|  |
| --- |
| **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 |

Ta strona powinna być pusta. Uwaga, ten tekst jest ukryty i nie będzie widoczny na wydruku.

Spis treści

[1 Wstęp 3](#_Toc512382062)

[2 Charakterystyka/analiza problemu 4](#_Toc512382063)

[3 Analiza istniejących rozwiązań 6](#_Toc512382064)

[4 Koncepcja własnego rozwiązania. 10](#_Toc512382065)

[5 Projekt ogólny 14](#_Toc512382066)

[5.1 Specyfikacja wymagań funkcjonalnych i niefunkcjonalnych 14](#_Toc512382067)

[5.2 Architektura systemu 15](#_Toc512382068)

[5.2.1 Warstwa kliencka 15](#_Toc512382069)

[5.2.2 Warstwa serwerowa 17](#_Toc512382070)

[5.2.3 X§ Przechowywanie danych 17](#_Toc512382071)

[5.2.4 JavaScript 18](#_Toc512382072)

[5.2.5 Angular 18](#_Toc512382073)

[5.2.6 HTML5 i CSS3 18](#_Toc512382074)

[5.2.7 NestJS 18](#_Toc512382075)

[5.2.8 ExpressJS 18](#_Toc512382076)

[5.2.9 TypeScript 18](#_Toc512382077)

[5.2.10 MongoDB 18](#_Toc512382078)

[5.2.11 Mongoose 18](#_Toc512382079)

[5.2.12 ChartJS 18](#_Toc512382080)

[5.3 Narzędzia programistyczne 18](#_Toc512382081)

[5.3.1 Visual Studio Code 18](#_Toc512382082)

[5.3.2 Postman 18](#_Toc512382083)

[5.3.3 Git 18](#_Toc512382084)

[5.4 Metody i narzędzia realizacji 19](#_Toc512382085)

[5.5 Koncepcja przechowywania danych 20](#_Toc512382086)

[5.6 Projekt interfejsu użytkownika 21](#_Toc512382087)

[6 Projekt techniczny 22](#_Toc512382088)

[7 Testy i weryfikacja systemu 23](#_Toc512382089)

[8 Przykładowy scenariusz wykorzystania systemu 24](#_Toc512382090)

[9 Zakończenie 26](#_Toc512382091)

[10 Bibliografia 28](#_Toc512382092)

[11 Spis rysunków 29](#_Toc512382093)

[12 Spis tabel 30](#_Toc512382094)

Ta strona powinna być pusta. Uwaga, ten tekst jest ukryty i nie będzie widoczny na wydruku.

# Wstęp

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 funkcje dodawanie wydatków i przychodów, zestawiania zależności zachodzących pomiędzy nimi na interaktywnych wykresach, bazując między innymi na kategoriach wydatków i przychodów oraz innych zdefiniowanych właściwościach. Aplikacja będzie miała wydzielony moduł do stworzenia tzw. *Budżetu Osobistego*, opisującego nasze oczekiwania, plany jak wyobrażamy sobie dany miesiąc w sensie wydatków i przychodów. W *Budżecie Osobistym* będziemy deklarować nasze oczekiwania zarobkowe a także to ile na daną kategorie wydatków chcemy przeznaczyć pieniędzy.

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.

# 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.

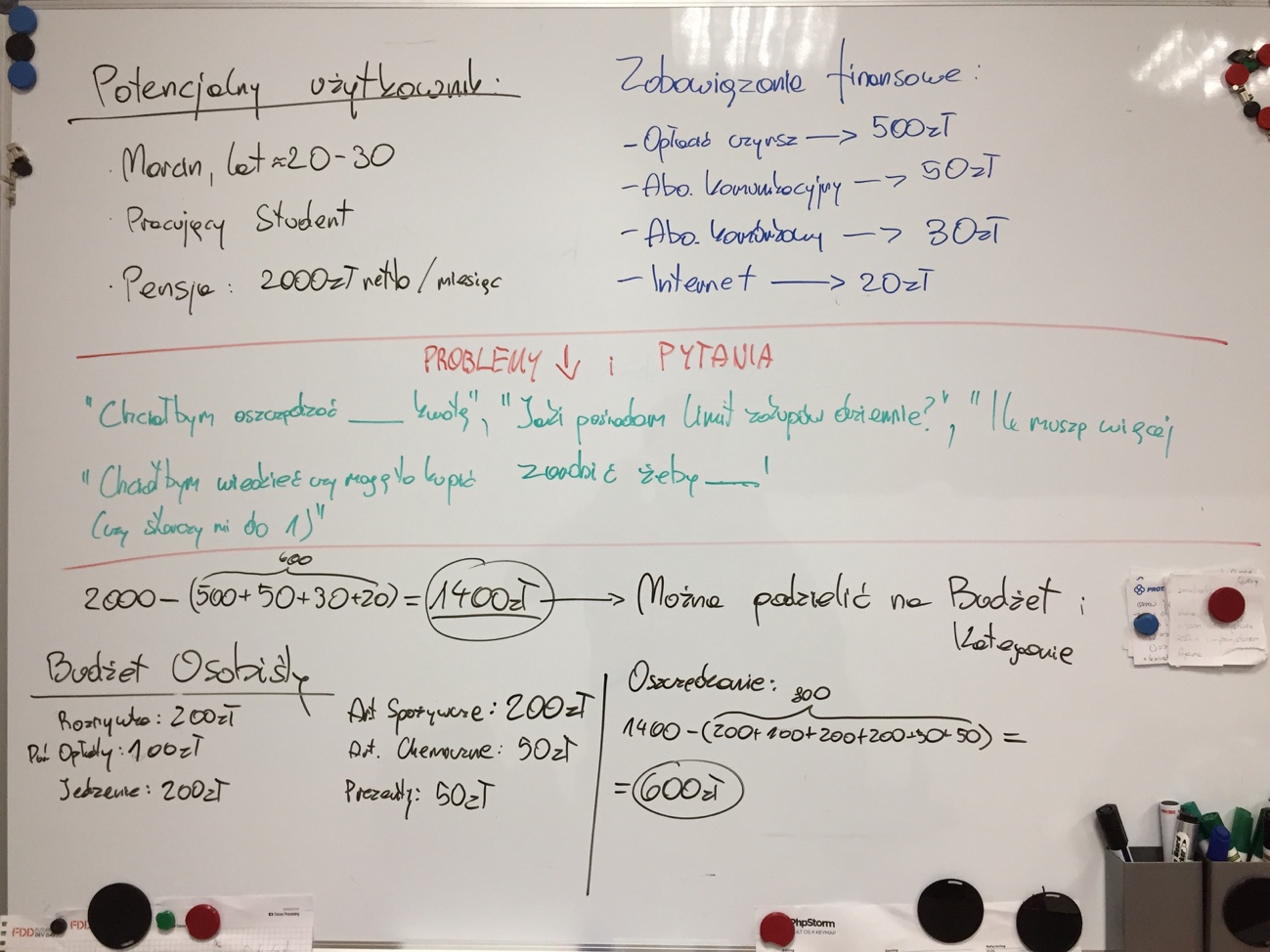
W zarządzaniu finansami różni ludzie oczekują różnych rezultatów, szukają innych profitów. Są jednostki, które potrzebują spisywać wydatki bo nie wiedzą gdzie „uciekają” ich pieniądze. Kiedy spisujemy takie rzeczy, na początku możemy nie widzieć ewidentnych plusów, jednak po miesiącu, dwóch jesteśmy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

Są także ludzie, którzy chcą by oprogramowanie automatyzowało decyzje zakupowe. Przykładowo, 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.

Bardzo długo zastanawiałem się nad różnymi wariantami tego problemu. Nad tym jakie zagadnienia związane z problemem finansów mogę podjąć. Wyobraziłem sobie każdą osobę z osobna, która może używać aplikacji.

Studenta, który musi bardzo sprawnie zarządzać niewielką ilością gotówki. Biznesmana który tej gotówki ma więcej, ale także ma różnego rodzaju zobowiązania finansowe, które musi zaadresować. Poniżej rozpisałem przykładowo sylwetkę pracującego studenta i jego oczekiwania.

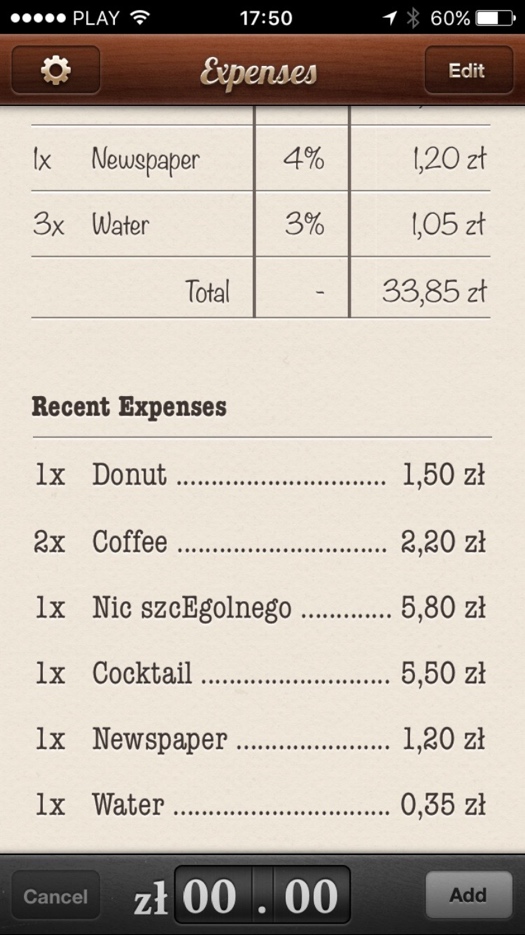


Rysunek Zobrazowanie problemu budżetu osobistego

Dzięki mojej aplikacji, tenże student, będzie w stanie na bieżąco monitorować swoje wydatki, swoje przychody. Będzie mógł widzieć zależność pomiędzy tym ile zostało z ostatniej pensji, a tym na co wydaje posiadane pieniądze. Będzie mógł w łatwy sposób, zobaczyć na wykresach zależności pomiędzy kategoriami do których przypisuje wydatki. A także, po zdefiniowaniu budżetu, widzieć na bieżąco na ile jeszcze może sobie pozwolić w tym miesiącu w określonych kategoriach. Jego życie stanie się bardziej poukładane, i zyska potencjał do oszczędzania nawet posiadając relatywnie mało środków.

# Analiza istniejących rozwiązań

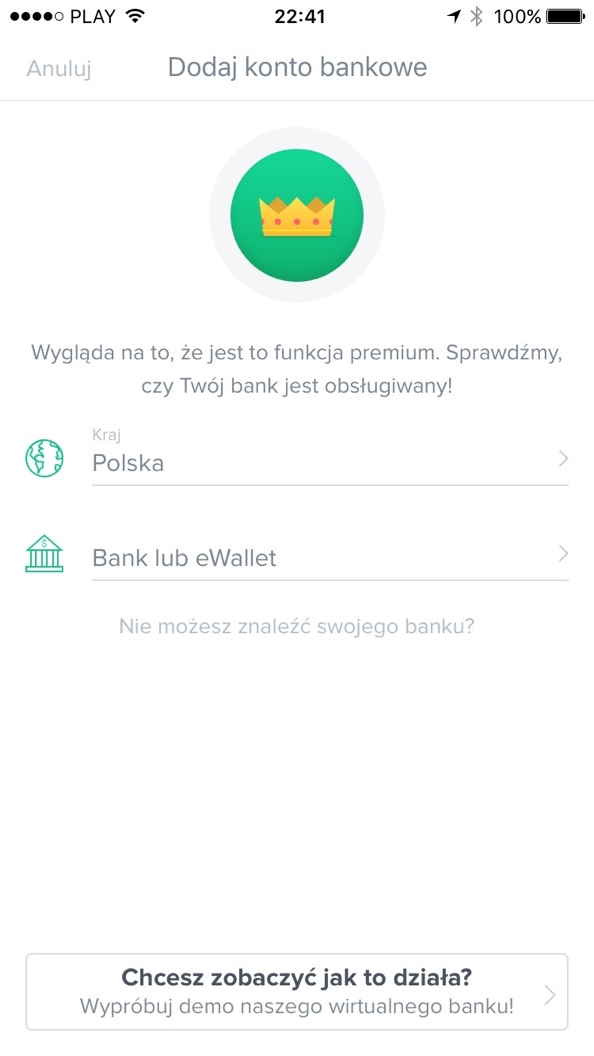
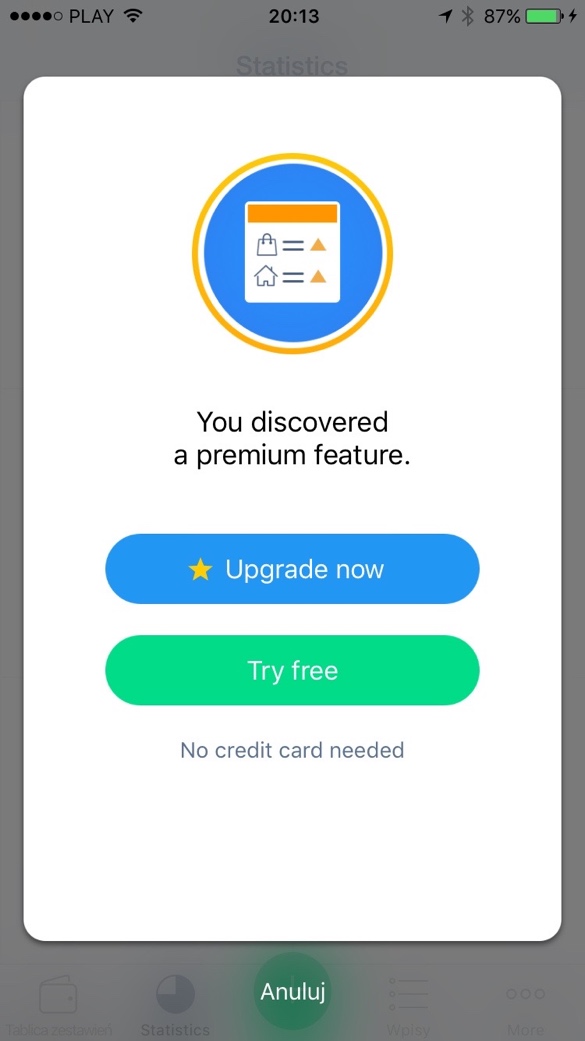
Istniejących rozwiązań jest sporo. Dla urządzeń typu desktop, na smartfony, tablety, urządzenia posiadające dostęp przez przeglądarkę. Jednak żadne z nich w pełni mnie nie usatysfakcjonowało. Każde z nich miało braki w funkcjach, które uważam za potrzebne, a kiedy dana aplikacja posiadała funkcje A, to nie posiadała funkcji B i przeciwnie. 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 Aplikacja Cents

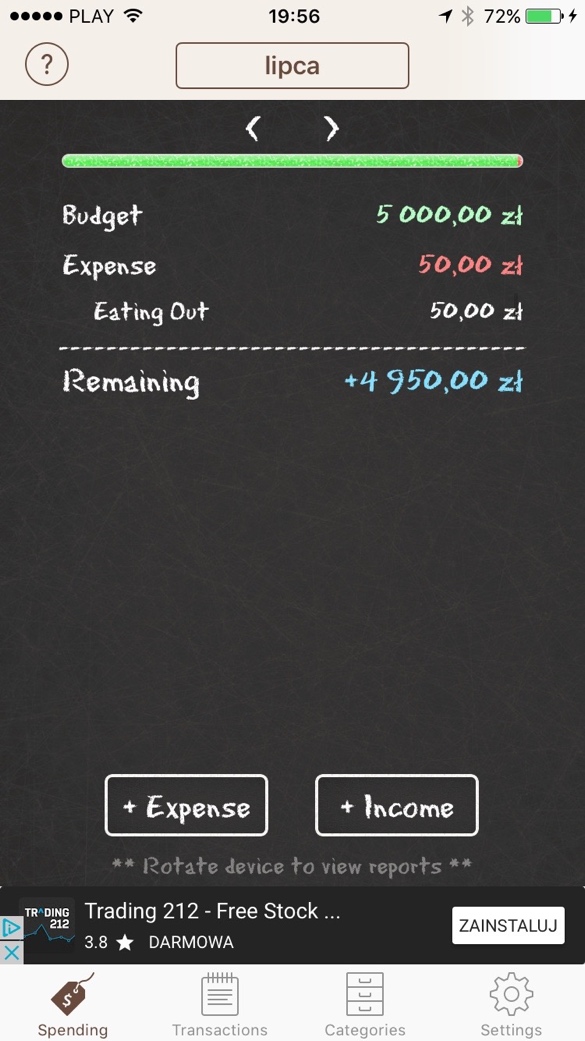
Aplikacja Cents ssłuży tylko do wpisywania wydatków i przychodów, oprócz tego jest zablokowana do kilkunastu wpisanych wydatków, możliwość wpisania większej ilości jest dostępna w wersji płatnej PREMIUM

Wiele aplikacji które testowałem miało bardzo uproszczone funkcje w wersji darmowej. Dopiero po wykupieniu abonamentu PREMIUM, 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 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 Aplikacja Cents nie wzbudza zaufania. (a aplikacja zarządzająca naszymi pieniędzmi powinna)

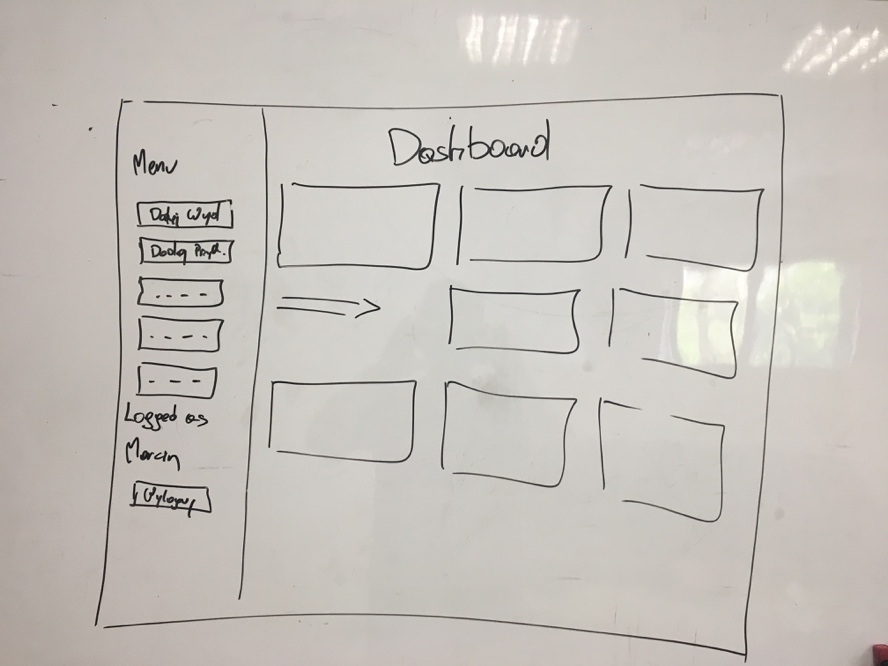
Podsumowaniem do tego rozdziału może być stwierdzenie, że po prostu na rynku aplikacji panuje pewnego rodzaju chaos. Jest bardzo dużo narzędzi do zarządzania własnymi finansami. Ale nie ma żadnej aplikacji, której bym na dłużej zaufał, która miała by wszystkie funkcje potrzebne. A nawet jeśli znajdzie 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.

# Koncepcja własnego rozwiązania.

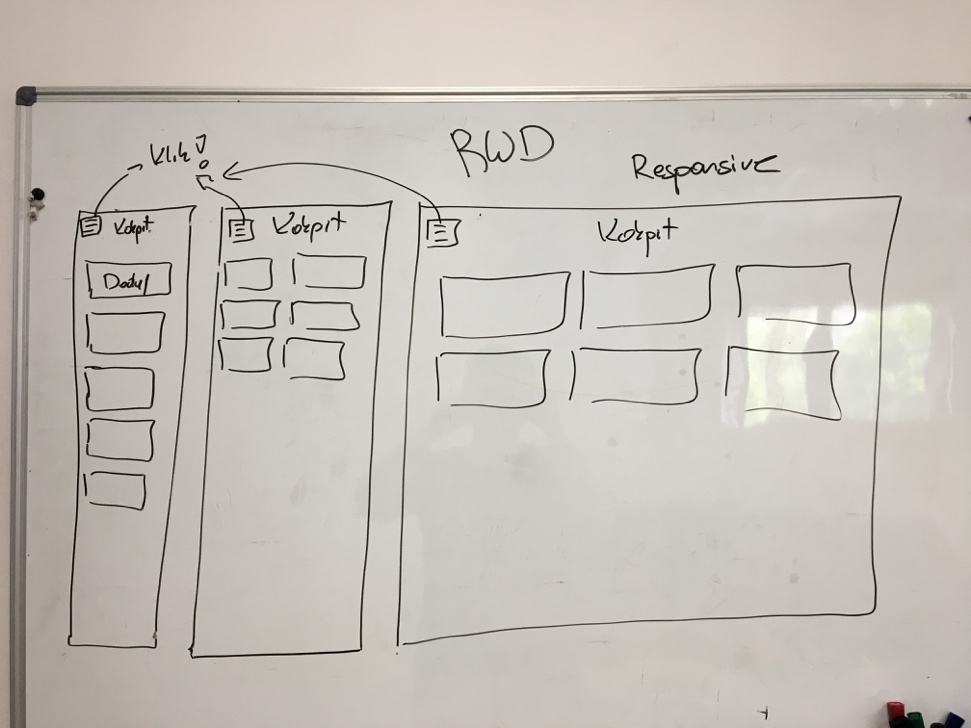
Widzimy już, że zarówno problem panowania na wydatkami i przychodami jest problemem, który wciąż jest nierozwiązany. Firmy próbują w różnoraki sposób podejść do tematu, jednak wciąż brakuję tu dobrego i prostego rozwiązania.

Moja propozycja to aplikacja **webowa**, która będzie wysoko dostępna i multiplatformowa dająca nam natychmiastowy dostęp do naszego budżetu. Postanowiłem wykorzystać potęge przeglądarek internetowych i najnowszych standardów by dotrzeć do jak największej ilości ludzi. Aplikacja **PersonalBudget** będzie bazować na interfejsie użytkownika stworzonym w technologiach takich jak **HTML, CSS, JavaScript**. To co było dla mnie istotne w trakcie projektowania i implementowania funkcji aplikacji była łatwa dostępność do poszczególnych elementów aplikacji. Wykorzystując gotowe komponenty graficzne mogłem w łatwy sposób dać użytkownikowi dostęp do rozsuwanego menu bocznego. W tymże użytkownik ma dostęp do wszystkich modułów aplikacji.



Rysunek – wysuwane boczne menu

Oczywiście zależało mi także na tym, by aplikacja była używalna na każdym urządzeniu jakie może mieć dostęp do internetu i przeglądarki, więc zaprojektowałem ją tak, by zależnie od rozdzielczości urządzenia wciąż była używalna i pokazywała te informację, które są kluczowe.

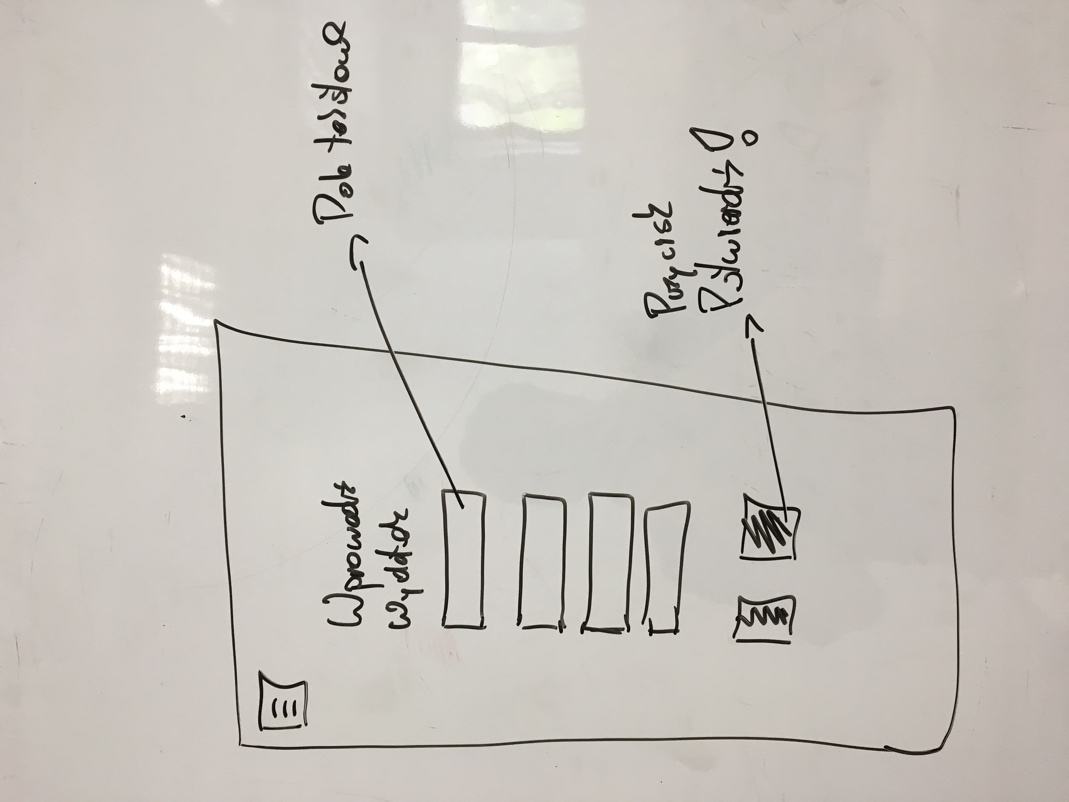


Rysunek – zasady responsywności – dostępność na każdym urządzaniu

W moim rozwiązaniu skupiam się na 3 najważniejszych aspektach.

1. **Zapisywanie**
2. **Budżetowanie**
3. **Analizowanie**

Pierwszy z nich, czyli **Zapisywanie** jest tak naprawdę najważniejszym z całej trójcy. Niestety, żeby móc stworzyć budżet a także analizować nasze wydatki i wynieść z tego jakąś korzyść, musimy najpierw stworzyć bazę informacji o nas, o naszych wydatkach, tak naprawdę o naszych zachowaniach. Poprzez szereg podstron umożliwiających dodawanie wydatków i przychodów jest realizowana ta funkcjonalność. W pracy inżynierskiej skupiam się jedynie na manualnym sposobie dodawania wydatków – w momencie wykonania zakupu, kierujemy się do aplikacji, wchodzimy na podstronę związana z dodawaniem wydatku i wypełniamy wskazane pola.



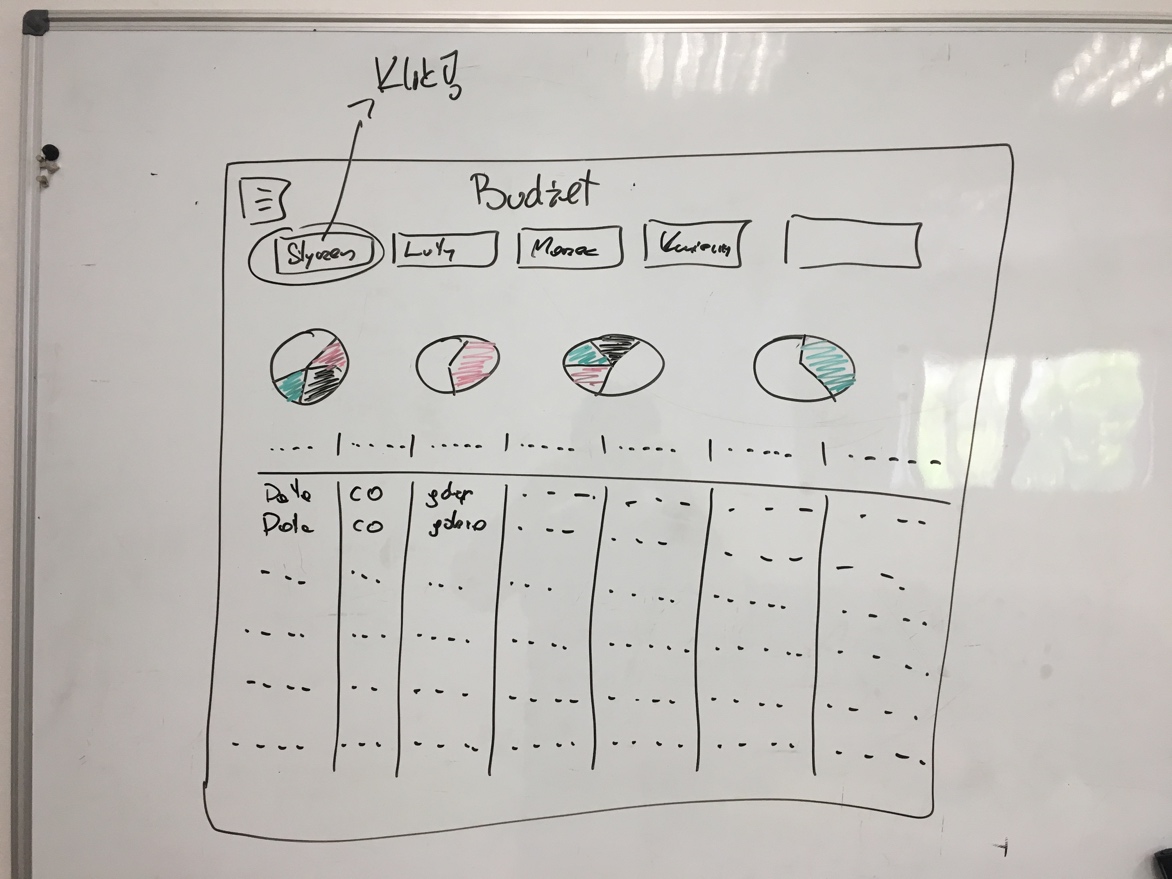
Rysunek – schemat dodawania wydatku/przychodu/budżetu

W tym miejscu, można by było pokusić się o implementacje rozwiązania pół automatycznego. Aktualne mechanizmy rozpoznawania pisma, rozpoznawania tekstu na obrazkach pozwoliły by na stworzenie modułu do aplikacji, który z sukcesem mógłby rozpoznać dużą ilość informacji z paragonu. Robiłoby się zdjęcie paragonu a aplikacja sama podjęłaby próbę dopasowania poszczególnych kawałków tekstów do pól tekstowych w podstronie z dodawaniem wydatku. Jednak niepełność rozwiązania i ograniczone zasoby czasowe nie pozwoliły mi dłużej pochylać się nad takim rozwiązaniem.

Kolejnym z najważniejszych podpunktów jest **Budżetowanie.** W sytuacji kiedy wyrobiliśmy już sobie nawyk zapisywania wydatków i korzystamy z prostego formularza w aplikacji, możemy dodać do tego tworzenie budżetu. Postanowiłem ułatwić mechanizm budżetowania tylko do stworzenia budżetu – zdefiniowanie nazwy budżetu, zdefiniowanie daty początku okresu i końca okresu, i zdefiniowanie poszczególnych kategorii w budżecie na podstawie których będzie można zobaczyć na interaktywnych wykresach na jakie wydatki jesteśmy w stanie sobie jeszcze pozwolić w tym okresie budżetowania. Po tym jak stworzymy budżet, możemy w trakcie dodawania jakiegokolwiek wydatku wybrać dodatkowo, do jakiej kategorii w jakim budżecie chcemy ten wydatek przypisać.

Co prowadzi nas do ostatniej z najważniejszych cech mojego oprogramowania czyli **Analiza.** Dzięki tak zapisywanym danym o naszych dochodach i wydatkach mamy dostęp do aktualizowanych w czasie rzeczywistym wykresów którą pokazują nam zależności pomiędzy poszczególnymi rodzajami wydatków.

Na poniższym rysunku, widzimy jedno z miejsc w których wykorzystane będą te wykresy, czyli na podstronie gdzie zarządzamy naszymi budżetami. Możemy wybrać konkretny budżet i analizować dane z nim związane.



Rysunek – podstrona budżetu osobistego z interaktywnymi wykresami

# Projekt ogólny

## Specyfikacja wymagań funkcjonalnych i niefunkcjonalnych

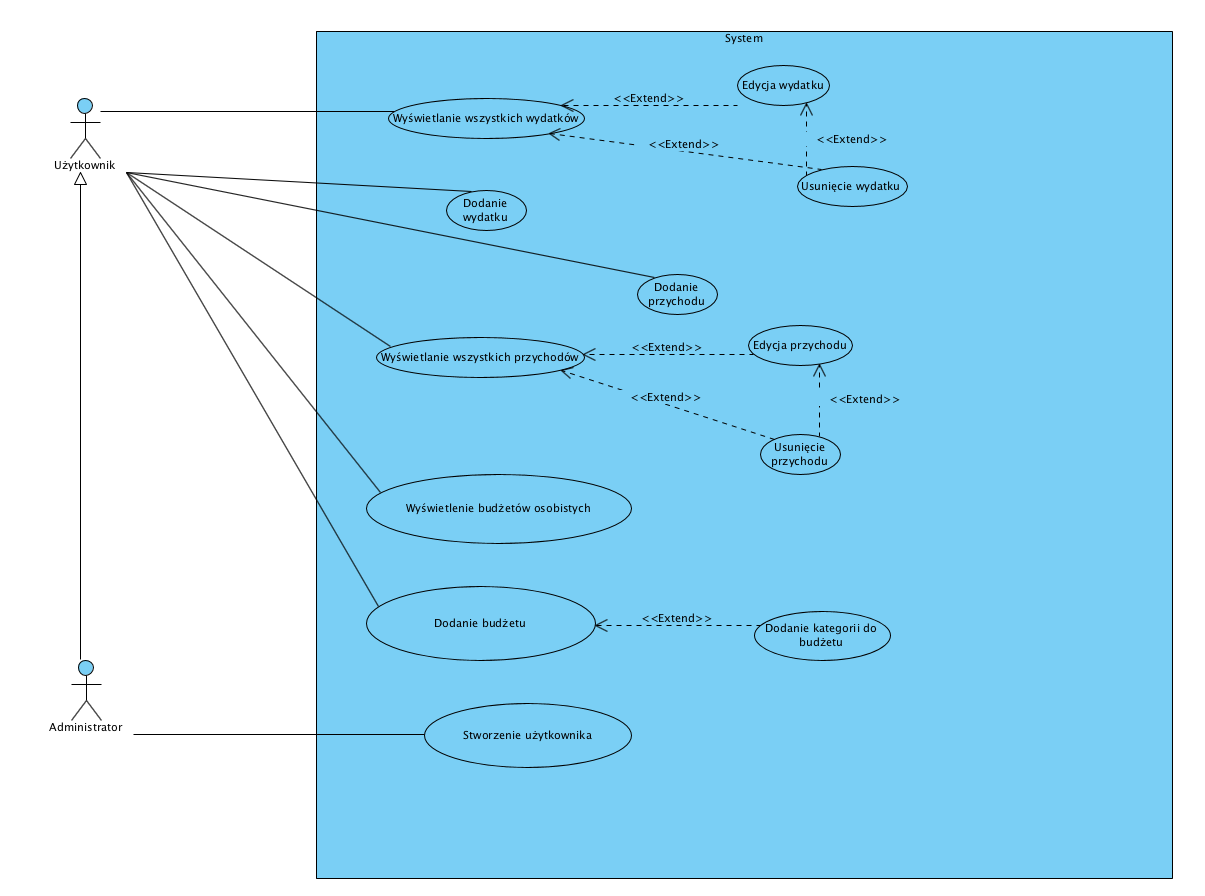
### Wymagania funkcjonalne

1. Zarejestrowanie użytkownika.
2. Bezpieczne zalogowanie użytkownika.
3. Dodanie wydatku.
4. Edycja wydatku.
5. Wyświetlenie listy wydatków.
6. Usunięcie wydatku.
7. Dodanie przychodu.
8. Edycja przychodu.
9. Wyświetlenie listy przychodów.
10. Usunięcie przychodu.
11. Dodanie budżetu.
12. Dodanie kategorii do budżetu.
13. Zdefiniowanie kwot maksymalnych dla poszczególnych kategorii w budżetach.
14. Dodanie wydatku do budżetu.
15. Możliwość analizy wydatków na interaktywnych wykresach.
16. Obsługa błędów i stosowne informacje dla użytkownika w przypadku błędnie wypełnionych pól formularza dodawania przychodu, wydatku, budżetu.

### Wymagania niefunkcjonalne

1. Bezpieczne logowanie
2. Uruchomienie i hostowanie aplikacji z wykorzystaniem technik wirtualizacji i kontenerów Docker
3. Nierelacyjna baza danych
4. RWD – responsywny design, dopasowywujący się do urządzenia które jest używane do wyświetlenia aplikacji
5. Prawidłowa dla urządzeń mobilnych wielkość przycisków i elementów interakcji
6. Prawidłowa dla urządzeń mobilnych wydajność pod kątem wolniejszych łącz internetowych i optymalizacja pod kątem procesorów mobilnych.
7. Prawidłowa, łatwo rozszerzalna, podzielona na moduły aplikacja, zarówno w części klienckiej jak i serwerowej.
8. Aplikacja typu SPA – Single Pager Application – umożliwia pracę bez przeładowania przeglądarki.
9. Aplikacja oparta o nowoczesny framework Angular.
10. Dowolny nowoczesny telefon i komputer z przeglądarką internetową będzie miał dostęp do aplikacji.

### Diagram przypadków użycia



## Architektura systemu

### Ogólne

Aplikacja będzie aplikacją przeglądarkową podzieloną na warstwę kliencką – odpowiadającą za interfejs użytkownika, prawidłowe wyświetlanie go, i interakcję z warstwą serwerową poprzez zapytania RESTowe. Warstwę serwerową – odpowiadającą za prawidłowe obsługiwanie zapytań przychodzących z frontendu, odpowiednie połączenie z bazą danych MongoDB i odbieranie, wprowadzanie, edytowanie danychw w niej zawartych.

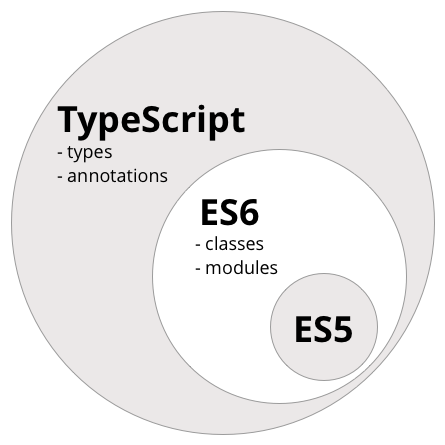
Środowisko developerskie tak jak i środowisko produkcyjne, będzie w prosty sposób (dla użytkownika chcącego uruchomić je) stworzone z wykorzystaniem narzędzi wirtualizacji i kontenerów Docker. Dla każdego z „serwisu” (frontend, backend, baza danych) będzie stworzony oddzielny kontener. Kontenery te będą ze sobą połączone specjalnymi mechanizmami Dockera (plik docker-compose.yml). Dzięki takiemu zaprojektowaniu, każdy nowy developer, czy też osoba która chciałaby obejrzeć oprogramowanie działające na lokalnej maszynie posiada prosty sposób na uruchomienie go. Wystarczy, że będzie miała zainstalowany **Docker Community Edition** i jedyne co musi zrobić, to w katalogu głównym aplikacji wykonać polecenie **docker-compose build** a później **docker-compose up** by bezproblemowo uruchomić każdą z warstw połączonych ze sobą w prawidłowy sposób.

### Warstwa kliencka

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 Czym jest TypeScript

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



Rysunek 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.

### Warstwa serwerowa

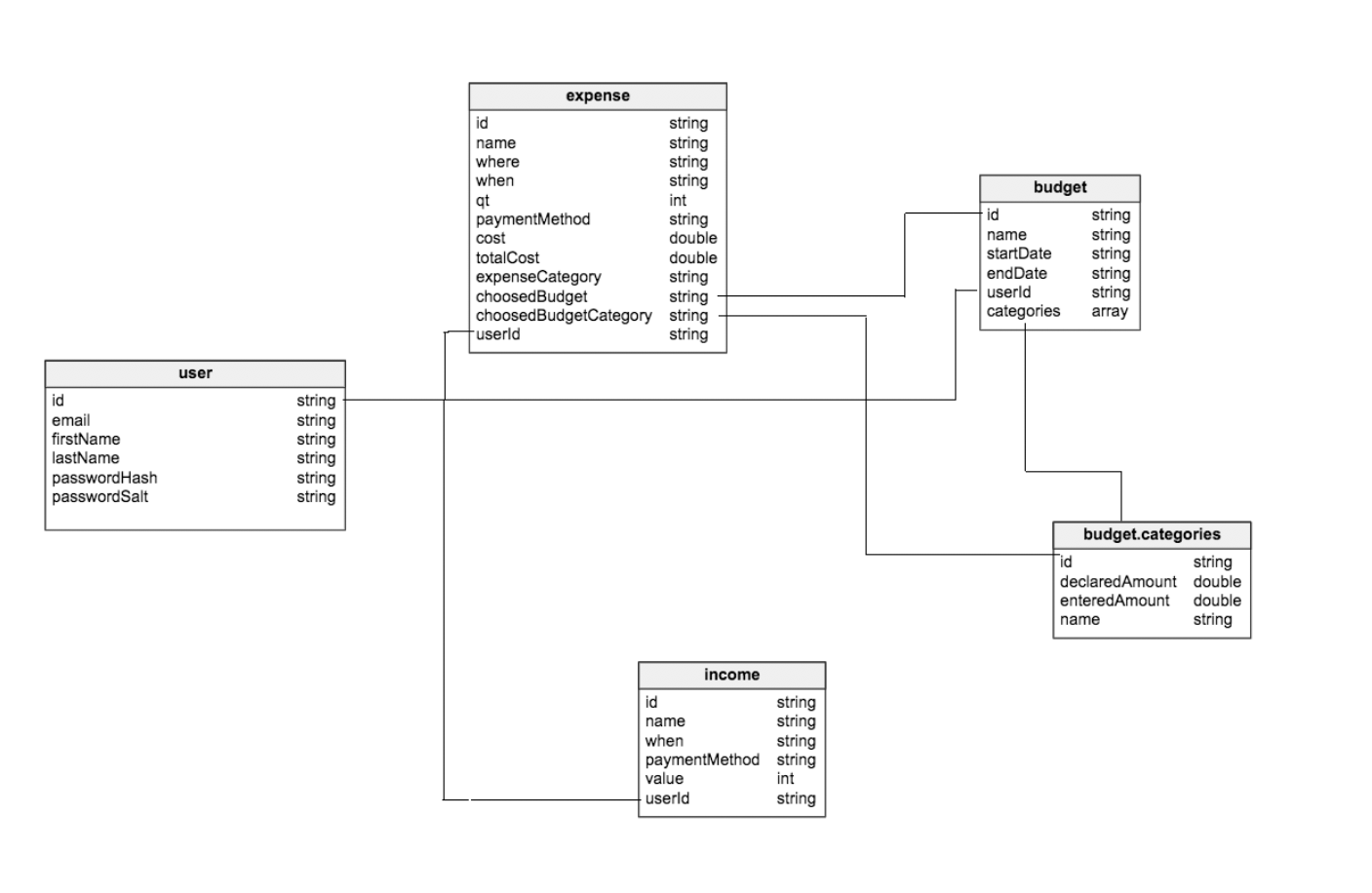
W mojej pracy inżynierskiej oprócz części klienckiej stworzyłem także część odpowiadającą za wszelkie akcje wykonujące się po stronie serwera. W tym miejscu pozwoliłem sobie na mały eksperyment i połączyłem kilka elementów by stworzyć dobrze funkcjonujący i skalowalny backend.

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.**

### 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.

Z powodów nierelacyjności bazy danych, poniżej przedstawiam pseudo model bazy danych, przedstawiający istniejące i pseudo powiązania pomiędzy nimi.



Jako komentarz do tego modelu warto nadmienić, że nie przetrzymuję oczywiście haseł użytkownika w bazie danych a jedynie sól i zahashowane posolone hasło potrzebne do późniejszej walidacji.

### JavaScript

### Angular

### HTML5 i CSS3

### NestJS

### ExpressJS

### TypeScript

### MongoDB

### Mongoose

### ChartJS

## Narzędzia programistyczne

### Visual Studio Code

### Postman

### Git

## Metody i narzędzia realizacji

## Koncepcja przechowywania danych

## Projekt interfejsu użytkownika

# 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.

# Testy i weryfikacja systemu

# Przykładowy scenariusz wykorzystania systemu

# Zakończenie

# 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.

# Spis rysunków

[Rysunek 1 Zobrazowanie problemu budżetu osobistego 6](#_Toc512375748)

[Rysunek 2 Aplikacja Cents 7](#_Toc512375749)

[Rysunek 3 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. 8](#_Toc512375750)

[Rysunek 4 Aplikacja Cents nie wzbudza zaufania. (a aplikacja zarządzająca naszymi pieniędzmi powinna) 9](#_Toc512375751)

[Rysunek 5 – wysuwane boczne menu 11](#_Toc512375752)

[Rysunek 6 – zasady responsywności – dostępność na każdym urządzaniu 12](#_Toc512375753)

[Rysunek 7 – schemat dodawania wydatku/przychodu/budżetu 13](#_Toc512375754)

[Rysunek 8 – podstrona budżetu osobistego z interaktywnymi wykresami 14](#_Toc512375755)

[Rysunek 9 Czym jest TypeScript 18](#_Toc512375756)

[Rysunek 10 Architektura aplikacji Angular 18](#_Toc512375757)

# Spis tabel

[Tabela 1 Przykładowa tabela 14](#_Toc134224217)