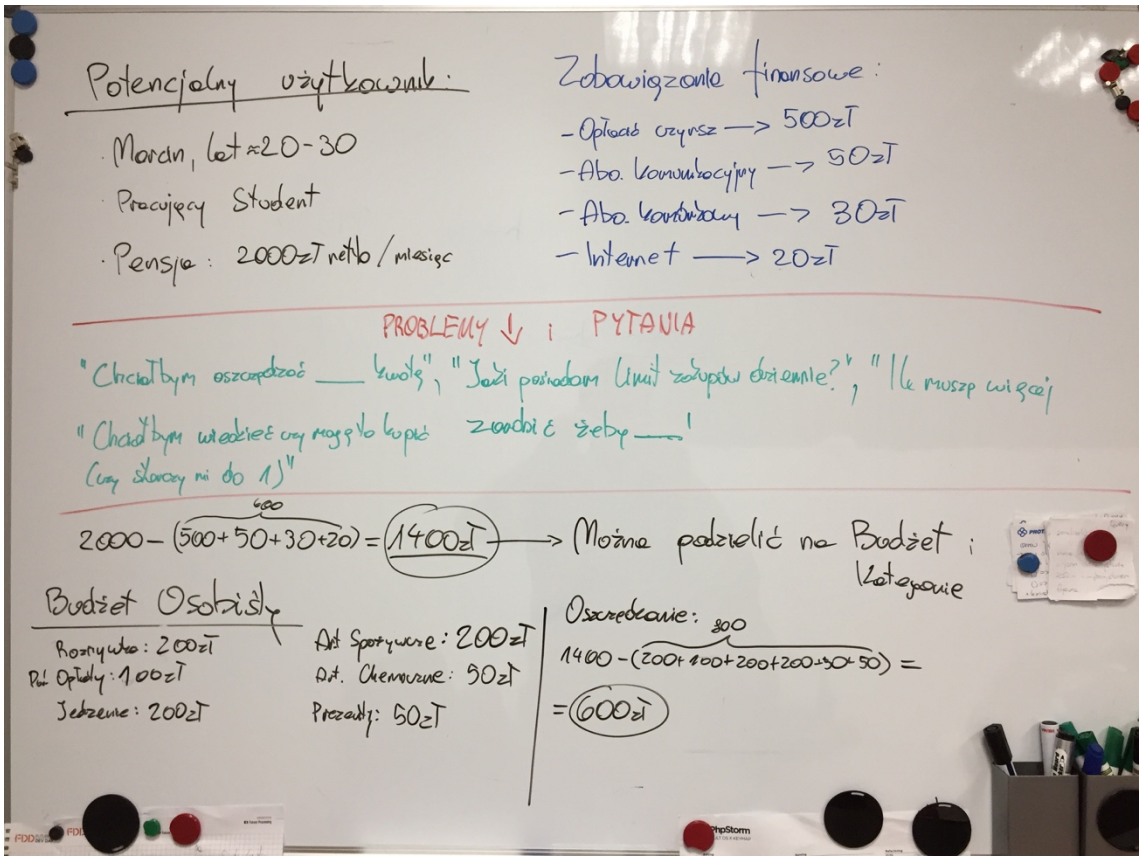
Kwestia pieniędzy zawsze jest kwestią dość drażliwą. Tak naprawdę większość z nas ma wystarczającą ilość pieniędzy na wykonywanie wielu różnych działań. Po prostu w wielu przypadkach nie potrafimy z tych pieniędzy w sensowny sposób korzystać. Nasze finansowe środki znikają bardzo szybko i często nie mamy pojęcia gdzie i dlaczego wyparowały. Co się wydarzyło, że nagle z naszej pensji w połowie miesiąca nie zostaje nic, a mamy wrażenie, że nic wielkiego nie kupowaliśmy.

W zarządzaniu finansami różni ludzie oczekują różnych rezultatów, szukają innych profitów. Są jednostki, które potrzebują spisywać wydatki bo nie wiedzą gdzie uciekają ich pieniądze. Kiedy spisujemy takie rzeczy, na początku możemy nie widzieć ewidentnych plusów, jednak po miesiącu, dwóch jesteśmy w stanie wyśledzić na jakie rozrywki, na jakie niepotrzebne sprawy wydaliśmy pieniądze. Bardzo często okazuje się, że wydajemy bardzo duże pieniądze na bardzo nieistotne zbytki. Często widzimy także, że to te małe zakupy, ale w powtarzającym się schemacie najbardziej wpływają na nasze braki finansowe.

Są także ludzie, którzy chcą by oprogramowanie automatyzowało za nich decyzje zakupowe. Czyli na przykład, po szybkim spojrzeniu na telefon widzimy, czy zdefiniowany budżet na zakupy ubrań i obuwia nie został już przekroczony, czy może spokojnie możemy pozwolić sobie na jeszcze jedną parę butów w tym miesiącu.

Oszczędzanie na konkretne cele bądź po prostu oszczędzanie, także jest celem mojej aplikacji. Kiedy mamy świadomość na co wydajemy pieniądze, kiedy ustalamy budżet i wiemy ile mamy do rozdysponowania funduszy na określone kategorie, możemy także tak przemyśleć nasz budżet, żeby zostawić trochę pieniędzy na oszczędności. Dzięki kontroli i przypomnieniom naszej aplikacji, będziemy wiedzieli dlaczego w tym miesiącu nie wydajemy pieniędzy na określone dobra.

Tutaj dodac trochę rysunkow koncepcyjnych, ideowych i napisac coś jeszcze na temat samego problemu finansów.

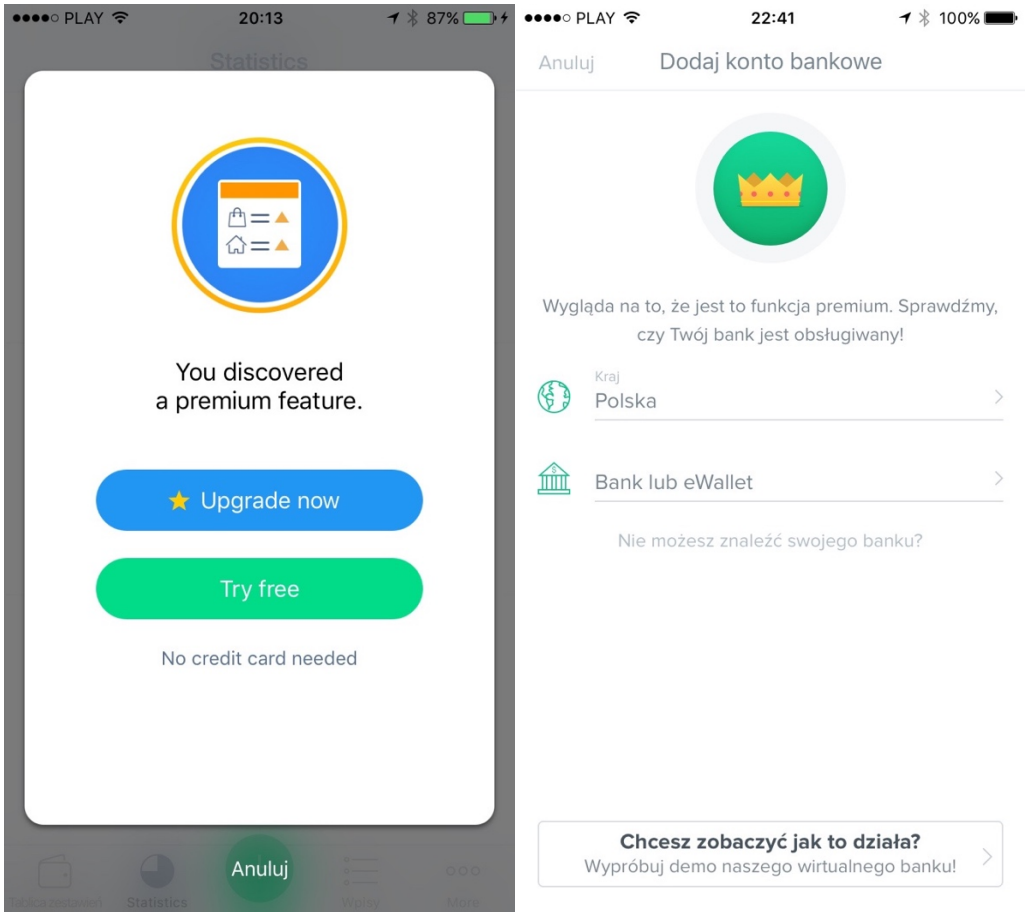

Potencjalny użytkownik:
· Mężczyzna, lat ≈ 20-30
· Pracujący Student
· Pensja: 2000zł netto / miesiąc
Zobowiązania finansowe:
- Opłaty czynsz —> 500zł
- Abo. komunikacyjny —> 50zł
- Abo. kawalerski —> 30zł
- Internet —> 20zł
PROBLEMY ↓ i PYTANIA
"Chciałbym oszczędzać ___ kwotę", "Jaki posiadam limit zakupów obecnie?", "Ile muszę wiązać
"Chciałbym wiedzieć czy mogę to kupić zanadto żeby ___!
(czy starczy mi do 1)"
2000 − (500+50+30+20) = 1400zł —> Można podzielić na Budżet i Kategorie
Budżet Osobisty
Rozrywka: 200zł
Pół Opłaty: 100zł
Jedzenie: 200zł
Art Sportowe: 200zł
Art. Chemiczne: 50zł
Prezenty: 50zł
Oszczędzanie:
1400 − (200+100+200+200+50+50) =
= 600zł

3 Analiza istniejących rozwiązań

Istniejących rozwiązań jest sporo. Na urządzenia typu desktop, na smartfony, tablety, urządzenia posiadające dostęp przez przeglądarkę. Jednak żadne z nich w pełni mnie nie usatysfakcjonowało. Każde z nich miało braki w funkcjach, które uważam za potrzebne, a kiedy dana aplikacja posiadała funkcje A, to nie posiadała funkcji B i tak naprzemiennie. Przykładowo, jedna aplikacja posiada funkcje tworzenia budżetu w różnym okresie czasu, ale nie posiada funkcji wyświetlania dobrze sformatowanych wykresów które są w stanie pokazać nam wiele zależności pomiędzy naszymi finansami. W innym przypadku, takie wykresy można było nawet samemu tworzyć, ale okazywało się, że budżet można ustawić tylko raz i to tylko na miesiąc, więc pewnej funkcjonalności brakowało. Bardzo często spotykałem się z aplikacjami, które rozwiązywały tylko część problemu związanego z zarządzaniem finansami osobistymi. Przykładowo dawały dostęp tylko do wpisywania wydatków, przychodów i nic poza tym.

Rysunek 1 Aplikacja Cents służy tylko do wpisywania wydatków i przychodów, oprócz tego jest zablokowana do kilkunastu wpisanych wydatków, możliwość wpisania większej ilości jest dostępna w wersji płatnej PREMIUM

Wiele aplikacji które testowałem miało bardzo wybrakowaną funkcjonalność w wersji darmowej. Dopiero po wykupieniu abonamentu PREMIUM za bardzo drogie pieniądze, można było liczyć na funkcje synchronizowania danych o wydatkach i dochodach poprzez zalogowanie do swojego banku (która także była dostępna tylko w niektórych bankach).



Rysunek 2 Po lewej aplikacja Spendee w której dodanie konta bankowego jest dopiero dostępne w funkcji PREMIUM, w Polsce, jedynie 3 banki obsługiwane. Po prawej aplikacja Wallet, podobnie.

Duża ilość aplikacji, które analizowałem, nie może się też poszczycić zbyt profesjonalnym i zachęcającym wyglądem. Często zaprojektowanie grafik dla narzędzia do zarządzania finansami jest infantylne i dość dziecinne.

Rysunek 3 Aplikacja Cents nie wzbudza zaufania. (a aplikacja zarządzająca naszymi pieniędzmi powinna)

Podsumowaniem do tego rozdziału może być stwierdzenie, że po prostu na rynku aplikacji panuje pewnego rodzaju chaos. Jest bardzo dużo narzędzi do zarządzania własnymi finansami. Ale nie ma żadnej aplikacji, której bym na dłużej zaufał, która miała by wszystkie funkcje potrzebne. A nawet jeśli znajdzie się aplikacja która jest prawdziwym skarbem i jest kompletna to jej tzw. „User Experience" jest bardzo zły i aplikację rzucamy w kąt bo nie jesteśmy w stanie z niej szybko i przyjemnie korzystać. W dzisiejszym zabieganym, dynamicznym świecie często właśnie łatwość użytkowania i prostota przyciąga do nas klientów i daje nam wygrywającą pozycję i polecenia od zadowolonych użytkowników.

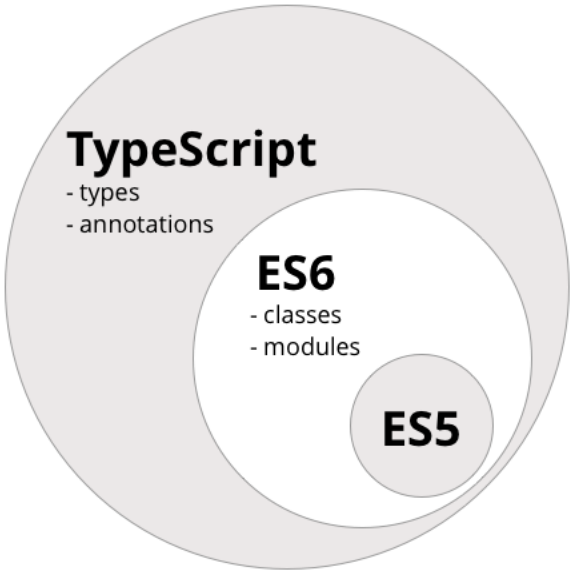
Moim głównym celem jest zaproponowanie użytkownikowi kompleksowego, aczkolwiek intuicyjnego systemu do zarzadzania finansami. Dość dobrym przykładem przyjętej metodyki jest zdanie **„Easy to began, harder to master"**. Łatwo rozpocząć, ciężej być mistrzem. Chciałbym poprowadzić użytkownika poprzez proste funkcje które dadzą mu od razu kontrolę nad swoim

bilansem finansowym powoli wprowadzając w bardziej zaawansowane narzędzia kontroli, oszczędzania, budżetowania.

4 Koncepcja własnego rozwiązania.

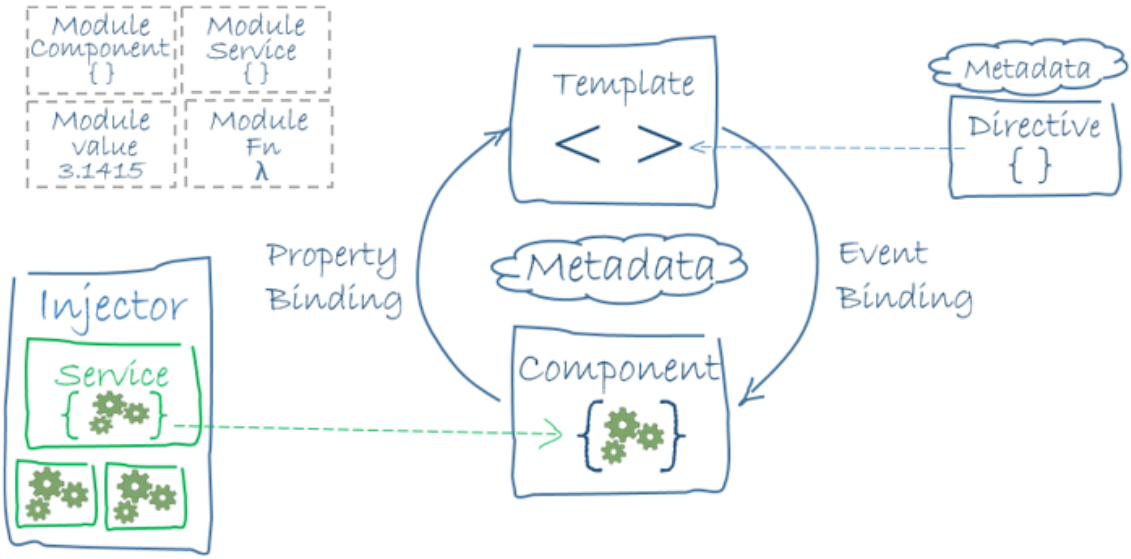
4.1 Frontend

Do zrealizowania aplikacji do kontrolowania wydatków postanowiłem wykorzystać aktualne najnowsze technologie jakie można spotkać w świecie programistów webowych. Aplikacja istnieje w środowisku przeglądarkowym więc oczywistym wyborem po stronie klienckiej będzie język **JavaScript.** Do tworzenia kolejnych widoków a także strony wizualnej wykorzystałem duet **HTML + CSS.** Oczywiście, w obu przypadkach koncentruję się na korzystaniu z najnowszych ich wersji czyli HTML w wersji 5 i CSS w wersji 3. Dodatkowym zwiększeniem możliwości języka CSS jest rozszerzenie go o tzw. Preprocesor – **SASS**. Dodaje on nam możliwość deklarowania zmiennych, tworzenia funkcji i większej ilości reużywalnych bloków kodu. Na użycie tego preprocesora także zdecydowałem się w pracy inżynierskiej. Jednak sam czysty JavaScript, HTML i CSS to za mało by móc umożliwić łatwe skalowanie aplikacji i jej potencjalny przyszły rozwój w zrównoważonym środowisku. Dlatego zdecydowałem się na użycie frameworka **Angular** w wersji **2+** tworzonego głównie przez firmę **Google.** Dzięki temu rozwiązaniu jestem w stanie dzielić aplikację na logiczne części, które w trakcie pracy można bezproblemowo dopisywać. Te części to **komponenty** na które składa się: widok napisany w HTMLu (jest to HTML poszerzony o dodatkowe dyrektywy i konstrukcje dostarczane przez Angulara), wygląd komponentów definiowany jest w dołączonym pliku CSS, z kolei logika komponentu jest pisana jako eksportowana klasa z wykorzystaniem języka **TypeScript**. TypeScript to tak naprawdę język JavaScript rozszerzony o możliwość deklarowania typów, interfejsów i innych możliwości znanych z języków takich jak C#. Wybór padł na TypeScript, ponieważ jest on domyślnie wspierany przez Angular, a sam Angular napisany jest właśnie z wykorzystaniem powyższego.



Rysunek 4 Czym jest TypeScript

Architektura aplikacji Angularowej jest przedstawiona na poniższym obrazku pobranym z oficjalnej dokumentacji frameworka.



Rysunek 5 Architektura aplikacji Angular

Oprócz wspomnianych wyżej komponentów drugim najważniejszym mechanizmem Angulara są tzw. Serwisy. Służą one do przechowywania logiki naszej aplikacji i odciążanie komponentów. W komponentach przechowujemy logikę związaną z tymi konkretnymi komponentami, zaś serwisy służą do

przechowywania logiki na wyższym poziomie abstrakcji. Bardzo częstym przypadkiem użycia serwisu jest udostępnianie za jego pomocą interfejsu do wykonywania zapytań HTTP do czego między innymi ja używam serwisów w mojej aplikacji. Sam serwis to klasa udostępniająca określoną funkcjonalność która dzięki zastosowaniu dekoratora @Injectable jest w stanie być wstrzykiwana do jakiegokolwiek komponentu z wykorzystaniem wzorca wstrzykiwania zależności.

4.2 Backend

W mojej pracy inżynierskiej oprócz części klienckiej stworzyłem także część odpowiadającą za wszelkie akcje wykonujące się po stronie serwera. W tym miejscu pozwoliłem sobie na mały eksperyment i połączyłem kilka elementów by stworzyć dobrze funkcjonujący i skalowalny backend.

Po pierwsze użyłem frameworka **NestJS** stworzonego przez polskiego programistę **Kamila Myśliwca**, który garściami czerpie z wzorców przedstawionych w Angularze. Między innymi takich jak wstrzykiwanie zależności, czy też tworzenie komponentów. Oprócz tego, framework daje nam możliwość tworzenia kontrolerów odpowiadających między innymi za wystawianie RESTowych końcówek przez które można zwracać i przekazywać dane. Jednocześnie, **NestJS** korzysta z biblioteki **Express.js**, która oparta jest na **Node.js** czyli implementacji języka **JavaScript** w środowisku serwerowym. Tutaj także pokusiłem się o ulepszenie języka jego typowaną wersją czyli skonfigurowałem wykorzystanie języka **TypeScript.**

4.3 Przechowywanie danych

Do zapisywania i przechowywania danych wykorzystałem nierelacyjną bazę danych **MongoDB**. Powodem wyboru tej technologii była bardzo duża ilość materiałów i pomocy na temat tejże bazy danych. Swego czasu jednym z bardzo popularnych stosów technologicznych do wytwarzania oprogramowania był tak zwany MEAN stack. Rozwinięciem tego akronimu jest: **M**ongo, **E**xpress, **A**ngular, **N**ode. Postanowiłem więc wykorzystać ten stos technologiczny poszerzając go o własną konfigurację (wspomniany NestJS i wykorzystanie języka TypeScript). By

nadać trochę zasad i uporządkowania do bazy danych opartej na MongoDB wykorzystałem bibliotekę ODM – (Object Data Modeling – modelowanie danych obiektowych) – **mongoose**. Zapewnia ona rygorystyczne środowisko do modelowania danych, wymuszając strukturę, przy jednoczesnym zachowaniu elastyczności.

5 Projekt ogólny

5.1 Specyfikacja wymagań funkcjonalnych i niefunkcjonalnych

5.2 Architektura systemu

5.2.1 JavaScript

5.2.2 Angular

5.2.3 HTML5 i CSS3

5.2.4 NestJS

5.2.5 ExpressJS

5.2.6 TypeScript

5.2.7 MongoDB

5.2.8 Mongoose

5.2.9 ChartJS

5.3 Narzędzia programistyczne

5.3.1 Visual Studio Code

5.3.2 Postman

5.3.3 Git

5.4 Metody i narzędzia realizacji

5.5 Koncepcja przechowywania danych

5.6 Projekt interfejsu użytkownika

6 Projekt techniczny

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed a metus nibh. Curabitur sit amet molestie nulla, ut porttitor tellus. Sed faucibus congue augue, sit amet dictum justo dapibus quis. Fusce iaculis efficitur arcu, eget volutpat est posuere sit amet. Curabitur semper orci ac purus aliquet, id molestie dolor lacinia. Mauris pharetra ullamcorper orci, at vehicula nisl lacinia a. Curabitur quam turpis, dapibus sit amet hendrerit ut, eleifend vitae purus. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Curabitur sollicitudin volutpat ante. Praesent faucibus tortor in semper cursus. In hac habitasse platea dictumst. Nulla quis accumsan odio, pellentesque aliquam nibh. Praesent vel lectus quam. Praesent ornare velit sit amet egestas laoreet. Cras feugiat metus a tincidunt egestas. Pellentesque erat sapien, vestibulum nec lectus id, posuere congue nisi.

7 Testy i weryfikacja systemu

8 Przykładowy scenariusz wykorzystania systemu

9 Zakończenie

10 Bibliografia

1. Autor anonimowy, Tworzenie bibliografii pracy dyplomowej [on-line], Polski Portal Edukacyjny, http://www.e_nauka.pl, [dostępne: 4 kwietnia 2016].
2. Kowalski J., Jak cytować materiały źródłowe, Oficyna Wydawnicza Example, Katowice, 2014.

11 Spis rysunków

12 Spis tabel