

UNIwersytet Gdański
Wydział Matematyki, Fizyki i Informatyki

Mateusz Kwiatkowski

nr albumu: 194 925

Walidacja w elektronicznym systemie zarządzania osiągnięciami studenta

Praca magisterska na kierunku:

INFORMATYKA

Promotor:

dr Włodzimierz Bzyl

Gdańsk 2015

Streszczenie

sa powinno zawierać omówienie głównych tez pracy magisterskiej, celów jakie autor sobie postawił

sb powinno zawierać informację czy udało się je zrealizować

sc należy także napisać jakimi metodami, technologiami się posłużono i jakie to przyniosło efekty

Pracę poświęcono zagadnieniu walidacji, kwestii ważnej i integralnie związanej z odpowiednim funkcjonowaniem sieci. W szczególności zwrócono uwagę na aspekt prawidłowego zarządzania jej jakością, co obligatoryjnie wiąże się z problemem odpowiedniego zabezpieczenia i odpowiedzialnego korzystania z niej.

Praca prezentuje sposób tworzenia oraz funkcjonowanie pakietu walidującego do frameworka *Meteor*. Pakiet ten ma umożliwić użytkownikowi wprowadzenie błędnych danych do systemu, dzięki czemu podniesiony zostanie poziom zaufania do korzystania z niego. Jednocześnie, aby przybliżyć i zademonstrować jego działanie, utworzono na potrzeby pracy, aplikację elektroniczny indeks. Wybór dokumentu nie był przypadkowy. Świadczy o tym jego wysoka ranga wśród uczelnianej dokumentacji urzędowej. Inny, równie istotny powód wyboru stanowi fakt, iż korzystanie z sieci komputerowej w systemie edukacyjnym stało się powszechne. Uczelnie wyższe wykorzystują sieć, by między innymi ułatwić kontakty na linii: wykładowca - student - administracja uczelni. Temu ma służyć wprowadzenie w ostatnich latach przez większość uczelni wyższych w Polsce, w tym Uniwersytet Gdański, elektronicznego systemu zarządzania osiągnięciami studentów tzw. elektronicznego indeksu.

W pracy, walidacji zostaną poddane operacje, które można wykonać w aplikacji elektroniczny indeks. Do stworzenia jej użyto frameworku *Meteor*, a dane wprowadzone do systemu, w celu ich przechowywania umieszczono w bazie danych - *MongoDB*.

Założono, że skutkiem tego działania będzie uproszczenie, a nawet intuicyjność obsługi oprogramowania. Cele założone przez autora pracy zostały zrealizowane, czego dowodem jest udostępnienie do pobrania pakietu walidacji systemu.

Spis treści

Wprowadzenie	5
1. Walidacja oprogramowania	9
1.1. Specjalistyczny wstęp do walidacji	12
1.2. Kategorie walidacji	14
1.3. Etapy walidacji	15
2. Elektroniczny indeks	18
2.1. Filtrowanie danych	18
2.2. Zarządzanie elektronicznym indeksem przez administratora	19
2.3. Funkcjonalność dla prowadzącego zajęcia	22
2.4. Funkcjonalność dla studenta	23
2.5. Funkcjonalność dla pracownika dziekanatu	24
3. Aplikacja Elektroniczny indeks w Meteor	25
3.1. Opis wybranych technologii użytych w pracy	25
3.1.1. Javascript	25
3.1.2. Meteor	25
3.1.3. MongoDB	26
3.1.4. Velocity	26
3.1.5. Distributed Data Protocol	26
3.1.6. Roles	26
3.1.7. Iron router	27
3.1.8. Accounts	27
3.1.9. Underscore string	27
3.2. Opis tworzenia aplikacji	27
3.3. Opis testowania aplikacji	32
4. Pakiet walidujący operacje elektronicznego indeksu	35
4.1. Funkcjonalność pakietu	35

4.2. Opis tworzenia pakietu	35
4.3. Implementacja pakietu w aplikacji	35
4.4. Przetestowanie pakietu	35
Zakończenie	36
A. Mesosphere	37
B. TinyTest	38
Bibliografia	39
Spis tabel	40
Spis rysunków	41
Oświadczenie	42

Wprowadzenie

- wa** jak nazwa wskazuje, ma wprowadzać w obszar problemowy pracy
- wb** powinno przedstawiać ogólne uwarunkowania problemu oraz opisać go w kontekście
- wc** powinno zawierać powód dlaczego poruszyło się taki temat
- wd** należy odnieść się do dorobku innych

Najcenniejszym walorem komputera i Internetu są przechowywane w nich dane - zarówno ich ilość, jak i jakość. Ze względu na to, z dnia na dzień, rośnie liczba użytkowników sieci. Jednocześnie zwiększa się liczebność i różnorodność usług sieciowych.

Komputer i Internet zmienił, wciąż zmienia naszą codzienność. To prawda oczywista. Usługi internetowe nie są już domeną urzędów, firm czy handlu. Chcemy za ich pomocą robić zakupy, obsługiwać konto w banku, a także załatwiać wszelkie formalności w urzędach. Jest to po prostu wymóg rozwoju cywilizacji, techniki oraz oszczędności czasu.

Coraz częściej systemy informatyczne wykorzystywane są w edukacji społeczeństwa. Jeszcze do niedawna na wszystkich uczelniach wyższych stosowano klasyczne indeksy papierowe, aby zarchiwizować osiągnięcia studentów podczas całego cyklu kształcenia. Jednak w wyniku rozwoju technologii internetowych coraz częściej rezygnuje się z klasycznych rozwiązań, zastępując je ich elektronicznymi odpowiednikami.

Dziś wiele szkół i uczelni wprowadziło do obszaru swego funkcjonowania nowoczesny system ewidencji osiągnięć ucznia czy studenta. W szkołach podstawowych, gimnazjach, liceach, technikach czy zasadniczych szkołach zawodowych jest nim tzw. dziennik elektroniczny. W uczelniach wyższych nazwano go elektronicznym indeksem. Zjawisko to stanowi nie lada wyzwanie, ponieważ wiąże się z problemem niezawodnego świadczenia usług w sieci komputerowej. Odbiorca, w tym przypadku uczeń lub student, musi mieć pewność, że dane są stałe, prawdziwe, od-

powiednio zabezpieczone przed ich utratą czy nieuprawnionym dostępem. Należy nadmienić, że taki poziom zaufania i poczucia bezpieczeństwa funkcjonowania systemu, powinna mieć również druga strona - nadawca, ten który wprowadza owe dane. Jest to tym bardziej ważną kwestią, gdyż coraz częściej mamy do czynienia ze zdarzeniami, wskazującymi na nieprawidłowe stosowanie sieci komputerowej lub jej nadużycie.

Rozwiązaniem, które zapewniłoby wzrost poziomu zaufania do korzystania z sieci, w tym również z elektronicznego systemu zarządzania osiągnięciami ucznia lub studenta jest, według autora niniejszej pracy, odpowiednie i odpowiedzialne zarządzanie jej jakością, czemu służy walidacja systemu.

Zjawisko to jest szeroko stosowane w technice i informatyce. Internetowy Słownik Języka Polskiego wyjaśnia hasło „walidacja” w następujący sposób: „walidacja (technika) - badanie odpowiedności, trafność lub dokładności czegoś”.[1]

Sam termin - „walidacja” pochodzi od angielskiego słowa „validate” i oznacza - w kontekście informatycznym - sprawdzanie poprawności i zgodności z zadanymi kryteriami. Jest on stosowany w odniesieniu do danych pochodzących od użytkownika, jak również w stosunku do zmiennych, obiektów, typów i klas w różnych językach programowania.[2]

Walidacja jest działaniem, mającym na celu potwierdzenie w sposób udokumentowany i zgodny z założeniami, że procedury, procesy, urządzenia, materiały, czynności i systemy, rzeczywiście prowadzą do zaplanowanych wyników. Znana jest także jako kontrola jakości oprogramowania.[3]

Wprowadzając dane do systemu, użytkownik może - świadomie lub nie - popełnić pomyłkę. Jeżeli dane odebrane przez użytkownika poddamy przetworzeniu bez weryfikacji, wówczas, w zależności od odporności aplikacji, możemy mieć do czynienia z różnymi rodzajami błędów, od drukowania w przeglądarce klienta komunikatów diagnostycznych, poprzez utratę spójności bazy danych, aż po ujawnienie niepowołanym użytkownikom informacji poufnych. Z tego powodu nie wolno ignorować wagi problemu.

Aplikacje pozbawione walidacji pozwalają użytkownikowi na wprowadzenie irracjonalnych danych do systemu. Przykładem takiej aplikacji jest wspomniany przez autora pracy, elektroniczny indeks. Operacje, takie jak: wystawianie studentowi ocen z ćwiczeń czy też oceny z egzaminu kończącej edukację z danego przed-

miotu, powinny być odpowiednio walidowane. Dzięki temu nie dojdzie do niepożądanych zjawisk typu:

- student nie uzyskał pozytywnej oceny z ćwiczeń, a otrzymuje ocenę z egzaminu kończącego przedmiot,
- student otrzymuje ocenę spoza skali oceniania systemu danej uczelni,
- student uzyskuje ocenę od osoby nieuprawnionej do jej wystawienia.

Dlatego też, autor pracy chce zwrócić uwagę na rodzący się problem, związany z wprowadzeniem przez uczelnie elektronicznego indeksu oraz jego odpowiednim funkcjonowaniem. Zaproponowanie zastosowania walidacji w elektronicznym systemie wystawiania ocen usprawni działanie oraz udoskonali jego funkcjonalność. Korzystając z aplikacji, w której zaimplementowana jest walidacja możemy mieć pewność, że nie dojdzie do sytuacji, by użytkownik wprowadził błędne dane do systemu. Należy również zwrócić uwagę na ekonomiczny aspekt walidacji. Mianowicie oszczędność czasu użytkownika czy zwiększenie efektywności jego pracy.

W celu ukazania i udowodnienia przydatności walidacji podczas korzystania z elektronicznego systemu zarządzania osiągnięciami studenta, pokazano w pracy działanie tego zjawiska w aplikacji stworzonej w frameworku *Meteor* oraz zaprezentowano ułożony pakiet oraz wyjaśniono, jak udostępnić go w prosty, jasny i zrozumiały sposób.

Tworzenie pakietu walidującego oraz aplikacji - elektroniczny indeks, która korzysta ze stworzonego w ramach pracy pakietu, oparto na doświadczeniu innych badaczy, zajmujących się oprogramowaniami komputerowymi, takich jak: Kelly Copley, Tom Coleman czy Sacha Greif. W pracy umieszczono ponadto uzasadnienie, dlaczego wybrane technologie, takie jak - *Meteor* oraz *MongoDB*, to najbardziej trafny wybór do generowania pakietu walidacyjnego elektronicznego zarządzania osiągnięciami studenta.

Autor niniejszej pracy miał kontakt z wieloma systemami zarządzania osiągnięciami studentów, ale w każdym można było doprowadzać do anomalii. Zajęcie się rozwiązaniem tego problemu jest, z punktu widzenia informatyka interesujące.

Efektem pracy może być nie tylko usprawnienie działania systemu, ale również poczucie, że praca z nim jest prosta, przyjemna i wręcz intuicyjna.

Walidacja oprogramowania

stressp Najbardziej istotna informacja

topicp Początek nowego zdania odnoszący się do istotnej informacji poprzedniego zdania

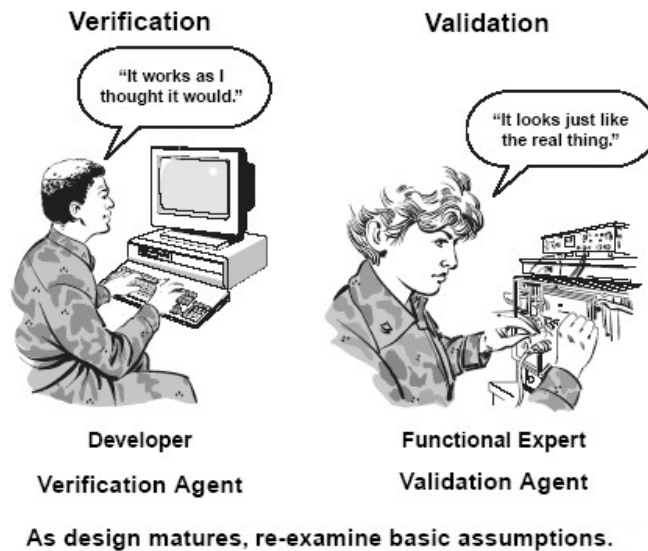
W testowaniu oprogramowania ważne są pojęcia – weryfikacja i walidacja. Pojęcie walidacji wyjaśniono we wprowadzeniu oraz rudymencie podrozdziału pierwszego. Należy wyjaśnić również pojęcie *weryfikacji*

Internetowy Słownik Języka Polskiego podaje następujące znaczenia tego słowa: *weryfikacja* z łac. *verifico* - „wycenić wartość, ustalić; 1. Sprawdzenie zgodności czegoś z prawdą; sprawdzenie autentyczności czegoś; weryfikacja hipotezy, teorii, faktów. 2. Sprawdzanie i ocena czyichś kwalifikacji - np. pracownika”. Różnica w znaczeniu tych pojęć jest istotna. Aby ją wskazać należy zwrócić uwagę na celowość tych dwóch określeń. Specyfika działania weryfikacji to zastosowanie pytania: „Czy produkt tworzony jest prawidłowo?”. Z kolei dla procesu walidacji typowym pytaniem będzie: „Czy tworzony produkt jest prawidłowy?”. *Prawidłowy* – to znaczy zgodny z wytycznymi programowania, przy zastosowaniu odpowiednich metod, języka programowania i algorytmów.

Pojęcia weryfikacji i walidacji są znaczeniowo na tyle bliskie, że mogą przysporzyć trudności. Zarówno weryfikacja jak i walidacja produktu są czynnościami, które służą sprawdzeniu, czy wytworzony produkt jest taki, jaki sobie życzyliśmy my, bądź inny interesariusz. *Produkt* możemy rozumieć jako:

- dany moduł systemu,
- blok kodu,
- cały system,
- istotny dokument, na bazie którego moduł lub system będzie zbudowany.

Warto zapamiętać, że obie te czynności zachodzą w wielu różnych momentach i mogą pojawić się w wielu fazach procesu tworzenia systemu. **Błędnym będzie zatem rozumienie, np. walidacji, jako jakiegoś etapu wieńczącego budowę systemu.**



Rysunek 1.1. Różnice między weryfikacją, walidacją

Źródło: <http://commons.wikimedia.org/wiki/>

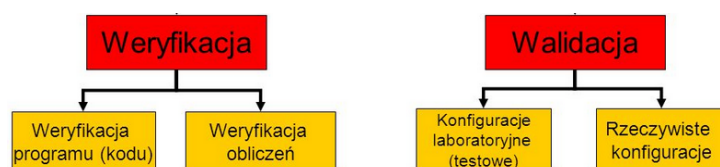
Różnice między weryfikacją i walidacją dotyczą przede wszystkim tego, z jakiej perspektywy dokonujemy sprawdzenia **czy jest to perspektywa technologiczna i typowa dla zespołów budujących system, czy raczej z perspektywy użytkownika końcowego, który nie ma obowiązku rozumieć technicznej strony systemu, z którego będzie korzystał.**

Najprościej jest zapamiętać, że weryfikujemy to, co da się wyliczyć, wykazać logicznie i nie ma jak zanegować tego, co już zweryfikowane. Przy dobrze spisanych wymaganiach weryfikacji dokonać może każdy człowiek rozumiejący tekst specyfikacji i wiedzący, jak wykonać test. Natomiast walidacja zawsze jest po stronie odbiorcy i to zaspokojenie jego potrzeb jest ostatecznym kryterium sukcesu. Odcinając się od możliwości konsultowania z klientem spraw dotyczących choćby

wygody użytkownika produktu, wytwórcy ryzykują niepowodzenie walidacji.

Weryfikacja produktu procesu polega na dostarczeniu dowodów, że dany produkt spełnia z d e f i n i o w a n e wymagania. Można spotkać się też z objaśnieniami tego terminu mówiącymi, że jest to sprawdzanie „czy aplikacja jest prawidłowo zbudowana”, czy produkty danego etapu produkcji spełniają wymogi założone na początku całego procesu. Kluczowym słowem jest wyraz - *zdefiniowane*. Weryfikacja domaga się odniesienia do przesłanek, które spisano w taki sposób, by można je było sprawdzić w sposób jednoznaczny. Weryfikacja polega na stwierdzeniu, że dany punkt np. specyfikacji technicznej jest informacją, która w sposób prawdziwy opisuje dany produkt poddany testowi. W praktyce powinniśmy zatem unifikować weryfikację z procesem, który zakończy się „zero-jedynkowym” rozstrzygnięciem, decydującym o tym, że dany produkt wygląda, bądź zachowuje się, bądź jest taki, jak ustalono dla niego w specyfikacji wszystkich przesłanek. Jeśli na przykład jakąś funkcjonalność można:

- sprawdzić poprzez odczytanie wyniku liczbowego, którego ona dostarcza;
- zbadać czy odpowiedź systemu nastąpi w konkretnie ustalonym czasie;
- sprawdzić czy następuje przesłanie odpowiedniego pliku między serwerem a klientem, wtedy możemy oczekiwać, że proces sprawdzania tego produktu jest weryfikacją.



Rysunek 1.2. Weryfikacja i walidacja

Źródło: <http://slideplayer.pl/slide/427709/> slajd 4

Walidacja produktu polega na sprawdzeniu poprawności i dostarczeniu dowodów, że proces wytwarzania go, spełnia potrzeby i wymagania użytkownika. Tłumaczy się ją również, jako „sprawdzenie czy aplikacja jest poprawna” / w starszych

wersjach słowników testerskich / [4]

Najistotniejsze do zrozumienia walidacji jest to, że to, co było jasno zdefiniowanym kryterium zaliczenia testu, jest tutaj zamienione na potrzeby i wymagania użytkownika. O ile zweryfikować można różne rzeczy bez wczuwania się w rolę użytkownika końcowego / bo wystarczy się oprzeć choćby na wyliczeniach i sprawdaniu, czy wyniki są zgodne z założeniami ze specyfikacji /, to walidacja służy właśnie do sprawdzenia wszystkich tych funkcjonalności, które użytkownik będzie oceniał pod kątem swoich osobistych upodobań, często subiektywnie, nie raz w oparciu o rozmaite nawyki. Sukces walidacji opiera się niejednokrotnie na sprawach, które ciężko jest przewidzieć podczas pisania specyfikacji funkcjonalnej. Odbiorca systemu nieraz nie uświadamia sobie swoich własnych aberracji i może stwierdzić ich brak dopiero wtedy, gdy przystępuje do finałowego testu akceptacyjnego. Często wtedy słyszonym argumentem z ust niezadowolonego odbiorcy jest to, że coś było „tak oczywiste, że nie trzeba tego było spisywać w specyfikacji”. O ile kluczem do udanej weryfikacji produktu jest klarowne zdefiniowanie kryteriów zaliczenia testu, o tyle kluczem do udanej walidacji jest pogłębiona i sprawna komunikacja z odbiorcą, zadawanie celnych pytań pomagających w zrozumieniu autentycznych potrzeb klienta, przeprowadzanie wraz z klientem symulacji zastosowań systemu na wszelkiego rodzaju makietach. Kolejną bardzo istotną sprawą jest umiejętne korzystanie z badań aktualnych rozwiązań i tendencji w produktach dostępnych na rynku, co wiąże się z przeglądaniem odpowiednich stron internetowych, sięganiem po prace znawców tematu i autorytetów, ciągłym dokształcaniem.

Walidację warto zapamiętać jako ogół tych czynności, które zwiększają szansę na zadowolenie odbiorcy tworzonego produktu i które będą brały pod uwagę konsultacje z tym odbiorcą, wspólne wypracowywanie rozwiązania i przewidywanie jego preferencji. [4]

1.1. Specjalistyczny wstęp do walidacji

Walidacja jest to proces wyznaczania kompatybilności stopnia użytkowania systemu, w którym dany model staje się wiernym odzwierciedleniem rzeczywistego systemu. Chodzi o skuteczną zgodność wprowadzonych danych z ich oryginalnymi.

nałem. Proces ten ma na celu zilustrowanie czy symulacja dostarcza użytkownikowi wiarygodnych danych wyjściowych, zgodnych z danymi wejściowymi użytkownika. Dokonuje więc, weryfikacji zgodności wizji projektanta z realnym światem. Pełni rolę autocenzury konkretnego systemu, by każdy wirtualny użytkownik, miał pewność, że dane wprowadzone do systemu są stałe i spełniają przydzieloną im rolę.

Każdy system informatyczny wymaga osiągnięcia odpowiedniego stopnia adekwatności, bezbłędności, stabilności oraz wyeliminowania błędów działania w modelu. Przez model, który przeszedł walidację, rozumieć należy ten, który został poddany serii operacji, mających na celu doprecyzowanie go do optymalnego poziomu, przez co, zgodnie z jego przeznaczeniem, będzie mógł sprostać postawionym przed nim zadaniom. Taki poziom wiarygodności modelu uzyskamy dzięki procesowi walidacji.

Systemy zautomatyzowane i skomputeryzowane stosowane szczególnie w przemyśle wysokich technologii, muszą być poddawane okresowym kontrolom, potwierdzającym ich jakość w celu wykrycia ewentualnych, potencjalnych zagrożeń, wynikających z bezpośredniego lub pośredniego wpływu na produkt końcowy. Walidacja zatem ma udokumentować, w jaki sposób należy zmienić i udoskonalić proces, aby zminimalizować ewentualne skutki jego nieprawidłowego działania.

Jeżeli istnieje możliwość, walidacja systemu powinna być poprzedzona rozmową z jego użytkownikiem. Ma ona spełnić rolę sondy i zebrania cennych informacji, by program walidacyjny stał się optymalny. Mając przez długi czas styczność z systemem rzeczywistym, ekspert często potrafi odpowiedzieć na pytania dotyczące całości systemu lub wskazać rażące błędy, których podstawą jest niezrozumienie rzeczywistego systemu, co staje się przyczyną generowania przez symulację złych wyników. [5]

Innym zagadnieniem mającym związek z procesem walidacji jest problem czasu tworzenia go oraz związanych z tym kosztów. Walidacja procesów, systemów czy urządzeń jest czasochłonna. Tworzenie dokumentacji, procedur, wykonywanie testów i działań naprawczych na ogół przeciąga się w czasie. Dobrze zarządzana walidacja, obciąża budżet projektu w skali 4 - 7 % kosztów. Nieprawidłowo prowadzona podnosi ją do 20 – 30% kosztów całkowitych. Poniesienie niskich nakładów może przynieść wymierne zyski ekonomiczne już w krótkiej perspektywie. Według

danych zgromadzonych przez osoby zajmujące się walidacją dla nowych urządzeń i systemów produkcyjnych, wydajność początkowa urządzenia, które były przedmiotem pełnego cyklu walidacji, może być 2 – 3 razy wyższa, niż tych uruchomionych bez jej przeprowadzania. Mechanizmy i systemy poddane walidacji osiągają pełną zdolność produkcyjną a ich kultura obsługi i serwisu jest wysoka, ponieważ w czasie testów walidacyjnych pracownicy zdobywają praktyczną wiedzę od dostawcy czy użytkownika. Właściwe opracowanie *specyfikacji wymagań użytkownika*

/ w skrócie URS / ¹, prowadzenie kwalifikacji projektu, nadzorowanie i współpraca z dostawcą od początku wiąże się z nakładami. Jednak poniesione koszty są niewspółmierne niższe od kosztów poprawy pracy czy usuwania usterek w gotowym urządzeniu, czy utworzonym systemie. Dlatego ważna jest współpraca z dostawcą / użytkownikiem od samego początku inwestycji. Wspólne rozwiązywanie problemów, tworzenie scenariuszy testowych, pozwala na obustronną wymianę wiedzy i znaczne ograniczenie kosztów w późniejszej eksploatacji systemu i urządzenia.[6]

1.2. Kategorie walidacji

Kryterium funkcjonalności wyróżnia następujący podział walidacji:

1. **Walidacja prospektywna** – zadaniem takiej walidacji jest, aby przed wprowadzeniem nowych produktów na rynek upewnić się, że funkcjonują prawidłowo i spełniają standardy bezpieczeństwa.
2. **Walidacja retrospektywna** – ten proces wyasygnowano dla produktów, które są już w użyciu oraz dystrybucji czy produkcji. Walidację tą przeprowadza się na podstawie wcześniej określonych oczekiwań specyfikacji produktu oraz danych historycznych. Jeśli jakiegokolwiek dane krytyczne są niepełne, to nie mogą być one przetworzone lub mogą być przetworzone częściowo. Zadania są uważane za konieczne, gdy:

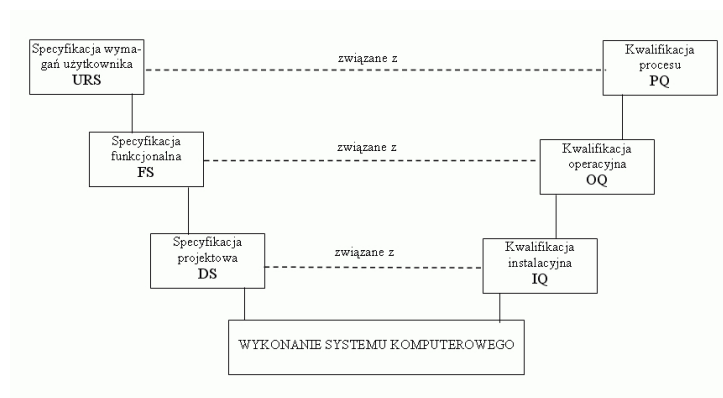
- walidacja prospektywna jest niewystarczająca lub błędna,

¹ang. User Requirements Specification - URS

- zmiana przepisów prawnych lub norm wpływa na zgodność produktów wypuszczonych na rynek,
 - przywrócenie produktu do użytkowania.
3. **Rewalidacja** – przeprowadza się dla produktu, który został odrzucony, naprawiony, zintegrowany, przeniesiony lub po upływie określonego czasu.[7]

1.3. Etapy walidacji

Zakres walidacji powinien uwzględniać wiele czynników systemu, w tym jego zamierzone zastosowanie i rodzaj walidacji oraz czy mają być dołączane nowe elementy systemu. Walidacja powinna być uznawana za część całego cyklu użytkowania systemu komputerowego. Obejmuje on planowanie, specyfikację, programowanie, badanie, odbiór techniczny, dokumentację, działanie, monitorowanie i modyfikowanie.



Rysunek 1.3. Cykl życia systemu

Źródło: <http://www.label.pl>

Projektowanie systemu skomputeryzowanego można podzielić na kilka etapów:

1. **Specyfikowanie** – URS oraz specyfikacja funkcjonalna². W planie walida-

²ang. Functional Specification - FS

cji tego etapu projektowania powinno się uwzględnić audyt dostawcy oraz przegląd specyfikacji.

2. **Projektowanie** – projekt hardware³, software⁴, projekt mechaniczny i elektryczny oraz projekt sieci informatycznej. Walidacją objęty jest przegląd poszczególnych projektów – kwalifikacja projektu⁵.
3. **Wykonanie** – utworzenie hardware, połączeń elektrycznych modułów systemu, oprogramowanie modułów, montaż całego urządzenia, wykonanie sieci informatycznej. Walidacja obejmuje przeglądy wykonania poszczególnych czynności oraz przegląd kodów źródłowych oprogramowania.
4. **Testowanie** – testowanie hardware, poszczególnych modułów oprogramowania, integracji oprogramowania oraz testy funkcjonalne kompletnego urządzenia. Walidacja obejmuje nadzór nad dostawcą poszczególnych elementów systemu.
5. **Instalacja** – instalacja hardware, software, urządzeń, sieci informatycznej, testy instalacyjne hardware oraz sieci informatycznej. Walidacja tego etapu projektowania systemu skomputeryzowanego dotyczy pełnej kwalifikacji instalacyjnej⁶.
6. **Odbiór** – testy akceptacji systemu, w tym kompletności dokumentacji. Na tym etapie realizacji, walidacja dotyczy kwalifikacji operacyjnej⁷ oraz procesowej⁸. Zakończenie jej uwieńczone jest raportem, który powinien określać przykładowo urządzenia produkcyjne, krytyczne parametry procesu i krytyczne zakresy operacyjne, charakterystykę produktu, sposób pobierania próbek koniecznych do zebrania danych z badań, ilość przebiegów procesu walidacyjnego i akceptowalne wyniki badań.

³ang. Hardware Design System - HDS

⁴ang. Software Design System - SDS

⁵ang. Design Qualification - DQ

⁶ang. Installation Qualification - IQ

⁷ang. Operational Qualification - OQ

⁸ang. Performance Qualification - PQ

7. **Użytkowanie systemu** – konserwacja i utrzymanie sprawności systemu, nadzór nad zmianami. W okresie użytkowania systemu przeprowadzane są okresowe, planowane rewalidacje, a system jest monitorowany.[8]

ROZDZIAŁ 2

Elektroniczny indeks

Elektroniczny indeks jest to elektroniczna platforma, która służy do wystawiania ocen studentom przez prowadzących zajęcia w czasie rzeczywistym, a także do wglądu do tych ocen przez studentów. Dzięki wdrożeniu do użytku wspomnianej platformy, zarówno studenci jak i prowadzący, mogą zaoszczędzić mnóstwo czasu oraz nerwów związanych z próbą zdobycia wpisu, odtworzenia indeksu, gdy zostanie on zgubiony lub uniknięcia długiego stania w kolejce do dziekanatu. Dodatkowo znacznie ułatwiona zostanie biurokracja na linii dziekanat – wykładowca, a samo zaliczenie semestru studentowi przez osoby do tego uprawnione staje się dużo prostsze i szybsze.

2.1. Filtrowanie danych

Filtrowanie danych w elektronicznym indeksie jest jego ważną częścią. Dzięki temu można uniknąć sytuacji, w której nieautoryzowany użytkownik uzyska dostęp do danych, do których nie powinien mieć dostępu. *Meteor* nie ma domyślnie zaimplementowanego pakietu, który pozwoliłby na sprawne zarządzanie dostępnością treści w aplikacji. Dzięki rosnącej popularności frameworku *Meteor* oraz coraz szerszej grupie deweloperów z każdym dniem pojawiają się nowe pakiety, które ułatwiają i zwiększają funkcjonalność tworzonych aplikacji. Do efektywnego przekazywania danych poszczególnym użytkownikom wykorzystano pakiet *meteor-roles*, który pozwala zarządzać rozsyłanymi danymi oraz jest kompatybilny z wbudowanym pakietem do zarządzania kontami. Instalacja pakietu do zarządzania rolami wymaga dołączenia do projektu pakietu *accounts-password*, co można zrobić następującym poleceniem w konsoli:

Listing 2.1. Instalacja accounts-password

```
meteor add accounts-password
```

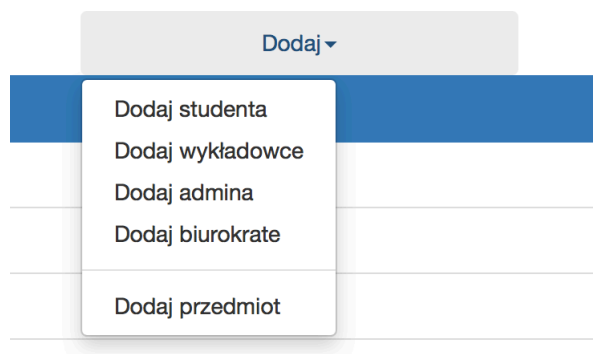
Następnie możemy dodać właściwy pakiet

Listing 2.2. Instalacja pakietu roles

```
meteor add alanning:roles
```

2.2. Zarządzanie elektronicznym indeksem przez administratora

Administrator jest osobą, która nad wszystkim czuwa, dlatego musi mieć dostęp do wszystkich danych i wszystkich funkcjonalności programu, aby móc kontrolować jego poprawność działania. Będąc głównym zarządcą platformy, administrator jako jedyny ma prawo dodawać do systemu nowe dane. Dzięki temu, że najważniejsze dane wyświetlają się administratorowi od razu po zalogowaniu do systemu, jest on w stanie szybko wykonać powierzone mu zadanie.



Rysunek 2.1. Menu dodawania

Źródło: Własne

Jako główny zarządca, administrator ma za zadanie przypisywać studentów do przedmiotów, na które są zobowiązani uczęszczać, a także do przedmiotów, które sami wybiorą w trakcie toku nauczania. Student, który został zapisany na dany przedmiot, nie pojawi się ponownie na liście, dzięki czemu nie ma możliwości zapisania danego użytkownika dwa razy na te same zajęcia.

Zapisz na przedmiot

Indeks	Imię	Nazwisko	
186410	Szymon	Domurat	
195001	Adrian	Elszkowski	
194983	Radosław	Głabiowski	
194913	Bartosz	Gniado	

Rysunek 2.2. Fragment listy studentów do zapisu na zajęcia

Źródło: Własne

Kolejnym zadaniem jest dodawanie przedmiotów, które odbywają się na uczelni. Administrator podaje jego nazwę, wybiera semestr, na którym dane zajęcia się odbywają i z rozwijanej listy wybiera prowadzącego dane wykłady czy ćwiczenia.

Przedmiot

Prowadzący

dr Włodzimierz Bzyl

Semestr

1

Rysunek 2.3. Formularz dodawania przedmiotu

Źródło: Własne

Następną równie ważną funkcją jest dodawanie studentów, prowadzących zajęcia, adminów oraz pracowników dziekanatu. Z menu dodawania admin wybiera, do jakiej grupy chce dodać użytkownika.

1. Dodawanie studenta



Formularz dodawania studenta. Zawiera pola tekstowe dla numeru albumu (194925), imienia (Mateusz), nazwiska (Kwiatkowski) oraz hasła (maskowane gwiazdkami). Na dole znajduje się przycisk 'Dodaj studenta'.

Rysunek 2.4. Formularz dodawania studenta

Źródło: Własne

2. Dodawanie wykładowcy



Formularz dodawania wykładowcy. Zawiera pole rozwinięte dla tytułu naukowego (dr), pola tekstowe dla imienia (Włodzimierz), nazwiska (Bzyl) oraz hasła (maskowane gwiazdkami). Na dole znajduje się przycisk 'Dodaj wykładowcę'.

Rysunek 2.5. Formularz dodawania wykładowcy

Źródło: Własne

3. dodawanie biurokraty



Formularz dodawania pracownika dziekanatu. Zawiera trzy pola tekstowe: 'Imię' z wartością 'Iwona', 'Nazwisko' z wartością 'Szreder' oraz 'Hasło' z maskowanymi znakami '*****'. Poniżej znajduje się przycisk 'Dodaj biurokrate'.

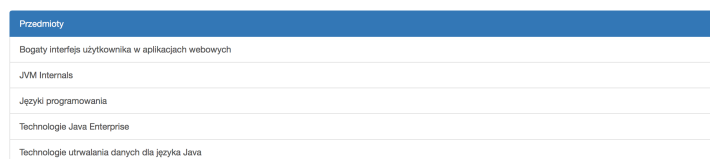
Rysunek 2.6. Formularz dodawania pracownika dziekanatu

Źródło: Własne

Jako login dla studenta posłuży jego numer albumu, natomiast wykładowca oraz pracownik dziekanatu logować się będzie do systemu dzięki loginowi składającemu się z pierwszej litery imienia oraz nazwiska bez używania polskich znaków.

2.3. Funkcjonalność dla prowadzącego zajęcia

Użytkownicy należący do grupy wykładowców muszą mieć możliwość do zarządzania listą studentów, na której widnieją ich nazwiska przypisane do przedmiotów, które prowadzą. Wykładowca ma dostęp wyłącznie do przedmiotów przez siebie prowadzonych. Dzięki temu, podobnie jak w przypadku administratorów, do interesujące wykładowcę dane, można uzyskać w prosty sposób, od razu po zalogowaniu. A prowadzący jest w stanie efektywnie zarządzać swoimi podopiecznymi. Na stronie głównej wykładowca znajdzie listę przedmiotów, które wykłada.



Przedmioty
Bogaty interfejs użytkownika w aplikacjach webowych
JVM Internals
Języki programowania
Technologie Java Enterprise
Technologie utrwalania danych dla języka Java

Rysunek 2.7. Lista przedmiotów wykładowcy

Źródło: Własne

Po wybraniu przez użytkownika interesującego go przedmiotu na ekranie pojawia się lista studentów, zapisanych na dane zajęcia. Przy każdym uczestniku znajdują się jego imię i nazwisko, numer indeksu oraz rozwijane listy z wyborem oceny za zaliczenie ćwiczeń, a także ocena za egzamin końcowy.

Seminarium magisterskie 3

Uczestnicy

Indeks	Imię	Nazwisko	Ocena z ćwiczeń	Ocena z egzaminu
186410	Szymon	Domurat	2	2
194925	Mateusz	Kwiatkowski	2	2
195032	Marta	Lewandowska	2	2
186426	Damian	Matulewski	2	2

Rysunek 2.8. Fragment listy studentów zapisanych na przedmiot

Źródło: Własne

2.4. Funkcjonalność dla studenta

Studenci są grupą, która ma najmniej do powiedzenia w działaniu aplikacji, ale to studenci najbardziej oczekują szybkiego umieszczenia danych w indeksie. Jedyną funkcjonalnością, z której mogą korzystać studenci, jest przegląd własnych osiągnięć w nauce. Po zalogowaniu do systemu student otrzymuje listę przedmiotów, w których uczestniczy. Po wejściu w interesujący użytkownika przedmiot, pojawiają się informacje o prowadzącym zajęcia, oraz oceny z ćwiczeń i egzaminu.

Przedmiot	Prowadzący	Ocena z ćwiczeń	Ocena z egzaminu
Seminarium magisterskie 3	dr Włodzisław Bzyl	4.5	5

Rysunek 2.9. Oceny studenta z wybranego przedmiotu

Źródło: Własne

2.5. Funkcjonalność dla pracownika dziekanatu

Kiedy już wykładowca wystawi zaliczenia w elektronicznym indeksie, nadchodzi czas, gdy zarówno wykładowca jak i student, muszą rozliczyć się z tych danych w dziekanacie. Zadaniem jego pracownika jest utrzymanie porządku oraz spójności danych pomiędzy tymi, które znajdują się w elektronicznej platformie oraz jej papierowym odpowiedniku. Po zalogowaniu do systemu, użytkownikowi pojawi się lista studentów oraz lista wykładowców. Gdy pracownik dziekanatu wybierze z listy interesującego go studenta, na ekranie pojawi się lista przedmiotów danego ucznia wraz z jego osiągnięciami w nauce, co pozwoli na szybką weryfikację czy dane, które dostarczył student zgadzają się z tymi w aplikacji.

Przedmiot	Prowadzący	Ocena z ćwiczeń	Ocena z egzaminu
Seminarium magisterskie 1	dr Włodzimierz Bzyl	3	3.5
Seminarium magisterskie 2	dr Włodzimierz Bzyl	3.5	4
Seminarium magisterskie 3	dr Włodzimierz Bzyl	4.5	5

Rysunek 2.10. Oceny studenta

Źródło: Własne

Jeśli wybrany zostanie z listy wykładowca, pojawi się na ekranie lista przedmiotów, które dany pracownik prowadzi. Z nowo wyświetlonej listy pracownik dziekanatu może wybrać interesujący go przedmiot, po czym na ekranie pojawią się wszyscy studenci, zapisani na wybrany przedmiot razem ze swoimi osiągnięciami, co pozwoli na szybką weryfikację czy dane dostarczone przez wykładowcę zgadzają się z tymi w elektronicznej platformie.

Przedmiot	Prowadzący	
Seminarium magisterskie 3	dr Włodzimierz Bzyl	
Seminarium magisterskie 3		
Uczestnicy		
Indeks	Ocena z ćwiczeń	Ocena z egzaminu
165334	5	4
179796	4,5	4,5
186410	4	4,5

Rysunek 2.11. Fragment listy uczestników zajęć

Źródło: Własne

Aplikacja Elektroniczny indeks w Meteor

3.1. Opis wybranych technologii użytych w pracy

Zanim rozpocznie się proces tworzenia aplikacji, programista musi zdecydować jakich technologii chce użyć do tego celu oraz jakie technologie będą do danego programu najodpowiedniejsze.

3.1.1. Javascript

Do stworzenia aplikacji elektroniczny indeks wykorzystano język javascript, którego zaletami są niewielkie rozmiary skryptów dzięki czemu krótszy jest czas ładowania z serwera, a co za tym idzie mniejsze obciążenie sieci, jest niezależny od systemu, a także nie wymaga kompilatora.

3.1.2. Meteor

W pracy użyto framework *Meteor* w wersji 1.1.0.2. Jest to framework javascriptowy, który zapewnia aplikacji działanie w czasie rzeczywistym / w skrócie RTA / ¹, dzięki czemu użytkownikom korzystającym z aplikacji, widoki aktualizują się natychmiast gdy w bazie danych zachodzą zmiany. Dodatkową zaletą tej technologii jest fakt, że wszystko, zarówno back-end jak i front-end piszemy w taki sam sposób, korzystając jedynie z javascriptu. *Meteor* pozwala również zaoszczędzić czas dostarczając deweloperom gotowe rozwiązania w postaci pakietów. Część pakietów jest wbudowana w framework i wystarczy dodać je do projektu:

Listing 3.1. Dodanie standardowego pakietu do projektu

```
meteor add nazwa_pakietu
```

¹ang. Real-Time Application - RTA

Istnieją również pakiety stworzone przez społeczność które można przeglądać na atmosphere.com. Instalujemy je w bardzo podobny sposób jak pakiety dostarczone w standardzie:

Listing 3.2. Dodanie zewnętrznego pakietu do projektu

```
meteor add autor:nazwa_pakietu
```

3.1.3. MongoDB

MongoDB jest najpopularniejszą nierelacyjną bazą danych. 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. Jest to także obecnie jedyna oficjalnie wspierana baza przez twórców *Meteor*. W pracy wykorzystana została w wersji 3.0.2.

3.1.4. Velocity

Velocity jest oficjalnym narzędziem do testowania aplikacji napisanych w *Meteor* w wersji 1.0+. *Velocity* pozwala tworzyć testy w popularnych bibliotekach takich jak *mocha*, *jasmine*, *cucumber* czy *selenium*. W pracy wykorzystana została biblioteka *jasmine*

3.1.5. Distributed Data Protocol

Distributed Data Protocol jest to prosty protokół oparty o format tekstowy JSON. Służy on do pobierania danych z serwera oraz aktualizacji pobranych danych gdy tylko dane na serwerze ulegną zmianie.

3.1.6. Roles

Meteor-roles jest to pakiet autoryzujący do frameworka *Meteor*, który pozwala zarządzać jakie dane zostaną wysłane do konkretnych grup użytkowników.

3.1.7. Iron router

Iron router jest to pakiet, którym definiujemy jak ma wyglądać mapa strony. Pakiet ten działa zarówno po stronie serwera i klienta. Routing po stronie klienta sprawia, że aplikacja jest naprawdę szybka gdy już jest załadowana, ponieważ przy każdej zmianie podstrony nie trzeba generować całej podstrony.

3.1.8. Accounts

Meteor account jest to kompletny pakiet zarządzania kontami użytkowników. Jedną linią kodu można zapewnić aplikacji możliwość logowania, tworzenia kont, walidacji email, przywracania hasła czy logowania się przez zewnętrzne serwisy jak *facebook* czy *twitter*. Dodatkowo pakiet daje możliwość dostosowania go pod własne potrzeby.

3.1.9. Underscore string

Underscore.string jest to pakiet służący do manipulacji stringami. Pierwotnie był rozszerzeniem *Underscore.js*, obecnie jest niezależną biblioteką.

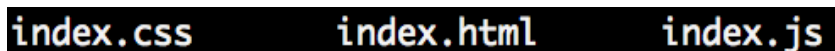
3.2. Opis tworzenia aplikacji

Tworzenie aplikacji we frameworku *Meteor* zaczyna się od utworzenia nowego projektu poleceniem w konsoli:

Listing 3.3. Tworzenie projektu

```
meteor create nazwa_projektu
```

Po wykokaniu tej komendy zostanie utworzony folder z projektem, w którym znajdują się trzy pliki.

A diagram showing three files: `index.css`, `index.html`, and `index.js`. The files are arranged horizontally and each is enclosed in a rounded rectangular box with a light blue border. The boxes are slightly overlapping.**Rysunek 3.1.** Nowy projekt

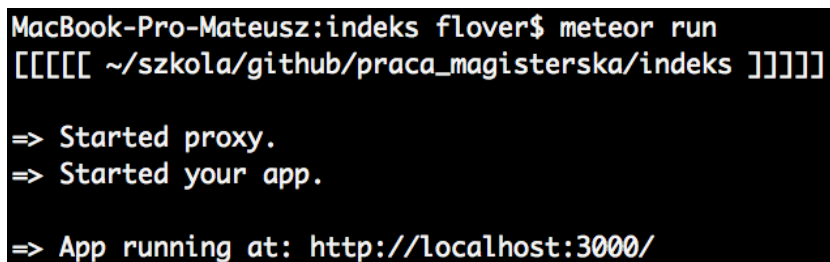
Źródło: Własne

Aby uruchomić aplikację musimy uruchomić lokalny serwer, który pozwoli zobaczyć przetworzony javascript w przeglądarce oraz uruchomić bazę na lokalnym komputerze. W konsoli zmieniamy lokalzację na folder z projektem, a następnie wprowadzamy następującą komendę:

Listing 3.4. Uruchomienie aplikacji

```
meteor run
```

Jeżeli na ekranie pojawi się

A screenshot of a terminal window with a black background and white text. The prompt is `MacBook-Pro-Mateusz:indeks flover$`. The command `meteor run` has been entered. The output shows the directory `~/szkola/github/praca_magisterska/indeks` and the status of the application: `=> Started proxy.`, `=> Started your app.`, and `=> App running at: http://localhost:3000/`.

```
MacBook-Pro-Mateusz:indeks flover$ meteor run
[[[[[ ~/szkola/github/praca_magisterska/indeks ]]]]]

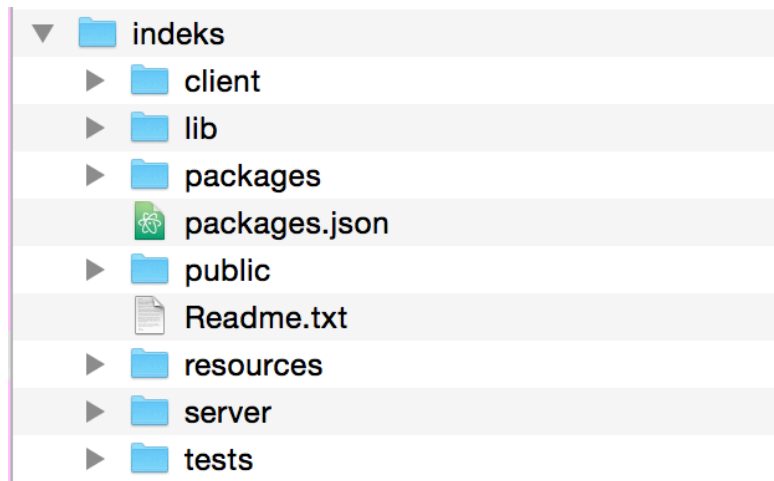
=> Started proxy.
=> Started your app.

=> App running at: http://localhost:3000/
```

Rysunek 3.2. Start aplikacji bez błędów

Źródło: Własne

oznacza to, że aplikacja uruchomiła się bez błędów i od teraz można z niej korzystać pod podanym adresem. Na początku warto podzielić projekt na część serwerową, kliencką oraz dostępną zarówno dla części serwerowej oraz klienckiej.

**Rysunek 3.3.** Podział projektu

Źródło: Własne

Warto również usunąć pakiety *autopublish* oraz *insecure* dzięki czemu zwiększone zostanie bezpieczeństwo aplikacji. Usunięcie pakietu *insecure* blokuje możliwość użytkownikom do zarządzania bazą danych po stronie klienta, natomiast wyrzucenie *autopublish* spowoduje, że trzeba samemu zadbać o wysyłanie danych z serwera do klienta.^[9]

Listing 3.5. Usuwanie pakietów

```
meteor remove autopublish
meteor remove insecure
```

W części klienckiej trzymane są widoki aplikacji, funkcje pomocnicze oraz router, a po stronie serwera plik *publish*, który definiuje jakie dane zostaną wysłane do użytkowników. Po podzieleniu projektu trzeba utworzyć kolekcje, w których będziemy trzymać informacje o przedmiotach, oraz ocenach studentów zdobytych na zajęciach. Do tego celu utworzymy kolekcję *subjects*, w której będzie trzymana nazwa przedmiotu, prowadzący, semestr, na którym przedmiot się odbywa oraz lista zapisanych studentów. Kolekcja *grades* będzie łączyła informacje z kolekcji z użytkownikami oraz przedmiotami. Znajdą się tu *login* oraz *id* użytkownika,

oraz id, nazwa przedmiotu, prowadzący przedmiot oraz oceny z ćwiczeń i egzaminu końcowego.

```
Subjects = new Mongo.Collection('subjects');  
Grades = new Mongo.Collection('grades');
```

Listing 1: Utworzenie kolekcji

Kiedy już mamy kolekcje, trzeba utworzyć widoki. Zaczynamy od widoku strony głównej, na której w zależności od grupy do której przynależeć będzie użytkownik będą wyświetleni studenci, przedmioty oraz wykładowcy

```
<template name="subjectList">  
  <div class="container">  
    <div class="subject-list">  
      <div class="list-group">  
        {{#if isInRole 'admin, wykładowca, student'}}  
        <a class="list-group-item active">  
          Przedmioty  
        </a>  
        {{#if isInRole 'student'}}  
        {{#each mySubjects}}  
        <a href="/subjects/{{id}}" class="list-group-item">{{subject}}</a>  
        {{/each}}  
        {{/if}}  
      </div>  
    </div>  
  </div>  
</template>
```

Listing 2: Template wyświetlający wszystkie przedmioty, na które uczęszcza student

Analogicznie działa wyświetlanie przedmiotów dla wykładowcy czy administratora. W tym momencie nawet jeśli w bazie będą wprowadzone przedmioty, nadal będzie pusta strona. Do przesyłu danych wykorzystamy *iron-router*.

```

waitOn: function () {
  return [ Meteor.subscribe('theSubjects') ];
},
onBeforeAction: function () {
  if(!Meteor.user()){
    this.layout('appLayout');
    this.render('login');
  }
  else {
    this.next();
  }
},
action: function () {
  this.layout('appLayout');
  this.render('subjectList', {
    'data': {
      'User': Meteor.user(),
      'mySubjects': Subjects.find(),
      'teacherSubjects': Subjects.find({'leading': Meteor.user().profile.title + " " + Meteor.user().profile.firstName
        ↪ + " " + Meteor.user().profile.lastName}, {sort: {'subject': 1}}),
      'myStudents': Meteor.users.find({'roles': 'student'}, {sort: {'profile.lastName': 1, 'profile.firstName': 1,
        ↪ 'username': 1}}),
      'myLeaders': Meteor.users.find({'roles': 'wykładowca'}, {sort: {'profile.lastName': 1, 'profile.firstName': 1,
        ↪ 'username': 1}}),
      'myGrades': Grades.find()
    }
  });
}
}

```

Listing 3: Powyższy fragment renderuje widok z przedmiotami oraz użytkownikami

W drugiej linii kodu widzimy return, który zwraca zasubskrybowane dane z theSubjects. Żeby te dane można było odebrać, serwer musi je opublikować.

```

Meteor.publish('theSubjects', function () {
  if(this.userId){
    if (Roles.userIsInRole(this.userId, 'admin')) {
      return [Meteor.users.find({}), Subjects.find({})];
    } else if (Roles.userIsInRole(this.userId, 'dziekanat')){
      return [Meteor.users.find({}), Subjects.find({})];
    } else if (Roles.userIsInRole(this.userId, 'student')){
      var user = Meteor.users.findOne({'_id': this.userId})._id;
      return Subjects.find({'students': user});
    } else if (Roles.userIsInRole(this.userId, 'wykładowca')){
      return [Subjects.find({}), Meteor.users.find({})];
    }
  }
  else {
    this.ready();
  }
});

```

Listing 4: udostępnianie danych poszczególnym grupom użytkowników

Dzięki pakietowi *meteor-roles* jesteśmy w stanie dokładniej określić jakie dane chcemy wysyłać poszczególnym grupom użytkowników. Dzięki temu w szybki sposób możemy również zarządzać treściami które pojawiają się na widokach dzięki blokom pomocniczym.

W aplikacji znajduje się szereg managerów, które przekazują operacje wykonane przez użytkownika do metod, które zostaną wykonane na serwerze. Funkcja *assignSubject*, która zostaje wywołana w momencie gdy użytkownik chce przypisać studenta do przedmiotu, pobiera id przedmiotu, nazwę przedmiotu, nazwę użytkownika, id użytkownika oraz prowadzącego zajęcia i przekazuje je do metody *addStudentToSubject*, która umieszcza te informacje w kolekcji *grades*, jednocześnie przypisując id studenta do listy osób zapisanych na przedmiot w kolekcji *subjects*, a także id przedmiotu przypisuje liście przedmiotów, na które student uczęszcza w kolekcji użytkowników. Funkcje *updateExerciseGrade* i *updateExamGrade* działają w taki sam sposób. Gdy wykładowca zatwierdza zmianę oceny, pobierają one z pola ocenę jaką zatwierdził prowadzący, a następnie przekazują id studenta, id przedmiotu oraz ocenę do metod o tych samych nazwach. Dzięki temu, że wysłane zostały także id przedmiotu i studenta, możliwa jest aktualizacja oceny konkretnego studenta dla konkretnego przedmiotu. Gdy administrator wysyła formularz *form* aby dodać nowy przedmiot, z pól formularza zostają pobrane informacje o nazwie przedmiotu, prowadzącym przedmiot oraz informacja, na którym semestrze zajęcia się odbywają, a następnie są one przesłane do metody *addSubject*, która umieszcza przedmiot w kolekcji *subjects*. W trakcie tworzenia nowego użytkownika, zostaje mu przypisana jedna z czterech dostępnych ról. Jeśli tworzenie użytkownika zakończy się powodzeniem, zostaje wywołana metoda *assignRole*, która otrzymuje stworzonego użytkownika oraz rolę do której będzie przypisany i ustawia danemu użytkownikowi wybraną rolę.

[10] [11] [12] [13] [14]

3.3. Opis testowania aplikacji

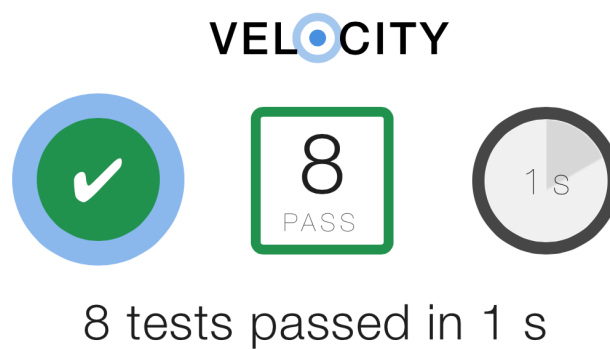
Od wersji *Meteor* 1.0, zostało oficjalnie udostępnione narzędzie do testowania aplikacji o nazwie *Velocity*. Instaluje się ono razem z jednym z czterech dostępnych fra-

meworków do testowania. W tej pracy użyty został framework *Jasmine*. Aby dodać go do projektu wystarczy w konsoli wpisać

Listing 3.6. Instalacja Velocity, Jasmine i html reporter

```
meteor add sanjo:jasmine
meteor add velocity:html-reporter
```

Html-reporter jest to reaktywny plugin, który prezentuje wyniki testów.



Rysunek 3.4. Html-reporter

Źródło: Własne

```
describe('Set user a role', function () {
  it('Should assign user to role', function () {
    if (Meteor.users.find().count() === 0) {
      Accounts.createUser({
        username: 'username1',
        email: 'username1@test.pl',
        password: 'abcdef',
        profile: {
          name: 'student',
          firstName: 'firstName1',
          lastName: 'lastName1',
          subjects: []
        }
      });
      Accounts.createUser({
        username: 'username2',
        email: 'username2@test.pl',
        password: 'abcdef',
        profile: {
          name: 'student',
          firstName: 'firstName2',
          lastName: 'lastName2',
          subjects: []
        }
      });
    }

    user = Meteor.users.find({'username': 'username1'}).fetch();

    Meteor.call('assignRole', user[0], 'student');
    expect(Meteor.users.findOne({'username': 'username1'}).roles[0]).toBe('student');

    Meteor.users.remove({'username': 'username1'});
    Meteor.users.remove({'username': 'username2'});
  });
});
```

Listing 5: Test przypisywania roli użytkownikowi

describe mówi nam jaka funkcjonalność będzie testowana, *it* opisuje, co dana funkcjonalność powinna robić, następnie do replikowanej bazy na potrzeby testu dodajemy użytkowników. Gdy już są utworzeni bierzemy użytkownika i wyłuskujemy wszystkie informacje o nim. Wywołujemy metodę, która przypisuje użytkownikowi rolę, a następnie sprawdzamy czy naszemu użytkownikowi została przypisana rola, której oczekiwaliśmy. Po wszystkim czyścimy zreplikowaną bazę.

ROZDZIAŁ 4

Pakiet walidujący operacje elektronicznego indeksu

4.1. Funkcjonalność pakietu

4.2. Opis tworzenia pakietu

[16] [11] [9]

4.3. Implementacja pakietu w aplikacji

4.4. Przetestowanie pakietu

[17]

Zakończenie

DODATEK A

Mesosphere

DODATEK B

TinyTest

Bibliografia

- [1] PWN. [Walidacja](#). *SJP PWN*, 2014.
- [2] Włodzimierz Gajda. [Walidacja danych](#). *gajdaw*, 2008.
- [3] Wikipedia. [Walidacja \(technika\)](#). *Wikipedia*, 2014.
- [4] Piotr Furtak. [Jak zrozumieć różnicę między weryfikacją a walidacją systemu, by nie mieć problemu podczas odpowiadania na egzaminie ISTQB?](#) *ALT-KOM Akademia*, 2014.
- [5] Wikipedia. [Weryfikacja i walidacja oprogramowania](#). *Wikipedia*, 2014.
- [6] Wikipedia. [Aspekty ekonomiczne procesu walidacji](#). *DPK Consulting*, 2008.
- [7] Wikipedia. [Verification and validation](#). *Wikipedia*, 2014.
- [8] W. Szkolnikowski A. Łobzowski. [Co firma LAB-EL ma do powiedzenia w tematyce GAMP 4 \(cz. 2\)](#). *LAB-EL*, 2009.
- [9] Tom Coleman and Sacha Greif. [Discover Meteor: Building Real-Time JavaScript Web Apps](#). First edition edition, 2013.
- [10] Stephen Walther. [An Introduction to Meteor](#). *stephenwalther*, 2013.
- [11] MeteorJS. [Meteor Documentation](#), 2014.
- [12] MongoDB. [The MongoDB Manual](#), 2014.
- [13] Kristina Chodorow. [Scaling MongoDB](#). First edition edition, 2011.
- [14] Rick Copeland. [Scaling with MongoDB](#). page 32, 2012.
- [15] Arunoda Susiripala. [Test Driven Development with Meteor](#). *Sitepoint*, 2013.
- [16] Stephen Walther. [Creating a basic meteorite smart package](#). *Vintyre Inc*, 2012.
- [17] Eventedmind. [Testing Packages with Tinytest](#), 2013.

Spis tabel

Spis rysunków

1.1.	Różnice między weryfikacją, walidacją	10
1.2.	Weryfikacja i walidacja	11
1.3.	Cykl życia systemu	15
2.1.	Menu dodawania	19
2.2.	Fragment listy studentów do zapisu na zajęcia	20
2.3.	Formularz dodawania przedmiotu	20
2.4.	Formularz dodawania studenta	21
2.5.	Formularz dodawania wykładowcy	21
2.6.	Formularz dodawania pracownika dziekanatu	22
2.7.	Lista przedmiotów wykładowcy	22
2.8.	Fragment listy studentów zapisanych na przedmiot	23
2.9.	Oceny studenta z wybranego przedmiotu	23
2.10.	Oceny studenta	24
2.11.	Fragment listy uczestników zajęć	24
3.1.	Nowy projekt	28
3.2.	Start aplikacji bez błędów	28
3.3.	Podział projektu	29
3.4.	Html-reporter	33

Oświadczenie

Ja, niżej podpisany(a) oświadczam, iż przedłożona praca dyplomowa została wykonana przeze mnie samodzielnie, nie narusza praw autorskich, interesów prawnych i materialnych innych osób.

.....

data

.....

podpis