Politechnika Wrocławska

Wydział Informatyki i Zarządzania

Kierunek studiów: Informatyka

Praca dyplomowa – inżynierska

# APLIKACJA WSPOMAGAJĄCA ZDALNE SZACOWANIE HISTORYJEK UŻYTKOWNIKA METODĄ PLANISTYCZNEGO POKERA.

Karol Kowalski

słowa kluczowe:
Firebase, ReactJS, FLUX, planning poker, estymacja

krótkie streszczenie:
W pracy przedstawiono projekt aplikacji umożliwiającej przeprowadzenie rozgrywki planning pokera online z użyciem ReactJS oraz Firebase.

| | | | |
|---|---|---|---|
| Opiekun pracy dyplomowej | dr. hab. inż. prof. PWr. Trawiński Bogdan | ..... | ..... |
| | Tytuł/stopień naukowy/imię i nazwisko | ocena | podpis |
| Ostateczna ocena za pracę dyplomową | | | |
| Przewodniczący Komisji egzaminu dyplomowego | | ..... | ..... |
| | Tytuł/stopień naukowy/imię i nazwisko | ocena | podpis |

*Do celów archiwalnych pracę dyplomową zakwalifikowano do:**

*a) kategorii A (akta wieczyste)*

*b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)*

** niepotrzebne skreślić*

pieczątka wydziałowa

Wrocław, rok 2018

## SPIS TREŚCI

## SPIS RYSUNKÓW

## SPIS LISTINGÓW

## SPIS TABEL

**STRESZCZENIE**

Celem pracy było opracowanie aplikacji zaprojektowanej do gry w planning pokera zdalnie. Dostępne aplikacje spełniają swoje zadanie, jednak autor nie znalazł aplikacji, która byłaby silnie zintegrowana z Github'em. Dlatego wyróżniającą cechą jego aplikacji będzie możliwość importu historyjek z Github'a które są tam w postaci issues oraz exportu ocen do Github'a w postaci etykiet. W ramach pracy autor stworzył aplikację opartą o architekturę FLUX z bazą danych firebase jako backend. Dzięki temu rozwiązaniu planning poker może być rozgrywany w czasie rzeczywistym. Oprócz tego praca zawiera omówienie pewnych metod zarządzania zwinnego. Użyteczność aplikacji będzie sprawdzona w oparciu o wyniki ankiety, które zostaną zawarte w niniejszej pracy.

**ABSTRACT**

The main goal of this thesis was development of web app designed to play planing poker online. Avaliable aplications fulfil their role but however author haven't found application, which would be strongly integrated with Github. That's why distinctive feature of author's app will be possibility is importing user stories from Github projects which are there in the shape of issues and exporting story points to Github in form of labels. Within work author made aplication based on FLUX architecture with firebase database as backend. Thanks this solution planning poker can be played in real time. Except of app description thesis describe some of agile managment and estimating methods. Usability of application will be tested thanks usability survey which reasults are consist within thesis.

# WSTĘP

Planowanie to odpowiadanie na pytanie "Co powinniśmy stworzyć i kiedy?". Jednak aby odpowiedzieć na to pytanie powinniśmy zadać również pytania o estymację ("Jak duże to jest?") oraz harmonogram ("Kiedy będzie skończone?" oraz "Ile będzie zrobione do tego czasu?"). Estymowanie i planowanie są bardzo istotne w sukcesie każdego projektu. Plany pomagają inwestorom podjąć decyzję. Na przykład możemy zacząć specyficzny projekt, jeżeli oszacujemy, że zajmie sześć miesięcy oraz będzie wymagać milion dolarów, ale odrzucimy go, jeżeli stwierdzimy, że zabierze nam dwa lata oraz 4 miliony dolarów.[3]

## OPIS PROBLEMU

W dzisiejszym świecie coraz więcej projektów jest tworzonych przez zespoły rozproszone. Zespół rozproszony to taki, którego członkowie realizują jeden projekt, cel i zadania w określonym czasie, pracując z różnych miejsc – kontynentów, krajów, miast, budynków czy biur. Zespoły rozproszone pracują jak tradycyjne zespoły zadaniowe, ale mają ze sobą na co dzień kontakt wirtualny i widują się sporadycznie.[12] Przez co nie mogą się spotykać co sprint w jednym pomieszczeniu by oszacować zadania do wykonania w sprincie. Dlatego muszą się spotkać w wirtualnym pokoju w chmurze. Jednak muszą też swoje estymacje mieć zapisane w centralnym miejscu, gdzie mieści się ich projekt. Bardzo często tym miejscem jest np. Github.

## CEL PRACY

Zadaniem jakie jakie postawiłem przed sobą jest stworzenie aplikacji umożliwiającej rozegranie planning pokera w czasie rzeczywistym w raz z rolami product ownera, scrum mastera oraz gracza by jak najwierniej zasymulować rozgrywkę w planning pokera. Dodatkowym celem jest synchronizowanie tego wszystkiego z aplikacją Github.

## ZAKRES PRACY

Praca obejmowała opracowanie projektu aplikacji, implementację w frameworku ReactJS oraz wdrożenie wszystkiego powyższego w bazie danych Firebase oraz zhostowanie tego wszystkiego na hostingu Firebase'a. Dodatkowym zadaniem jest sprawdzenie użyteczności programu za pomocą ankiety.

# 1. WPROWADZENIE

Najpopularniejszą definicją projektu jest definicja Project Management Institute (PMI – międzynarodowe stowarzyszenie zrzeszające kierowników projektów. Project Management Institute powstał w 1969 w Pensylwanii w USA jako stowarzyszenie non profit zrzeszające profesjonalistów w dziedzinie zarządzania projektami- istnieje również oddział we Wrocławiu.): „Projekt, to tymczasowa działalność podejmowana w celu wytworzenia unikatowego wyrobu, dostarczenia unikatowej usługi lub otrzymania unikatowego rezultatu".[5] Już w starożytnym Egipcie istniały metody zarządzania skomplikowanym przedsięwzięciem, np. budowa piramid. Było to olbrzymie wyzwanie, które wymagało wiedzy zarówno planistycznej, jak i logistycznej. W ubiegłym stuleciu, w latach 50 stosowano podejścia zwane obecnie współczesnymi technikami zarządzania projektami. Weźmy np. projekt systemu rakiet balistycznych Polaris. Okazał się on swoistym koszmarem technicznym i administracyjnym. Nad projektem pracowała olbrzymia ilość zespołów badawczych, projektowych i produkcyjnych. Dla udokumentowania wszystkich działań zużyto tony papieru, a samo zarządzanie projektami zaczęto uznawać za dziedzinę bardzo skomplikowaną, niedostępną, opartą na wiedzy specjalistów.[11]

## 1.1. NAJPOPULARNIEJSZE METODY ZARZĄDZANIA PROJEKTAMI

Przy dokonywaniu wyboru metodyki zarządzania projektem należy przeprowadzić adekwatną analizę w celu doboru odpowiedniego podejścia, gdyż każda z metodyk posiada wady i zalety. Wyróżniamy dwa podejścia (klasyczne i zwinne), które różnią się dość mocno między sobą w kilku płaszczyznach (przekrojach),takich jak: odpowiedzialność za produkt, rola menedżera w zespole, istota prac wstępnych, zdefiniowanie produktu czy odpowiedź zwrotna użytkowników.Co uwzględniono w tabeli 1.1.

Tabela 1.1. Różnice między podejściem klasycznym a zwinnym

| Płaszczyzna | Podejście klasyczne | Podejście zwinne |
|---|---|---|
| **Odpowiedzialność za produkt** | Podzielona między marketera, menadżera produktu i menadżera projektu | Istnieje tylko jeden właściciel produktu |
| **Rola menedżera w zespole** | oddzielony od zespołów deweloperskich | Jest członkiem zespołu i ściśle z nim współpracuje. |
| **Istota prac wstępnych** | Przeprowadzane są szczegółowe badania rynku, planowanie produktu i analizy biznesowe | Ograniczają się do stworzenia wizji, która ogólnie opisuje wygląd i działanie produktu. |
| **Zdefiniowanie produktu** | Wymagania są określane i zatwierdzane w początkowej fazie | Produkt odkrywany jest stopniowo, a wymagania krystalizują się w trakcie |
| **Odpowiedź zwrotna** | Dostępna po wypuszczeniu produktu na rynek | Wczesna i częsta odpowiedź zwrotna po małych wdrożeniach |

### 1.2. PODEJŚCIE KLASYCZNE

Podejście klasyczne, reprezentowane przez PMBoK (Kompendium wiedzy o zarządzaniu projektami) lub metodykę PRINCE (kompleksowa metodą zarządzania projektami, zalicza się ją do podejścia klasycznego) oraz jej następcę PRINCE2, ma na celu wytworzenie kompletnego produktu przy uprzednim, dokładnym określeniu jego cech. Takie podejście charakteryzuje się ogromnym formalizmem, weźmy na przykład dokonywanie zmian, które wiąże się z wypełnianiem dokumentów (RfC – ang. Request for Change) -prośby o zmianę. Każda odpowiedź, to z kolei oczekiwanie, aż zostanie przeanalizowana i zatwierdzona bądź odrzucona. Dodatkowo osoby odpowiedzialne zwykle nie pracują wraz z zespołem, w związku z czym często występują bariery komunikacyjne.[1]

### 1.3. PODEJŚCIE ZWINNE

Podejście zwinne ukierunkowane jest na zespół, który w pełni odpowiada za wykonanie swojej części zadania i stopniowo dostosowuje je do potrzeb przyszłych użytkowników. Przykładowe metodyki zwinne, to m.in.: Scrum, Lean, Cobit, SixSigma, Kanban, XP (ang. eXtream Programming), TDD (ang. Test-Driven Development) i FDD (ang. Feature-Driven Development). W praktycznej działalności często zespoły nie wykorzystują jednej metodyki, a opierają się na kilku. Przykładem może być tutaj niedawno powstały Scrum-ban, który jest połączeniem dobrych praktyk zaczerpniętych ze Scruma i Kanbana.[15] Ciekawym rozwiązaniem jest także TDD, czyli programowanie sterowane testami. Proponuje ono utworzenie przypadków testowych, zanim powstanie fragment kod.[10] Każde z prezentowanych rozwiązań posiada swoje zalety, jednak najważniejsze jest dobranie odpowiedniej metodyki do realiów pracy i prawidłowa adaptacja względem realiów biznesowych, gdyż

ścisłe stosowanie wszystkich praktyk może być nadmiernie pracochłonne w zastosowaniu do małych projektów.

### 1.4. MANIFEST AGILE

Manifest Agile powstał w 2001 roku, ale nie jest to sam początek tego ruchu zwinnego oprogramowania. Już wcześniej istniały pewne metodyki, jak również istniały podstawy teoretyczne dla wprowadzenia takich rozwiązań. Jeśli chodzi o podstawy teoretyczne, to trzeba zwrócić uwagę przede wszystkim na 3 kwestie:

— kwestię podejścia systemowego i szkoły systemowej (czyli lata 70-te XX w.), która dostarczyła dość znaczącej wiedzy pozwalającej na współczesne zarządzanie projektami;
— drugi aspekt, to zarządzanie jakością, koncepcje, metody zarządzania jakością, które są wykorzystywane bardzo mocno w metodykach zwinnych;
— trzeci aspekt, to zarządzanie wiedzą, czyli chociażby Takeuchi i Nonaka, którzy jako pierwsi wspomnieli o idei młyna (w artykule The new product development game, opublikowanym w „Harvard Business Review" w 1986 r.), skąd wzięła się później metoda „scrum".

W latach 90-tych zaobserwowano znaczące skomplikowanie oprogramowania, tworzenia oprogramowania. Projekty dotychczas zarządzane klasycznie okazały się zbyt mało elastyczne, nie pozwalały na wystarczająco szybkie tworzenie oprogramowania. Zdecydowano więc, że trzeba jakoś zmodyfikować sposób, w jaki tworzymy oprogramowanie, aby odpowiadać na potrzeby klientów, na potrzeby rynku wystarczająco szybko. Pierwsze próby podjęto już na początku lat 90-tych XX w. Zostały one uwieńczone publikacją w 1995 r. metodyki scrum, opisu, w jaki sposób można stosować tą metodykę. Rok później pojawiła się Metodyka Extreme Programming, a więc już w połowie lat 90-tych mieliśmy metodyki zwinne, które stosowane były najpierw w ograniczonym zakresie, potem coraz szerzej. Również poszczególne organizacje, przedsiębiorstwa zaczęły tworzyć swoje odmiany tych metod. Tak więc dziś mamy całe bogactwo metodyk związanych z zarządzaniem zwinnym w projektach. Agile nie było zatem pierwsze, było po prostu pewnym podsumowaniem pierwszego etapu rozwoju tych metodyk zwinnych. W 2001 r. powstał Manifest, który określił, co jest ważne w zwinnym zarządzaniu projektem. rys.1.1


Rys. 1.1. Manifesto for Agile Software Dev.[www.medium.com]

Ten Manifest pokazywał pewien system wartości, co jest ważniejsze, a co mniej ważne w zarządzaniu projektem. Mamy więc takie cztery porównania:

— Autorzy Manifestu twierdzili, że ludzie i interakcje między nimi są ważniejsi, niż procesy i narzędzia- to nie znaczy, że procesy i narzędzia nie są istotne, ale są mniej istotne, mniej ważne. Trzeba położyć większy nacisk na ludzi, na interakcje- to powoduje bardziej nieformalną komunikację, jej przyspieszenie, również przyspieszenie realizowania zadań i umożliwia bardziej elastyczne realizowanie tych zadań, kiedy zmieniają się warunki;
— Druga zasadą jest orientacja bardziej na działające oprogramowanie, niż na dokumentację. Jeśli zerkniemy do starych wersji oprogramowania z lat 80-tych, 90-tych, do każdego programu dodawana była gruba instrukcja. Dzisiaj już o tym zapomnieliśmy. Dzisiaj wiele aplikacji nie ma w ogóle nawet instrukcji- mówimy, że działają intuicyjnie (przynajmniej powinny). Dzięki temu, że orientujemy się na realizację tych najważniejszych efektów w projekcie, możemy lepiej wykorzystać zasoby, możemy szybciej osiągnąć te efekty, a rzeczy mniej ważne, mniej istotne, takie właśnie, jak szczegółowa dokumentacja (jakaś dokumentacja przecież musi być), możemy odłożyć, możemy przeznaczyć dla nich mniejsze zasoby.
— Również jeśli chodzi o współpracę z klientem, w metodykach zwinnych proponuje się zmianę podejścia. Zamiast negocjować szczegółowo umowy, budujemy współpracę z tym klientem dlatego, że nie jesteśmy w stanie z góry przewidzieć, jaki będzie, tak do końca, zakres naszego projektu, co w tym projekcie zrealizujemy, co będzie potrzebne za rok, kiedy nasz produkt będzie prawie gotowy. Czy te wymagania się nie zmienią wielokrotnie, biorąc pod uwagę szybkość zmiany technologii, potrzeb, oczekiwań klientów, szybkość zmian na rynku. Zatem klient powinien być blisko, powinien dostarczać bieżące informacje o swoich potrzebach, a w kontrakcie zawieramy tylko te informacje, które są najważniejsze.
— I w końcu reagowanie na zmiany zamiast szczegółowego planowania. Oczywiście

planowanie występuje w metodykach zwinnych, ale jest ono ograniczone tylko do tego, żeby dało się zarządzać takim projektem. Natomiast przede wszystkim orientujemy się na reagowanie na zmiany: zmiany potrzeb klienta, zmiany na rynku. Na dostosowanie naszego projektu, w kolejnych iteracjach, do tego, czego klient oczekuje.

Czasem niektórzy mówią, może żartobliwie, ale nieraz całkiem serio, że jeżeli czegoś nie zaplanowali, to stosowali właśnie Agile. Nic bardziej błędnego: w Agile każda iteracja jest planowana, w każdym dniu planujemy swoją pracę, stosujemy inne metody, rzadko stosujemy harmonogram Gantta, ale także planujemy te działania. Zatem taki polski Agile („polnische Agile", jak niektórzy mówią), to przykład niewłaściwego zarządzania przedsięwzięciami i raczej nie należy się tym chwalić. Warto jednak zauważyć, że nie do każdego projektu możemy zastosować metodyki zwinne. One się lepiej sprawdzają wtedy, kiedy mamy:

— bardzo krótkie, napięte terminy;
— projekty mają charakter unikatowy;
— są skomplikowane.

Mamy do zrealizowania coś nowego, nieoczekiwanego i mało czasu. Wtedy ta metodyka zwinna rzeczywiście jest bardziej uzasadniona niż metodyki klasyczne. Stosowanie metodyk zwinnych, szczególnie żądanie tej interakcji między pracownikami ogranicza nam wielkość zespołu, a więc ogranicza nam wielkość projektu. Generalnie metodyki zwinne stosujemy:

— w małych i średnich projektach, rzadziej w projektach dużych;
— konieczne jest, aby w metodyce zwinnej dostępny był dla nas klient, klient musi się na bieżąco kontaktować z nami i mówić czego potrzebuje, jakie są jego oczekiwania, czy jest zadowolony z tego, co uzyskuje w poszczególnych iteracjach;
— tematyka projektu musi być taka, aby klient z każdej iteracji miał jakąś wartość, bowiem staramy się często wypuszczać oprogramowanie, często wprowadzać nowe jego wersje, ale to powoduje, że ta nowa wersja musi dostarczyć jakąś wartość dla klienta.

W przypadku oprogramowania jest to oczywiste. W przypadku, kiedy budujemy jakiś budynek, być może Agile wtedy nie jest aż tak przydatny. Trzeba się zastanowić, czy możemy zastosować całą metodykę Agile, czy jak współcześnie w wielu projektach, zastosować ją tylko w odniesieniu do wybranych modułów projektu, tam, gdzie rzeczywiście ma ona zastosowanie.

Więcej informacji można znaleźć w:[3]

# 2. NAJCZĘŚCIEJ WYKORZYSTYWANE NARZĘDZIA W ZARZADZANIU PROJEKTAMI.

Zarządzanie projektami informatycznymi ściśle wiąże się z wykorzystaniem narzędzi informatycznych, które wspomagają ten proces. Przy ich wyborze warto pamiętać, iż mają pomagać w pracy projektowej, a nie przeszkadzać w jej realizacji, dlatego należy wybierać je mądrze. Przykładem nieodpowiedniego doboru narzędzia może być sytuacja, w której kierownik projektu nie dotrzymuje terminów swoich prac, ze względu na zajmowanie się raportowaniem postępu prac lub aktualizacją harmonogramu.[7]

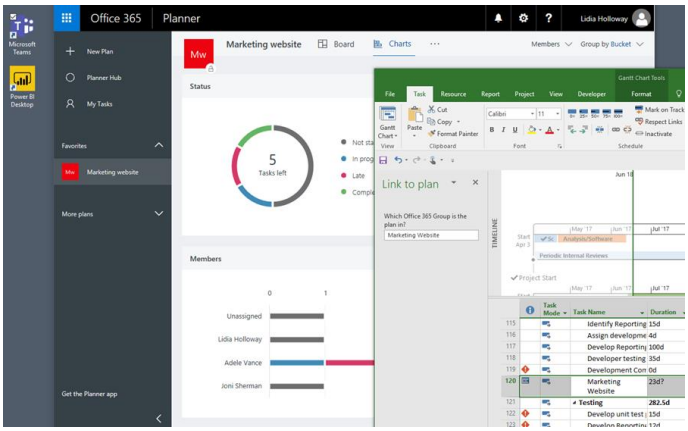
## 2.1. NARZĘDZIA STOSOWANE DLA METOD KLASYCZNYCH.

### 2.1.1. Microsoft Project

Najbardziej popularnym narzędziem stosowanym w metodykach klasycznych jest Microsoft Project. Pozwala on na rozpisanie całego harmonogramu działań, zaplanowanie budżetu czy stworzenie wykresu Gantta, tak niezbędnego w pracy kierownika. Pozwala także na tworzenie raportów, prezentacji i wykresów z postępów prac. rys.2.1

### 2.1.2. Gantt Project

W niektórych przedsiębiorstwach w zarządzaniu projektami używa się programu Gantt Project. Jest to darmowe narzędzie umożliwiające dynamiczne tworzenie diagramów Gantta z podziałem na poszczególne zadania wraz z rozplanowaniem ich w czasie. Dodatkowo pozwala ono na tworzenie wykresów PERT (ang. Program Evaluation and Review Technique) wraz ze ścieżkami krytycznymi.[9]




Rys. 2.1. Microsoft Project[www.microsoft.com]

## 2.2. NAJPOPULARNIEJSZE NARZĘDZIA STOSOWANE DLA METOD ZWINNYCH ( TEAM ESTIMATION GAME, KANBAN, PLANNING POKER).

W małych zespołach projektowych, które wykorzystują metodyki zwinne, odchodzi się zazwyczaj od Microsoft Project czy GanttProject stosując aplikacje webowe. Przykładem takich aplikacji są: Jira, Mantis, TFS, Trello, Sllack, KanbanTool. Przy metodykach zwinnych, właściciel produktu tworzy rejestr produktowy (ang. Product Backlog), który ma formę listy zawierającej wymagania klienta z określoną wagą (priorytetem) i czasochłonnością.[13] Wymienione wyżej narzędzia pozwalają na taką pracę i umieszczanie danych w chmurze, dzięki czemu każdy członek zespołu ma do nich dostęp, niezależnie od lokalizacji, czy pracuje w biurze czy poza nim. Zadania w backlogu[1] powinny być rozpisane dla obecnego sprintu [2] oraz zawierać dodatkowe zadania, które będą zasilać kolejny sprint bądź zostaną wykonane w istniejącym, gdy nadarzy się taka możliwość. Takie odejście do pracy umożliwia wykonanie części zadań przed wyznaczonym czasem. Narzędzia te pozwalają również na przypisanie konkretnego zadania do danego użytkownika, dzięki czemu każdy pracownik zna zakres swojej odpowiedzialności; takie rozwiązanie zapobiega wykonaniu tego samego zadania przez kilka osób. Niektóre firmy, a można nawet powiedzieć, że wiele firm, niezależnie od wykorzystywanej metodyki używa Microsoft Excela do zarządzania projektami (przyzwyczajenia?). Umożliwia on przedstawienie danych w postaci tabeli oraz różnych grafów. Każda firma może zarządzać nim na swój własny sposób, dzięki czemu proces ten jest bardzo elastyczny. Ponadto ułatwia on wykonanie prognoz przyszłych dochodów, kalkulacji wybranych parametrów i wskaźników. Kolejnym narzędziem wykorzystywanym na szeroką skalę jest SharePoint, który umożliwia współdzielenie plików czy tworzenie listy zadań, która może być wykorzystana w MS Project.

Przejdźmy wreszcie do narzędzi stosowanych w metodach zwinnych. Bardzo często

[1] Backlog – rejestr sprintu / lista zadań[6]

[2] Sprint – jeden z etapów niektórych metodyk zwinnych, który wyznacza rytm pracy. W jego trakcie następuje faktyczne wykonanie określonej funkcjonalności[8]


Rys. 2.2. Team Estimation Game[www.scrum-master.pl]

w projektach zwinnych planujemy bez użycia dni i godzin. W klasycznych projektach tworzymy harmonogramy, szczegółowe plany, gdzie mamy daty, gdzie mamy godziny, gdzie każde zadanie ma określony czas realizacji. W przypadku projektów zwinnych często odchodzimy od tak szczegółowego planowania, ale jakaś forma planowania oczywiście jest potrzebna, dlatego stosujemy różne alternatywne metody i jedną z takich alternatywnych metod jest metoda nazwana Team Estimation Game.

#### 2.2.1. Team Estimation Game

Gra Team Estimation działa tak:

— Członkowie zespołu kolejno pobierają kartę ze stosu i przyklejają ją do ściany. Każda karta przedstawia jedną historyjkę. Po umieszczeniu pierwszej karty w środku, każdy członek zespołu umieszcza swoją kartę po prawej, jeśli jest trudniejsza, po lewej, jeśli jest mniej trudna, lub pod inną kartą, jeśli historyjki są mniej więcej takie same. rys.2.2
— Użytkownik może użyć swojej kolejki, aby przesunąć kartę już na ścianie po prawej lub lewej stronie.
— Proces może być kontynuowany po tym, jak stos kart zostanie zużyty, aż do ogólnego konsensusu w rankingu kart. Gracze mogą omawiać historie i wpływać na swoje decyzje.
— Następnie każdy członek drużyny odbiera kartę z numerem i umieszcza ją na jednej z kolumn. Członek zespołu może użyć swojej kolejki, aby zmienić przypisanie numeru wykonane przez innego członka zespołu. Trwa to aż do osiągnięcia konsensusu.

Zbiór liczb zawiera tylko te w sekwencji Fibonacciego, aby odzwierciedlić ogólną zasadę, że ryzyko wzrasta geometrycznie proporcjonalnie do złożoności.

#### 2.2.2. Kanban

Inną z takich alternatywnych metod jest metoda Kanban. Kanban, to koncepcja wzięta z produkcji, która została świetnie zaadaptowana właśnie do zarządzania projektami zwinnymi. Kanban w przypadku projektów jest po prostu tablicą, na której pokazujemy, rejestrujemy to, co się dzieje w projekcie: tabela 2.1

Zaczynając od lewej strony tej tablicy, mamy backlog produktu, a więc wszystkie nasze epiki, wszystkie nasze historyjki, które określają to, co jest w projekcie do realizacji.

Tabela 2.1. Tablica w metodzie Kanban.

| **Backlog** | **Breakdown(10*)** | | **Develop(15*)** | | | **Validate(10*)** | |
|---|---|---|---|---|---|---|---|
| Epics,User Stories | Working | Done | Working | Track | Done | Working | Done |

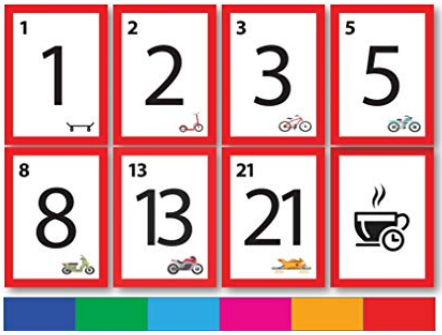
Następnie mamy trzy główne etapy realizacji projektu, w których nasze historyjki będą się za chwilę znajdowały:

— Pierwszym takim etapem jest breakdown albo analysis, czyli analizowanie user stories, rozbijanie ich na mniejsze części, na zadania, analiza tego, co jest do zrobienia, w jaki sposób mamy te funkcjonalności zrealizować. W tym miejscu musimy także zastanowić się nad kompatybilnością i powiązaniami pomiędzy poszczególnymi funkcjonalnościami, a także nad priorytetami, a więc bierzemy przede wszystkim te rzeczy, które mają wysoki priorytet oraz wszystkie funkcje, które są z nimi powiązane, abyśmy mogli dostarczyć nowy produkt.
— Kolejny etap, to jest rozwój, development- w sytuacjach, gdzie opracowujemy nasz projekt, a więc tutaj zespól developerski tworzy faktycznie te rozwiązania, które miały być opracowane.
— Trzeci etap, to validate, czyli kontrola, testy, ocena tego, co zostało zrobione. Sprawdzamy, czy rzeczywiście to, co miało być osiągnięte, zostało osiągnięte. Jeżeli tak, no to możemy dołączyć te funkcjonalności do produktu i przekazać je do klienta.

W każdym z tych etapów widzimy co najmniej dwie kolumny:

— jedna kolumna, to working, czyli pracujemy nad czymś,
— druga kolumna, to done, czyli takie pole odkładcze- wykonaliśmy to zadanie i dalej zadanie może przejść do kolejnego etapu, do kolejnej części prac.
— Dodatkowo, w przypadku kolumny związanej z rozwojem, mamy taką kolumnę track – to sytuacja, w której opracowaliśmy jakąś funkcjonalność, ale nie jesteśmy jej w stanie wdrożyć dalej, ponieważ czekamy na opracowanie innych, powiązanych z nią funkcjonalności. Czasem jest tak, że pewna funkcjonalność może być już gotowa, ale nie może działać, dopóki inne elementy nie zostaną przygotowane. Dlatego jest takie pole oczekiwania na inne elementy, które muszą być przygotowane.

Widzimy więc, że w tej metodyce nie mamy iteracji jako takich, mamy stały przepływ historyjek, stały przepływ pracy, która jest realizowana w naszym projekcie. Ale ktoś może zapytać, w jaki sposób, w takim razie określamy poziom wykonania prac, w jaki sposób określamy, ile tej pracy możemy mieć jednocześnie rozpoczętej? Temu służą te liczby w nawiasach, które widzimy w nagłówku tabeli. Te liczby określają nam liczbę punktów, które mogą być aktualnie w obróbce. Historyjki oceniamy nie względem czasu, lecz względem punktów trudności, punktów zaangażowania, które jest niezbędne, aby daną funkcję zrealizować. Jeżeli zatem, w danym etapie naszej tablicy Kanban mamy dopuszczone 10 punktów, to znaczy, że możemy tam mieć w trakcie obróbki historyjki, których




Rys. 2.3. Przykładowe karty do Planning Pokera[www.amazon.com].

wartość punktowa nie przekracza 10-ciu punktów. Tak samo w kolejnych etapach. Zatem to określa nam poziom pracy, która jest wykonywana i powoduje, że praca będzie przebiegać płynnie, nie będzie korków, nie będzie zatrzymań i będziemy w stanie w odpowiednim czasie dostarczać klientowi nowe produkty.

#### 2.2.3. Planning Poker

Jeszcze inną z takich alternatywnych metod (bardzo popularną) jest metoda, której poświęcona jest niniejsza praca. Ta metoda nazywa się Planning Poker i pozwala nam oszacować trudność, czasochłonność poszczególnych zadań, które są do wykonania w ramach projektu. Ta metoda ma też tą zaletę, że integruje zespół i pozwala uwzględnić różne punkty widzenia, a także przyczynia się do lepszego zrozumienia zadań, które są do wykonania. O co w tej grze chodzi? Bazujemy tutaj z grubsza na liczbach z ciągu Fibonacciego. Rys.2.3

Określamy trudność poszczególnych zadań wykorzystując kolejne liczby z tego ciągu: 1, 2, 3, 5, 8,13, 21…itd. Jeżeli jakieś zadanie oceniliśmy na 5, to znaczy, że jest ono trudniejsze od zadania, które oceniliśmy na 3, ale ile ono zajmie? Tego nie wiemy, to nas chwilowo nie interesuje. Staramy się określić przede wszystkim, na ile trudne i na ile pracochłonne, naszym zdaniem, mogą być zadania. Praktyka pokazuje, że planując sprinty, wcale nie musimy mieć szczegółowych informacji na temat czasu, te informacje zresztą często się nie sprawdzają, Taka forma punktowa zupełnie nam wystarcza. Kiedy wiemy, ile jesteśmy w stanie w ciągu sprintu zrealizować takich punktów, to wówczas jesteśmy w stanie stwierdzić, ile możemy zadań w danym sprincie upchnąć.

Jak to wygląda technicznie?

— Zespół deweloperski spotyka się na spotkaniu, siada przy stole, każdy otrzymuje swoje karty i mamy pakiet zadań (czy pakiet historyjek) z backlogu, którymi się musimy zająć. Zazwyczaj spotkanie prowadzi scrum master, przedstawia on historyjkę, którą mamy się zająć, określa czego ona dotyczy.
— Następnie każdy z członków zespołu wyciąga kartę, która określa trudność, jego zadaniem, realizacji tego zadania i kładzie ją na stole tak, aby nie było widać tej liczby.

— Kiedy wszyscy są gotowi, odwracamy karty i patrzymy, czy wynik, który uzyskaliśmy jest zgodny, czy też nie. Jeżeli wszyscy pokazali praktycznie tą samą liczbę, to sprawa jest załatwiona, to znaczy, że wszyscy rozumiemy zadanie w taki sam sposób, tak samo oceniamy jego trudność i możemy przejść do kolejnego.
— Kiedy jednak występuje zróżnicowanie, konieczna jest dyskusja. Pytamy osoby, która podała najniższą liczbę, dlaczego tak uważa. Pytamy osoby, która podała najwyższą liczbę, dlaczego tak uważa. Bardzo często te różnice wynikają z tego, że nie do końca rozumiemy zadanie.

A więc Planning Poker pozwala nam lepiej zrozumieć zadania w zespole, dzięki czemu nie ma później nieporozumień i dzięki czemu to planowanie jest bardziej precyzyjne. Kiedy przejdziemy przez wszystkie historyjki i zwymiarujemy, oszacujemy ich trudność w punktach, możemy przejść do planowania sprintu. Z doświadczenia wiemy, że nasz zespół jest w stanie zrealizować określoną liczbę punktów w ciągu danego sprintu. Ta liczba nazywa się team velocity, czyli szybkość, z jaką zespół pracuje. Jest typowa dla każdego zespołu, a więc nie jest tak, że jeżeli jeden zespół jest w stanie zrealizować 30 punktów, to każdy inny będzie w stanie tyle samo zrealizować. Co więcej, nie można tej liczby traktować jako wskaźnika motywacji (zróbcie pięć punktów więcej, to dostaniecie premię) dlatego, że zaobserwowano, iż w takich przypadkach każde kolejne szacowanie po prostu powoduje, że członkowie zespołu przydzielają więcej punktów, czyli określają zadania jako trudniejsze, niż są w rzeczywistości, a więc mamy taką inflację punktów-nie ma realnego zwiększenia realnego wykonanej pracy, jest po prostu zwiększenie liczby punktów. Taki sposób motywacji nie jest zatem dobrym pomysłem. Dzięki tej metodzie możemy odejść od określania, czy coś będzie trwało 2,5 godziny, czy 3,5 godziny, możemy skupić się na lepszym zrozumieniu zadań, jak również na tym, aby lepiej rozplanować pracę.

# 3. ROZPROSZONE ZESPOŁY PROJEKTOWE.

Według niemieckiego pisma Focus Money Magazin, ok 30 % wszystkich pracowników na świecie (a podobno w USA ponad 43 %) pracuje w tzw. zespołach rozproszonych lub wirtualnych. Wiele firm wprowadza zarządzanie na odległość, aby w ten sposób stymulować rozwój, rozwijać projekty międzynarodowe, czy chociażby ze względu na oszczędności lub optymalizację procesów.

Pojawienie się zjawiska zespołów rozproszonych może się wiązać z efektem globalizacji obszaru działania firm, wkraczaniem na nowe rynki, powstawaniem startupów, jak również działaniem tzw. freelancerów, którzy oferują wyspecjalizowane usługi. Powodem może być również wzrost kosztów pracy, zmiany na rynku pracy oraz brak na rynku lokalnym wykwalifikowanych pracowników, jak i również zmiany w kulturze pracy (np. pokolenie tzw. Milenialsów). Wszystko to powoduje zmiany w organizacji pracy.

## 3.1. ZALETY ZARZĄDZANIA PROJEKTAMI W ZESPOŁACH ROZPROSZONYCH.

Realizacja projektów zespołami rozproszonymi:

— pozwala na pozyskiwanie profesjonalistów w ramach całej organizacji lub poza nią;
— niweluje problem ograniczeń korzystania wyłącznie z zasobów jednego biura, czy w ramach jednego kraju;
— daje szerokie możliwości czerpania z dużo większych zbiorów doświadczeń oraz kreatywności;
— pozwala na elastyczność w rozbudowywaniu zespołu adekwatnie do pojawiających się nowych zadań;
— daje organizacjom szansę dostosowywania się do zmian pokoleniowych, gdzie często młode osoby wyrażają potrzebę wykonywania pracy w domu (Home Office);
— zwiększa efektywność kosztową.

## 3.2. DODATKOWE WYZWANIA DLA EFEKTYWNEGO FUNKCJONOWANIA ZESPOŁÓW ROZPROSZONYCH.

Każdy zespół projektowy napotyka w swojej pracy na najróżniejsze wyzwania, które związane są ze współpracą, komunikacją, czasami niejasnym podziałem obowiązków, brakiem odpowiedzialności, czy zaangażowaniem. Jednak zespoły rozproszone stają często przed dodatkowymi wyzwaniami, jakie wynikają z ich specyfiki, czyli pracy w różnych lokalizacjach.

Oto kilka najczęstszych wyzwań takich zespołów:

— **Zarzadzanie zespołem i synchronizacja pracy jego członków.** To wyzwanie nie tylko dotyczy zespołów rozproszonych, ale w właśnie w takich zespołach ważna jest umiejętność prowadzenia projektu i kwalifikacje związane z zarządzaniem ludźmi. W zespole rozproszonym powinny funkcjonować jasne zasady współpracy i komunikacji. Bardzo istotne jest ogólne zaangażowanie, wzajemne rozliczanie się z zadań, odpowiedzialność za rezultaty, otwartość, szacunek dla innych, zakładanie zawsze dobrych intencji, jasny podział obowiązków. Ważne jest również dla budowania zespołu i jego efektywności, aby zaplanować (w budżecie oraz czasowo) okresowych, wspólnych spotkań wszystkich członków.
— **Przepływ informacji.** Czasem kluczowe informacje nie docierają do wszystkich członków zespołu. Odpowiedzią na to może być ustalenie pewnego „rytmu" komunikacyjnego i regularne spotkania projektowe. Muszą to jednak być spotkania z sensem, tak zaplanowane, aby nie marnować czasu pracowników, aby rozwiązywały konkretne problemy i wspierały realizację celów projektu. Sposobów komunikacji jest wiele, można zatem je dobrze zaplanować i stosować w zależności od złożoności zespołu, celu i rodzaju projektu.
— **Komunikacja w obcym języku.** Coraz powszechniejsza jest konieczność porozumiewania się w języku obcym, ponieważ wiele firm posiada zagraniczne filie i siedziby. Zazwyczaj wykorzystywany do komunikacji jest język angielski, ale poziom jego znajomości może być różny i może powodować różne zabawne sytuacje lub nawet nieporozumienia. Tutaj może pomóc np. nauka prostych technik coachingowych- pytań doprecyzowujących i tzw. uważne słuchanie.

Jednak zespoły rozproszone muszą się borykać z problemami takimi jak: Różnica czasu, Różnice kulturowe, Odpowiednia rekrutacja, Motywowanie i wsparcie zespołu, Efektywność i produktywność, zwiększa efektywność kosztową

Dochodzimy wreszcie do kwestii, które próbuję rozwiązać, realizując projekt, który jest zasadniczym tematem niniejszej pracy inżynierskiej. Jest to wyzwanie:

— **Wsparcie techniczne i merytoryczne.** W zespołach rozproszonych wręcz niezbędne jest zadbanie o wsparcie techniczne. Co więcej, wielu menedżerów zarządzających zespołami rozproszonymi, wskazuje na potrzebę wsparcia merytorycznego , szczególnie w zakresie moderowania tzw. metodyk zwinnych (SCRUM). Świetne połączenie internetowe, sprzęt , sale i programy do tele i wideokonferencji, to powinien być standard w organizacjach, które pracują w zespołach wirualnych.

Więcej informacji o zespołach rozproszonych można znaleźć w:[12]

# 4. IMPLEMENTACJA PROJEKTU

Projekt aplikacji wspomagającej zdalne szacowanie historyjek metodą planning pokera został wykonany w technologii webowej o nazwie ReactJS oraz Firebase po stronie backendu. ReactJS to biblioteka javascript stworzona przez firmę Facebook oraz Instagram dzięki której budowanie dużych oraz kompleksowych interfejsów użytkownika jest łatwiejsze. Jest ona przeznaczona do wykorzystania z innym frameworkiem, który wprowadza backend. Podczas gdy Angular, Ember oraz Backpone są popularnymi wyborami do tego, Firebase wprowadza najłatwiejszą i najszybszą integrację z trwałym backendem czasu rzeczywistego do aplikacji napisanej w ReactJS – zajmuje to tylko kilka linijek w Javascript.

## 4.1. CZYM JEST REACTJS

Twórcy ReactJS opisują go jako Widok w architekturze MVC. Nie ma na celu zastąpienie Angular'a oraz Ember'a; zamiast tego rozszerza ich funkcjonalność przez wprowadzenie bardzo wydajnej drogi utrzymania widoku zsynchronizowanego z javascript. Ten specjalny sos który renderuje HTML używa wyjątkowo szybkiego algorytmu wirtualnego drzewa DOM, wprowadzający o wiele lepszą wydajność niż konkurujące platformy. Ma jednokierunkowy reaktywny przepływ danych, który jest znacznie bardziej zrozumiały niż tradycyjne przepływy danych. Komponenty – podstawowe bloki aplikacji React-owych – są zorganizowane w drzewie hierarchicznym, w którym komponenty rodzice wysyłają dane do swoich dzieci przez zmienne właściwości. Każdy komponent ma także zmienną stanu, która determinuje obecne dane dla tego widoku. Za każdym razem, gdy stan jest zmieniany, komponent wywołuje metodę render() a React znajduje najbardziej efektywną metodę aktualizacji drzewa DOM.

Odkąd głównym zadaniem React'a jest interfejs użytkownika, aplikacje na nim zrobione potrzebują czegoś jeszcze, co będzie zachowywało się jak backend. To jest miejsce gdzie Firebase wkracza. Dodaje Model i Kontroler w MVC do aplikacji napisanych w ReactJS, czyniąc z nich w pełni funkcjonujące aplikacje. Używając React'owego jednokierunkowego systemu wiązania danych łatwo jest zintegrować go z Firebase.[14]

## 4.2. FIREBASE

Firebase jest platformą dla aplikacji webowych oraz mobilnych która wprowadza dla deweloperów mnóstwo narzędzi oraz usług by pomóc im tworzyć wysokiej jakości aplikacje oraz zwiększyć ich bazę użytkowników.


Rys. 4.1. Usługi Firebase.[hackernoon.com]

### 4.2.1. Historia Firebase

W 2011 roku, zanim Firebase było Firebase'em był startup nazwany Envelope. Jako Envelope wprowadził dla deweloperów API, które umożliwiało wprowadzenie czatu do ich strony internetowej.

Interesującym było to, że ludzie używali aplikacji by przekazywać dane, które były czymś więcej niż wiadomościami chatowymi. Deweloperzy używali Envelope by synchronizować dane aplikacji jak stan gry w czasie rzeczywistym pomiędzy ich użytkownikami.

To poprowadziło założycieli Envelope, Jamesa Tamplina oraz Andrew Lee do pomysłu by rozdzielić chat oraz architekturę czasu rzeczywistego. W kwietniu 2012 Firebase został stworzony jako oddzielna firma które wprowadziła Backend jako usługę z funkcjonalnościami czasu rzeczywistego.

Po tym jak firma została przejęta przez Google w 2014 roku szybko ewoluowała do wielofunkcyjnej platformy mobilnej oraz webowej jaką znamy dzisiaj. rys.4.1

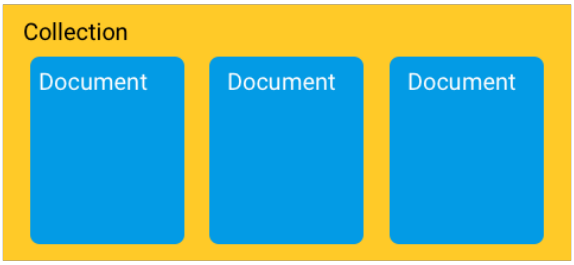
## 4.3. WYKORZYSTANE USŁUGI FIREBASE

### 4.3.1. Firebase Authentication

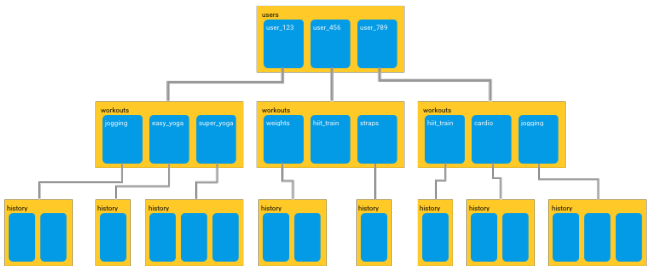
Przede wszystkim w swojej aplikacji skorzystałem z usługi Fierbase Authentication, która wprowadza serwerowe usługi, łatwe narzędzia dla deweloperów oraz gotowe biblioteki interfejsu by wprowadzić usługę logowania i rejestrowania do naszej aplikacji.

Normalnie zajęłoby miesiące by ustawić system autoryzacji na własną rękę. A nawet wtedy twórcy musieliby mieć dedykowany zespół by utrzymywać ten system. Ale jeżeli wykorzystają Firebase'a, mogą ustawić cały system w mniej niż 10 linijek kodu, które zajmą się wszystkim włącznie z kompleksowymi operacjami jak scalanie kont.

W swoim projekcie skorzystałem z usług logowania anonimowego oraz przez Github'a co było kluczowe w moim projekcie.




Rys. 4.2. Postać danych w firestore [hackernoon.com]



Rys. 4.3. Ułożenie danych w firestore [hackernoon.com]

### 4.3.2. Firestore

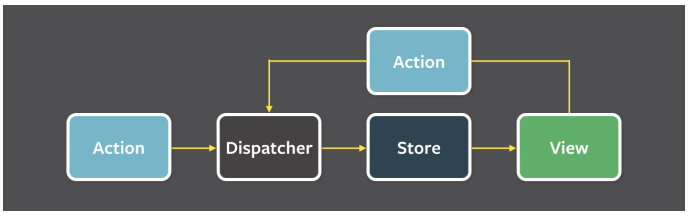
Firestore jest nie relacyjną bazą dokumentów, która pozwala łatwo przechowywać, synchronizować oraz przeszukiwać dane dla aplikacji mobilnych oraz webowych – w globalnej skali.

Firestore przechowuje dane w postaci obiektów zwanych dokumentami. Te dokumenty posiadają pary klucz-wartosć oraz mogą zawierać jakiekolwiek rodzaje danych od łańcuchów po dane binarne a nawet obiekty, które przypominają drzewa JSON. Dokumenty są pogrupowane w kolekcje. rys.4.2

Fierstore może zawierać wiele kolekcji zawierających dokumenty, które wskazują na subkolekcje. Te subkolekcje mogą znowu zawierać dokumenty oraz sub kolekcje i tak dalej. Można zatem powiedzieć że w tym modelu danych wszystko jest ułożone chierarchicznie.[4] rys.4.3

## 4.4. REDUX CZYLI IMPLEMENTACJA ARCHITEKTURY FLUX.

Jedną z najważniejszych cech komponentów ReactJS jest wbudowany w nie stan. Jest to bardzo przydatny koncept. Komponent posiada stan, który może ulec zmianie w wyniku interakcji użytkownika z aplikacją. Zmiana stanu pociąga za sobą operację re-renderowania drzewa Virtual DOM. W wyniku tego, pewne części interfejsu widocznego na ekranie ulegają zmianie. Oczywiście wiadomym jest też, że jeden komponent może zależeć od




Rys. 4.4. architektura flux [wwww.nafrontendzie.pl]

innego komponentu. Możemy przecież przekazywać stan komponentu rodzica do jego komponentów dzieci itd. To wszystko działa świetnie. Niestety w miarę jak aplikacja rośnie, rozrasta się poziom skomplikowania poszczególnych komponentów. Z tego względu programiści Facebooka odpowiedzialni za rozwój ReactJS wymyślili architekturę aplikacji, która rozwiązuje ten problem. Architektura ta nazwa się Flux.

### 4.4.1. Architektura FLUX

Flux jest architekturą aplikacji którą Facebook używa do budowania aplikacji po stronie klienta. Jest to uzupełnienie komponentów React'a przez wykorzystanie jednokierunkowego przepływowi danych. To jest raczej pewien wzorzec niż framework i można z niego korzystać bez wielu nowych linijek kodu. rys. 4.4

Przepływ rozpoczyna się od lewej strony. Najpierw tworzone jest akcja – jest to zwykły obiekt zawierający właściwość type. Oprócz tego może on posiadać więcej właściwości służących do przekazywania dodatkowych danych. Akcja taka tworzona jest przez funkcję zwaną action creator czyli kreator akcji. W przypadku redux'a jednak akcja jest zwykłą funkcją.

```
const mapStateToProps = (state) => {
return { counter: state.counter };
};
const mapDispatchToProps = (dispatch) => {
return {
onIncrement: () => dispatch({ type: 'INCREMENT' }),
onDecrement: () => dispatch({ type: 'DECREMENT' })
}
};

Counter = connect(mapStateToProps, mapDispatchToProps)(Counter);
```

Listing 4.1: Przykładowe akcje licznika i ich stan

W tym przykładzie przedstawiony jest przykład prostego licznika z dwoma akcjami, które albo zwiększają albo zmniejszają licznik o jeden. Stan oraz funkcję rozsyłające dostępne są w obiekcie this.props. Spójrzmy na kod odpowiedzialny za ich mapowanie.

### Funkcja mapStateToProps

Funkcja *mapStateToProps* pobiera *state* jako parametr i zwraca nowy obiekt. Częstą praktyką jest po prostu przekazanie całego stanu do "propsów", jednak jest to też właściwe miejsce by odfiltrować dane.

### Funkcja mapDispatchToProps

Kolejna funkcja to **mapDispatchToProps**. Zwraca ona obiekt zawierający metody. Za pomocą wywołania funkcji **dispatch** rozgłasza ona obiekty akcji do **store**. W powyższym przykładzie mamy pewne uproszczenie: do metody dispatch przekazywane obiekty akcji przekazywane są bezpośrednio. Zwykle w projekcie definiuje się to jako specjalne kreatory akcji. Dzięki kreatorom możemy opóźnić wywołanie akcji, lub zmienić dane w naszym redux'ie tylko wtedy, gdy są spełnione określone warunki.

```
export const startAddUserToGame = (owner, repo, game, user = {}) =>
↪ {
return dispatch => {
var userUpdate = {}
const tempUser = {
email: user.email,
isAnonymous: user.isAnonymous,
id: user.uid,
name: user.displayName,
online: true
};
userUpdate["users." + user.uid.toString()] = tempUser
const gameRef = db
.collection("users")
.doc(owner.toString())
.collection("repos")
.doc(repo.toString())
.collection("games")
.doc(game.toString())

gameRef.update(userUpdate).then(() => {
dispatch(addUserToGame(owner, repo, game, tempUser))
});
}
}
```

Listing 4.2: Przykładowy kreator akcji z projektu

W tym przykładzie gracz jest dodawany do gry, tylko wtedy, kiedy jest dodany do gry w bazie danych firestore. Jak widać w przykładzie kreator zwraca funkcję, która

wykorzystuje dyspatch by wysłać akcję do store. Aby zadziałać funkcja musi znać dokładną lokalizację gry w bazie oraz dane użytkownika. Aby dostać się do gry, musimy znać właściciela oraz nazwę repozytorium w której znajduje się gra, a później identyfikator gry. Później dodajemy gracza do naszej gry dzięki funkcji update dostając się odpowiednio do kolekcji użytkownicy, dokumentu użytkownika, kolekcji repozytoria, dokumentu naszego repozytorium. Później z repozytoriów wchodzimy do kolekcji gier, a dzięki znajomości identyfikatora gry dostajemy się do obiektu gry i wstawiamy do obiektu gry obiekt gracza. Jeżeli akcja w bazie danych zakończy się sukcesem rozsyłamy akcję dodawania gracza do store.[2]

### Funkcja connect

Ale wróćmy do tematu mapowania. W przedstawionym wcześniej przykładzie najbardziej istotna jest ostatnia jego linia. Jak widzisz, wywołuję w niej funkcję connect. Przyjmuje ona funkcje mapStateToProps oraz mapDispatchToProps jako parametry i wyniki ich wywołania łączy w odpowiedni obiekt. Następnie zwraca ona funkcję, która jako parametr przyjmuje komponent. Funkcja ta wstrzykuje przygotowany wcześniej obiekt do this.props tego komponentu.

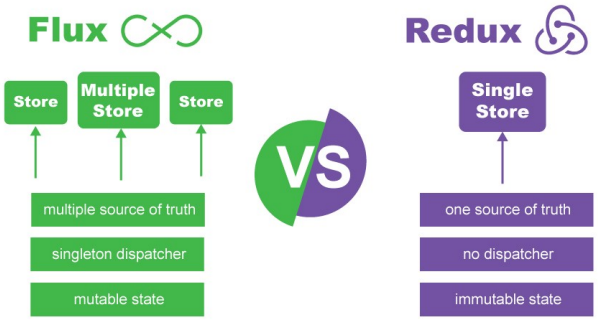
Zwróć też uwagę, że ostateczny wynik opisanych działań jest przypisywany ponownie do obiektu komponentu. To dlatego, że funkcja zwracana przez funkcję connect opakowuje przekazany komponent i zwraca nową jego wersję.

Ale wróćmy do FLUX. Tak tworzona akcja jest dostarczana do store za pomocą wywołania funkcji zwaną dispatcher. Funkcja ta w zasadzie zarządza całym przepływem danych. Każdy store w aplikacji rejestruje w dispatcherze swoje funkcje wywołania zwrotnego w celu obsługi przychodzących akcji. W momencie gdy akcja jest rozsyłana (ang. dispatch), wywoływane są po kolei wszystkie te callbacki. Jeden z nich powinien umieć rozpoznać akcję po jej typie i być przygotowany na jej odpowiednią obsługę.

Generalnie wszystkie obiekty store zawierają łącznie cały stan aplikacji. Store zawiera implementację funkcji wywoływania zwrotnego, która jest rejestrowana w „dispatcherze”, i która obsługuje akcję związane z danym store. W redux'ie cały stan jest przechowywany w jednym obiekcie.

Kolejny element na diagramie to widok. Można powiedzieć, że jest on reprezentowany po prostu przez komponent ReactJS. Używa on stanu aplikacji zapisanego w obiekcie store, tak jakby był on wewnętrznym stanem komponentu. Zmiana stanu w store powoduje re-renderownie komponentu. Dodatkowo komponent widoku może rozsyłać kolejne akcje, na przykład kiedy użytkownik kliknie na jakiś guzik na ekranie. To powoduje zmianę stanu zapisanego w store. W redux'ie komponent ma dostęp do akcji i stanu dzięki wspomnianej przeze mnie wcześniej funckji connect.

Wszystko to po prostu zestaw zasad. To programista może zdecydować jak to dokładnie będzie zaimplementowane. Na szczęście nie jest on zdany sam na siebie ponieważ istnieje




Rys. 4.5. Różnice między Redux a Flux [www.medium.com]

kilka gotowych implementacji architektury Flux w postaci bibliotek.[2] Jedną z nich jest oczywiście redux.

### 4.5. CZYM REDUX RÓŻNI SIĘ OD FLUX

Redux jest inspirowany pewnymi ważnymi cechami architektury Flux. Tak jak Flux Redux przepisuje twój model danych do oddzielnej warstwy twojej aplikacji (“stores” in Flux, “reducers” in Redux). Tak jak we Flux wszystkie operacji na danych są opisywane w postaci akcji. W przeciwieństwie do Flux Redux nie ma konceptu rozsyłacza, ponieważ opiera się na czystych funkcjach zamiast emiterów akcji, a czyste funkcje są łatwe do tworzenia i nie potrzebują żadnej encji by zarządzać nimi. Jest to więc pewne odstępstwo od Flux’a.

Inną ważną różnicą od Flux’a jest to że Redux zakłada że dane są niemutowalne. Możesz używać używać statycznych obiektów oraz tablic dla swojego stanu, ale mutowanie ich wewnątrz reducerów jest silnie nie rekomendowana. Zawsze powinieneś zawsze zwracać nowy obiekt. Ogólnie Redux może być opisany przez trzy fundamentalne zasady: rys.4.5

— Pojedyncze źródło prawdy - stan całej aplikacji przetrzymywany jest w drzewie obiektów wewnątrz pojedynczego obiektu store.
— Stan jest tylko do odczytu - jedynym sposobem na zmianę stanu jest wywołanie akcji, która zwraca obiekt opisujący co powinno się stać.
— Zmiany wykonywane są w ramach czystych funkcji - aby określić jak drzewo stanu transformowane jest przez akcje musisz tworzyć “czyste reducery”.

## ZAKOŃCZENIE

W pracy udało mi się dużo zrobić. Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam congue neque id dolor.

Mnóstwo innych rzeczy da się poprawić i rozwinąć. Donec et nisl id sapien blandit mattis. Aenean dictum odio sit amet risus. Morbi purus. Nulla a est sit amet purus venenatis iaculis. Vivamus viverra purus vel magna. Donec in justo sed odio malesuada dapibus. Nunc ultrices aliquam nunc. Vivamus facilisis pellentesque velit. Nulla nunc velit, vulputate dapibus, vulputate id, mattis ac, justo. Nam mattis elit dapibus purus. Quisque enim risus, congue non, elementum ut, mattis quis, sem. Quisque elit.

# Dodatki

## A. TO POWINIEN BY? DODATEK

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

## BIBLIOGRAFIA


[1] 4pm, *Przyjrzyjmy się tradycyjnym projektom (porównanie prince2 oraz agilepm)*, `http://4pm.pl/artykuly/przyjrzyjmy-sie-tradycyjnym-projektom`. Ost. dost. 4 grudnia 2018.

[2] Bartek Dybowski, *Podstawy redux - zarządzanie stanem aplikacji reactjs*, `https://www.nafrontendzie.pl/podstawy-redux-zarzadzanie-stanem-react`. Ost. dost. 4 grudnia 2018.

[3] Cohen, M., *Agile Estimating And Planning*, Robert C. Martin Series (Pearson Educatin, Inc., Massachusetts, 2006).

[4] GeekyAnts, *Introduction to firebase*, `https://hackernoon.com/introduction-to-firebase-218a23186cd7`. Ost. dost. 4 grudnia 2018.

[5] Institute, P.M., *A Guide to the Project Management Body of Knowledge* (Project Management Institute, Inc., Newton Square, Pensylvania USA, 2000).

[6] Jabłoński, B., *Wykorzystanie metodyk zwinnych do poprawy wiedzy i umiejętności projektowych studentów kierunków technicznych*, Edukacja - Technika - Informatyka. 2016, tom 94.

[7] Kopczewski, M., *Alfabet zarządzania projektami. Zarządzanie projektem od A do Z*, 2 wyd. (Helion, Massachusetts, 2015).

[8] Ćwiklicki M., Jabłoński M., W.T., *Samoorganizacja w zarzadzaniu projektami metodą Scrum* (Milfes.pl, wydawnictwo@milfes.pl, 2010).

[9] Mariusz Sołtysik, M.W., *Współczesne trendy w zarządzaniu projektami* (Mfiles.pl, Kraków, 2016).

[10] Percival, H.J., *Test Driven Development in Python* (Helion, ul. Kościuszki 1c, 44-100 Gliwice, 2015).

[11] Portny, S.E., *Project Management For Dummies*, 3d edition wyd., For Dummies (Helion, ul. Kościuszki 1c, 44-100 Gliwice, 2013).

[12] projektgamma, *Zespoły rozproszone*, `https://www.projektgamma.pl/strefa-wiedzy/wiki/zespoly-rozproszone`. Ost. dost. 4 grudnia 2018.

[13] Shwaber, K., *Sprawne zarządzanie projektami metodą Scrum*, polish wyd. (APN PROMISE, Warszawa, 2005).

[14] Wenger, J., *Using firebase with reactjs*, `https://firebase.googleblog.com/2014/05/using-firebase-with-reactjs.html`. Ost. dost. 4 grudnia 2018.

[15] Wolf, H., *Zwinne projekty w klasycznej organizacji Scrum, Kanban, XP* (Helion, ul. Kościuszki 1c, 44-100 Gliwice, 2014).