

**OŚWIADCZENIE**

*„Wyrażam zgodę na udostępnianie mojej pracy przez Archiwum WAT”.*

Dnia ........................ .................................

(podpis)

*Pracę przyjąłem*

*promotor pracy*

*dr inż. Tomasz Protasowicki*

***TU WSTAWIĆ KARTKĘ ZADANIEM DO PRACY DYPLOMOWEJ OTRZYMANĄ Z INSTYTUTU/DZIEKANATU - nie przepisywać jej treści!!***

***TU WSTAWIĆ KARTKĘ ZADANIEM DO PRACY DYPLOMOWEJ OTRZYMANĄ Z INSTYTUTU/DZIEKANATU - nie przepisywać jej treści!!***

Spis treści

[Wstęp 2](#_Toc91668254)

[I. Problemy metodologiczne 2](#_Toc91668255)

[1.1. Wprowadzenie 2](#_Toc91668256)

[1.2. Zarys sytuacji problemowej 2](#_Toc91668257)

[1.2.1. Podstawy teoretyczne 2](#_Toc91668258)

[1.2.2. Terminologia 2](#_Toc91668259)

[1.3. Cel pracy 2](#_Toc91668260)

[1.4. Problemy inżynierskie 2](#_Toc91668261)

[1.5. Organizacja praktyki inżynierskiej 2](#_Toc91668262)

[1.5.1. Cykl życia systemu informatycznego 2](#_Toc91668263)

[1.5.2. Metodyka zarządzania procesem wytwórczym 2](#_Toc91668264)

[1.5.3. Metodyka wytwarzania systemu informatycznego 2](#_Toc91668265)

[1.5.4. Metody, narzędzia i techniki inżynierii systemów 2](#_Toc91668266)

[1.5.5. Metamodel zawartości repozytorium projektowego **Błąd! Nie zdefiniowano zakładki.**](#_Toc91668267)

[1.6. Ocena piśmiennictwa 2](#_Toc91668268)

[1.7. Wnioski 2](#_Toc91668269)

[II. aNALIZA BIZNESOWA 2](#_Toc91668270)

[2.1. Wprowadzenie 2](#_Toc91668271)

[2.2. Analiza organizacji 2](#_Toc91668272)

[2.2.1. Charakterystyka organizacji/przedsięwzięcia 2](#_Toc91668273)

[2.2.2. Dziedzina działalności organizacji 2](#_Toc91668274)

[2.2.3. Otoczenie organizacji 2](#_Toc91668275)

[2.3. Analiza interesariuszy 2](#_Toc91668276)

[2.4. Role biznesowe i aktorzy biznesowi 2](#_Toc91668277)

[2.4.1. Katalog ról biznesowych i odpowiedzialności 2](#_Toc91668278)

[2.4.2. Mapa aktorów biznesowych 2](#_Toc91668279)

[2.5. Obiekty / encje biznesowe 2](#_Toc91668280)

[2.5.1. Model konceptualny - diagramy klas poziom konceptualny 2](#_Toc91668281)

[2.5.2. Maszyny stanowe 2](#_Toc91668282)

[2.6. Procesy biznesowe 2](#_Toc91668283)

[2.6.1. Mapa procesów biznesowych 2](#_Toc91668284)

[2.6.2. Macierz CRUD dla relacji PROCES-OBIEKT/ENCJA 2](#_Toc91668285)

[2.6.3. Macierz RACI dla relacji ROLA BIZNESOWA-PROCES BIZNESOWY 2](#_Toc91668286)

[2.6.4. Proces - *obsługa pożyczki* 2](#_Toc91668287)

[2.6.5. Podproces – *Złożenie wniosku o pożyczkę* 2](#_Toc91668288)

[2.7. Reguły biznesowe 2](#_Toc91668289)

[2.8. Potrzeby biznesowe 2](#_Toc91668290)

[2.8.1. Wymagania biznesowe 2](#_Toc91668291)

[2.8.2. Wymagania interesariuszy 2](#_Toc91668292)

[2.9. Diagramy wymagań 2](#_Toc91668293)

[2.10. Wnioski 2](#_Toc91668294)

[III. ANALIZA SYSTEMU INFORMATYCZNEGO 2](#_Toc91668295)

[3.1. Wprowadzenie 2](#_Toc91668296)

[3.2. Wymagania funkcjonalne 2](#_Toc91668297)

[3.2.1. Obszary funkcjonalne systemu 2](#_Toc91668298)

[A. zarządzanie kontem 2](#_Toc91668299)

[B. obsługa pożyczki 2](#_Toc91668300)

[C. zarządzanie wnioskiem o pożyczkę 2](#_Toc91668301)

[3.2.2. Wymagania funkcjonalne w obszarze A 2](#_Toc91668302)

[3.2.3. Wymagania funkcjonalne w obszarze B 2](#_Toc91668303)

[3.2.4. Wymagania funkcjonalne w obszarze C 2](#_Toc91668304)

[3.3. Wymagania niefunkcjonalne 2](#_Toc91668305)

[3.3.1. Wydajność 2](#_Toc91668306)

[3.3.2. Rozmiar 2](#_Toc91668307)

[3.3.3. Niezawodność 2](#_Toc91668308)

[3.3.4. Bezpieczeństwo 2](#_Toc91668309)

[3.3.5. Łatwość użytkowania 2](#_Toc91668310)

[3.3.6. Inne 2](#_Toc91668311)

[3.4. Założenia i ograniczenia 2](#_Toc91668312)

[3.4.1. Założenia 2](#_Toc91668313)

[3.4.2. Ograniczenia 2](#_Toc91668314)

[3.5. Aktorzy i role systemowe 2](#_Toc91668315)

[3.6. Przypadki użycia 2](#_Toc91668316)

[3.6.1. Przypadki użycia w obszarze funkcjonalnym A 2](#_Toc91668317)

[3.6.2. Przypadki użycia w obszarze funkcjonalnym B 2](#_Toc91668318)

[3.6.3. Przypadki użycia w obszarze funkcjonalnym C 2](#_Toc91668320)

[3.7. Reguły systemowe 2](#_Toc91668321)

[3.7.1. Diagram reguł decyzyjnych 2](#_Toc91668322)

[3.7.2. Tablice decyzyjne 2](#_Toc91668323)

[3.7.3. Drzewa decyzyjne 2](#_Toc91668324)

[3.7.4. Inne sposoby reprezentacji reguł **Błąd! Nie zdefiniowano zakładki.**](#_Toc91668326)

[3.8. Model dziedziny systemu 2](#_Toc91668327)

[3.8.1. Diagramy klas - poziom logiczny (atrybuty) 2](#_Toc91668328)

[3.8.2. Diagramy maszyny stanowej 2](#_Toc91668329)

[3.8.3. Diagramy sekwencji - poziom konceptualny 2](#_Toc91668330)

[3.9. Wnioski 2](#_Toc91668331)

[IV. PROJEKT systemu informatycznego 2](#_Toc91668332)

[4.1. Wprowadzenie 2](#_Toc91668333)

[4.2. Strategia projektowania systemu informatycznego 2](#_Toc91668334)

[4.2.1. Wariant minimalny 2](#_Toc91668335)

[4.2.2. Wariant maksymalny 2](#_Toc91668336)

[4.2.3. Wariant realny 2](#_Toc91668337)

[4.2.4. Wybór wariantu projektowania systemu informatycznego 2](#_Toc91668338)

[4.3. Architektura wysokopoziomowa systemu 2](#_Toc91668339)

[4.3.1. Decyzje architektoniczne 2](#_Toc91668340)

[4.3.2. Wzorce architektoniczne 2](#_Toc91668341)

[4.3.3. Podział warstwowy i układ warstw systemu 2](#_Toc91668342)

[4.3.4. Komponenty systemu 2](#_Toc91668343)

[4.3.5. Interfejsy 2](#_Toc91668344)

[4.4. Mechanizmy bezpieczeństwa 2](#_Toc91668345)

[4.4.1. Mechanizm bezpieczeństwa zabezpieczający transakcje 2](#_Toc91668346)

[4.4.2. Mechanizm bezpieczeństwa zabezpieczający logowanie 2](#_Toc91668347)

[4.5. Logiczny model danych 2](#_Toc91668348)

[4.5.1. Model bazy danych 2](#_Toc91668349)

[4.5.2. Model struktur danych //przechowywanych poza bazą danych 2](#_Toc91668350)

[4.6. Algorytmy 2](#_Toc91668351)

[4.6.1. Algorytm przebiegu transakcji 2](#_Toc91668352)

[4.6.2. Algorytm weryfikacji przyznania pożyczki 2](#_Toc91668353)

[4.6.3. Algorytm przyznawania punktów *rating* 2](#_Toc91668354)

[4.7. Wnioski 2](#_Toc91668355)

[V. IMPLEMENTaCJA I TESTY systemu informatycznego 2](#_Toc91668356)

[5.1. Wprowadzenie 2](#_Toc91668357)

[5.2. Charakterystyka środowiska wytwórczego 2](#_Toc91668358)

[5.2.1. Narzędzia / środowisko programistyczne 2](#_Toc91668359)

[5.2.2. Języki programowania 2](#_Toc91668360)

[5.2.3. Szkielet budowy aplikacji 2](#_Toc91668361)

[5.2.4. Biblioteki i komponenty 2](#_Toc91668362)

[5.2.5. Protokoły komunikacyjne// Oprogramowanie pośredniczące // Komunikacja w sieci 2](#_Toc91668363)

[5.2.6. Bazy danych 2](#_Toc91668364)

[5.2.7. Inne oprogramowanie 2](#_Toc91668365)

[5.3. Charakterystyka środowiska uruchomieniowego 2](#_Toc91668366)

[5.3.1. Konfiguracja sprzętowa 2](#_Toc91668367)

[5.3.2. Konteneryzacja //Warstwa wirtualizacji 2](#_Toc91668368)

[5.3.3. Systemy operacyjne 2](#_Toc91668369)

[5.3.4. Wymagane licencje 2](#_Toc91668370)

[5.3.5. Sposób monitorowania systemu 2](#_Toc91668371)

[5.4. Charakterystyka repozytorium kodu 2](#_Toc91668372)

[5.4.1. Struktura repozytorium 2](#_Toc91668373)

[5.4.2. Standardy tworzenia kodu źródłowego 2](#_Toc91668374)

[5.4.3. Standardy nazewnicze 2](#_Toc91668375)

[5.5. Dobre praktyki programistyczne **Błąd! Nie zdefiniowano zakładki.**](#_Toc91668376)

[5.5.1. Mechanizm **Błąd! Nie zdefiniowano zakładki.**](#_Toc91668377)

[5.6. Zastosowane wzorce projektowe 2](#_Toc91668378)

[5.6.1. Wzorce logiki dziedziny 2](#_Toc91668379)

[5.6.2. Wzorce obsługi źródeł danych 2](#_Toc91668380)

[5.6.3. Wzorce zachowań dla mapowania obiektowo-relacyjnego 2](#_Toc91668381)

[5.6.4. Wzorce struktury dla mapowania obiektowo-relacyjnego 2](#_Toc91668382)

[5.6.5. Wzorce odwzorowań obiektów i relacyjnych metadanych 2](#_Toc91668383)

[5.6.6. Wzorce warstwy prezentacji 2](#_Toc91668384)

[5.6.7. Wzorce dystrybucji 2](#_Toc91668385)

[5.6.8. Wzorce współbieżności autonomicznej 2](#_Toc91668386)

[5.6.9. Wzorce stanu sesji 2](#_Toc91668387)

[5.6.10. Wzorce podstawowe 2](#_Toc91668388)

[5.6.11. Wzorce obsługi błędów 2](#_Toc91668389)

[5.7. Testy systemu informatycznego 2](#_Toc91668390)

[5.7.1. Testy jednostkowe 2](#_Toc91668391)

[5.7.2. Testy integracyjne 2](#_Toc91668392)

[5.7.3. Testy wydajnościowe 2](#_Toc91668393)

[5.7.4. Testy akceptacyjne 2](#_Toc91668394)

[5.8. Wnioski 2](#_Toc91668395)

[VI. EWALUACJA systemu informatycznego 2](#_Toc91668396)

[6.1. Wprowadzenie 2](#_Toc91668397)

[6.2. Metoda oceny uzyskanych wyników 2](#_Toc91668398)

[6.3. Ocena uzyskanych wyników 2](#_Toc91668399)

[6.3.1. Ocena stopnia implementacji systemu 2](#_Toc91668400)

[6.3.2. Ocena stopnia pokrycia wymagań 2](#_Toc91668401)

[6.3.3. Ocena funkcjonalności systemu 2](#_Toc91668402)

[6.3.4. Ocena jakości systemu 2](#_Toc91668403)

[6.3.5. Ocena użyteczności systemu 2](#_Toc91668404)

[6.4. Kierunki rozwoju i doskonalenia 2](#_Toc91668405)

[6.4.1. Rekomendacje kierunku rozwoju systemu 2](#_Toc91668406)

[6.4.2. Rekomendacje w zakresie doskonalenia metod inżynierii systemów 2](#_Toc91668407)

[6.5. Wnioski 2](#_Toc91668408)

[Zakończenie 2](#_Toc91668409)

[BIBLIOGRAFIA 2](#_Toc91668410)

[Wykaz Rysnunków 2](#_Toc91668411)

[Wykaz Tabel 2](#_Toc91668412)

# Wstęp

W tym miejscu należy zawrzeć[[1]](#footnote-1):

1. ogólne wprowadzenie do problematyki pracy
2. zwięzłe wyjaśnienie **głównego motywu** wyboru tematu pracy
3. wskazanie **głównego celu** pracy
4. omówienie struktury pracy i zwięzła merytoryczna charakterystyka treści poszczególnych rozdziałów pracy
5. Wskazanie istotnych wyników pracy, podkreślenie nowatorstwa i/lub oryginalności ujęcia problemu - należy szczegółowo wyartykułować co dokładnie wg dyplomanta jest nowatorskie/oryginalne

Tu zilustrowano coś na poniższym rysunku/diagramie (rys. 1).

Rys. . Zabawa u optometrysty.

C:\Program Files (x86)\Microsoft Office\MEDIA\CAGCAT10\j0090070.wmf

Źródło: opracowanie własne na podstawie [1], s. 11 - 12.

A tu podano jakieś dziwne dane w poniższej tabeli (tab. 1).

Tab. . Taka sobie tabelka

|  |  |  |
| --- | --- | --- |
|  |  |  |

Źródło: [2], s. 11.

\* \* \*

Tu można napisać podziękowania jeśli ktoś bardzo tego potrzebuje. Jeżeli nie - to należy usunąć powyższe \*\*\*

# Problemy metodologiczne

## Wprowadzenie

System obsługi mikropożyczek przeznaczony jest zarówno dla osób potrzebujących wsparcia finansowego, jak i okazujących pomoc, jednakże wymagany jest status studenta. Dzięki temu platforma jest skupiona wokół społeczności akademickiej. Daje to większe szanse na udzielenie pożyczki pomiędzy studentami.

## Zarys sytuacji problemowej

Rozpoczęcie studiów w większości przypadków wiąże się z koniecznością wyprowadzki z domu rodzinnego. Wynajmując mieszkanie studenci są zmuszeni do dokonywania regularnych opłat. Do tego dochodzą wydatki związane z utrzymaniem np. na pożywienie, transport, lekarza czy inne niespodziewane wydatki. Często dochodzi do sytuacji, kiedy te wydatki przewyższają dochody i nie są one wystarczające na wyżej wspomniane obciążenia. Brak funduszy na podstawowe środki do życia wymusza konieczność zaciągnięcia pomocy finansowej.

Nie każdy student wiąże koniec z końcem, jednakże wydatki uniemożliwiają odłożenie większej ilości funduszy i zbudowanie odpowiedniego kapitału, który jest potrzebny, aby rozpocząć własną działalność, wprowadzić nowy produkt czy realizować inne plany. Jego brak wiąże się z tym, że przełomowe pomysły, wymagające większego wkładu finansowego nie mogą dojść do skutku.

Z drugiej strony istnieją również studenci, którym tych funduszy nie brakuje, a nawet bywa i tak, że mają odłożone większe kwoty, z którymi nic nie robią. Chcąc pomnożyć swój kapitał mają możliwość udzielenia pożyczki z ustalonymi uprzednio warunkami, dotyczącymi kwoty, oprocentowania oraz czasu, na jaki zostanie udzielona.

Problemem jest zapewnienie obsługi procesu pożyczki. Należy uwzględnić zabezpieczenie przebiegu procesu pożyczki oraz zwrotu. Ponadto szczególną kwestią do rozpatrzenia jest wiarygodność zarówno pożyczkodawcy jak i pożyczkobiorcy. Istnieje tu pewne ryzyko pożyczkowe, które powinno zostać ówcześnie ocenione. Pożyczkę można zaciągnąć na dowolny cel. Platforma ma dać szanse każdemu, komu potrzebna jest pomoc finansowa – niezależnie od powodu.

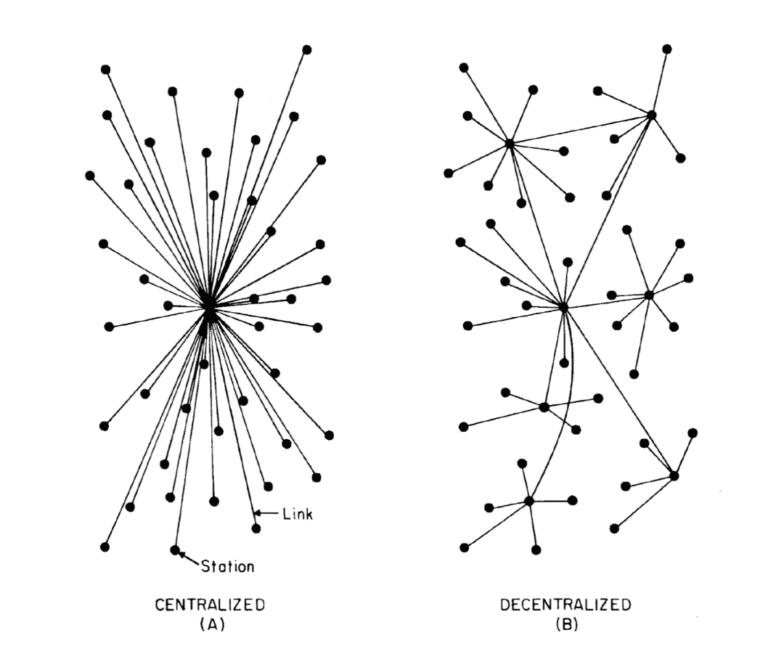
Zaletą tego rozwiązania jest pominięcie instytucji finansowych w zaciągnięciu pożyczki. Będąc studentem nie posiada się wystarczającej zdolności kredytowej, co powoduje, że banki niechętnie udzielają pomocy finansowej. Kolejnym pozytywnym aspektem korzystania z systemu jest możliwość zaciągnięcia pożyczek o bardzo niskich kwotach co jest niemożliwe w przypadku banku. Biorąc pod uwagę sytuację, kiedy potrzebujemy pożyczyć małą ilość pieniędzy, zwrócenie się do banku mija się z celem, ponieważ nie realizują one pożyczek przykładowo w kwocie 50 zł. Dodatkowo sam przebieg procedury jest nieskomplikowany i przejrzysty, eliminując zbędną papierową dokumentację.

### Podstawy teoretyczne

BLOCKCHAIN

Blockchain, inaczej łańcuch bloków jest technologią rozproszonego rejestru, która służy do przechowywania oraz przesyłania informacji o transakcjach zawartych w Internecie. Jest to innowacyjna koncepcja zdecentralizowanej bazy danych, która nie znajduję się na centralnym serwerze, lecz stanowi sieć równoważnych replik. Łańcuch bloków to współużytkowany, nieedytowalny rejestr, dzięki któremu proces rejestrowania transakcji oraz śledzenia zasobów ulega znacznemu uproszczeniu.

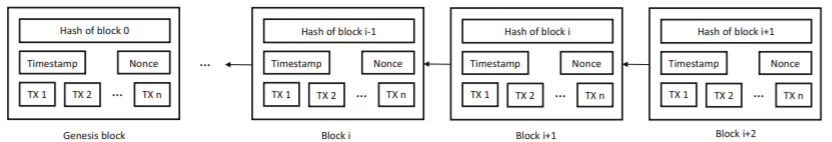
Rysunek 1. Różnica pomiędzy siecią a) zcentralizowaną b) decentralizowaną



Źródło: P. Baran, On Distributed Communication Networks, Rand Corporation, 1962, s. 4.

Informacje ułożone są w postaci następujących po sobie bloków danych. W przypadku gdy jeden z bloków wypełni się danymi, tworzony jest następny, który jest zintegrowany z poprzednim. Każdy z bloków zawiera oznaczenie czasu(*time-stamp*), kiedy został stworzony oraz niepowtarzalny identyfikator zwany *hash’em,* będący powiązaniem do poprzedniego bloku.W ten sposób tworzy się pewien rodzaj nierozerwalnego łańcucha bloków danych.

Rysunek 2. Architektura łańcucha bloków danych

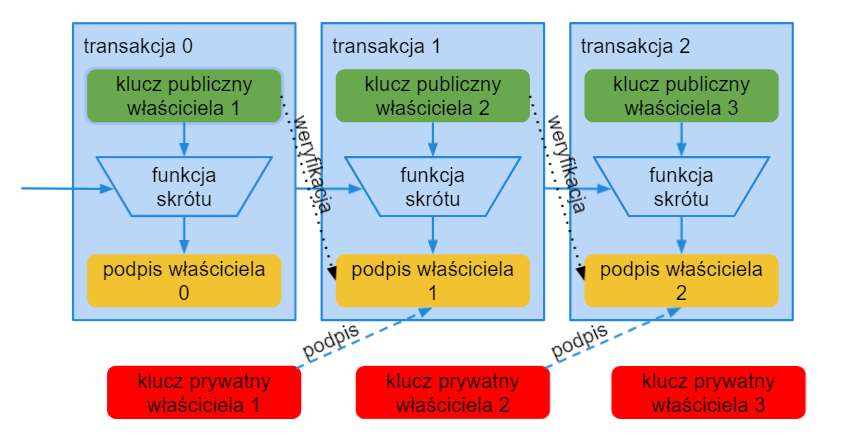


Źródło: Nofer, M., Gomber, P., Hinz, O., & Schiereck, D. (2017). Blockchain. *Business & Information Systems Engineering*, *59*(3), 183-187.

Do utworzenia identyfikatora wykorzystywana jest kryptograficzna funkcja skrótu, która generuje *hash*. Jest to krótki ciąg znaków przyporządkowany do dowolnie dużego zbioru danych. Wygenerowany *hash* zapewnia:

* Odporność na kolizję – niemożliwe jest znalezienie dwóch odrębnych danych wejściowych, które w wyniku zastosowania funkcji skrótu dadzą taki sam wynik (*hash*)
* Jednokierunkowość oraz nieodwracalność – nie da się odtworzyć danych wejściowych, znając jej skrót (*hash)*

Powyższe cechy zapewniają, iż transakcje zawarte w bloku są odporne przed manipulacją i fałszowaniu, co zapewnia bezpieczeństwo danych. Każda próba zmiany w obrębie jednego bloku wiąże się z zmianą kolejnego po nim bloku. W przypadku podjęcia próby nieautoryzowanej transakcji, węzły blockchain w procesie weryfikacji stwierdzają wystąpienie niezgodności w sieci i odmawiają zapisu w łańcuchu bloków. Transakcja, która została zapisana nie może zostać usunięta lub zmieniona. Zapobiega to możliwości ingerowania w dane przez osoby trzecie oraz tworzy wiarygodny rejestr transakcji dla uczestników sieci.

źródło: Krzysztof Szydłowski, „Pieniądz i bitcoin. Przeszłość i przyszłość”

Rysunek 3. Przebieg transakcji zawieranych w blockchain

Blockchain jest terminem nadrzędnym, pod którym kryje się całe grono różnego rodzaju platform. Różnią się one od siebie parametrami, możliwością zastosowania jak i przeznaczeniem. Podstawowym kryterium podziału sieci łańcucha bloków jest prawo dostępu do bazy danych, zatem wyróżniamy rodzaje blockchain:

* Publiczny – dostępny dla wszystkich użytkowników, pozwalający zarówno na odczyt zachodzących transakcji, jak i na zgłaszanie transakcji do zapisu w sieci. Sieć jest pseudo anonimowa, co oznacza, że nie znamy tożsamości drugiej osoby, jednakże posiadamy jej adres publiczny. Przykładem publicznej sieci jest Bitcoin lub Ethereum.
* Prywatny – dostępny jedynie dla zamkniętej grupy użytkowników, mających prawo do odczytu zachodzących transakcji jak i również zgłaszania transakcji do zapisu. Kontrola dostępu do prywatnej sieci uniemożliwia nieupoważnionym użytkownikom na interakcję z blockchainem. Bazuje ona głównie na zaawansowanych mechanizmach kryptografii asymetrycznej. Prywatny blockchain jest wykorzystywany w sytuacjach, kiedy sieć biznesowa zawiera poufne dane, które nie mogą zostać udostępnione publicznie lub gdy regulacje prawne nie zezwalają na korzystanie z publicznego blockchaina. Przykładem prywatnej sieci jest R3 Corda lub HyperLedger.

W celu lepszego zrozumienia zostaną zestawione cechy dwóch czołowych przedstawicieli platform obsługujących *smart contract* o dostępie publicznym oraz prywatnym - Ethereum oraz Hyperledger. Każde z rozwiązań cechuje się innymi właściwościami. Budując system należy dokonać odpowiedniego doboru technologii, kierując się ustalonymi założeniami.

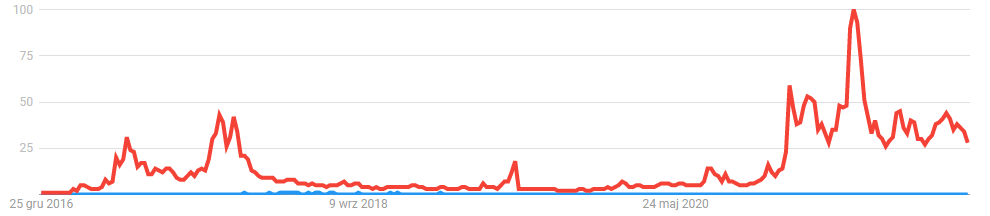
Tabela 1. Zestawienie różnic pomiędzy Ethereum a Hyperledger

|  |  |  |
| --- | --- | --- |
|  | Ethereum | Hyperledger |
| Typ portfela | Publiczny | Prywatny (z upoważnieniami) |
| Sieć | Otwarta, dostępna dla wszystkich użytkowników | Ograniczona, dostępna dla upoważnionych użytkowników |
| Organizacja | Ethereum, założona przez Vitalik Buterin | The Linux Foundation |
| Kryptowaluta | Ether | Brak |
| Transakcje | Widoczne dla każdego | Widoczne dla upoważnionych użytkowników |
| Opłaty za transakcje | W postaci tokenu Gas | Brak opłat |
| Język programowania *smart contract* | Soldity | Java / Golang / NodeJS |
| Wydajność | ~ 20 transakcji / s | ~ 2000 transakcji / s |

Źródło: opracowanie własne na podstawie: Martin Valenta, Philipp Sandner, „Comparison of Ethereum, Hyperledger Fabric and Corda”, str. 2

Poniższy wykres przedstawia popularność wyszukiwania haseł „*Ethereum”* oraz „*Hyperledger*” w ostatnich 5 latach. Liczby reprezentują poszczególne zainteresowania w wyszukiwaniu względem najwyższego punktu na wykresie. Wartość 100 oznacza najwyższą popularność hasła.

Rysunek 4. Wykres zainteresowania w ujęciu czasowym  
(czerwony – Ethereum, niebieski – Hyperledger)



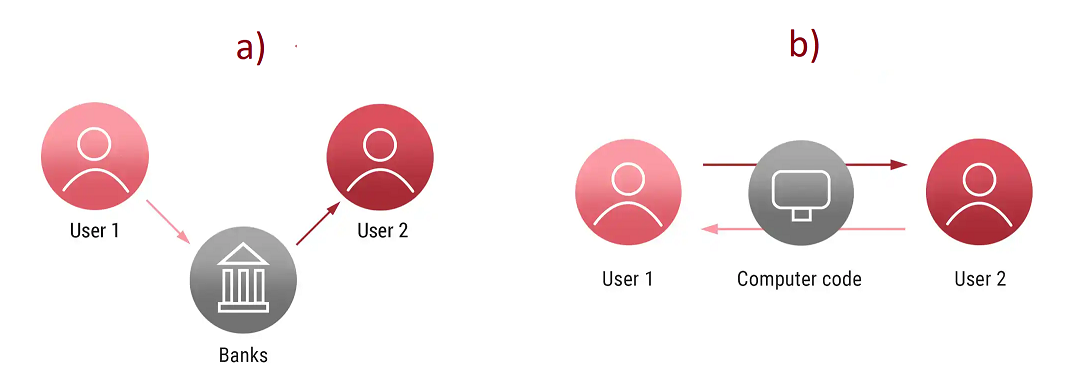
• Ethereum

• Hyperledger

Źródło: opracowanie własne na podstawie *Google Trends*

Dużo większym zainteresowaniem cieszy się publiczna platforma Ethereum, która swój szczyt popularności osiągnęła w maju 2021r. Fakt ten może wynikać z tego, iż Ethereum jest obiektem zainteresowań nie tylko w społeczności programistów ale w większej mierze inwestorów. Cena natywnego tokena (kryptowaluty) platformy Ethereum zwanego Etherem na początku jego wprowadzenia na giełdy (10.03.2016) wynosiła 46 zł, gdzie obecnie (03.09.2021r.) wynosi 15 625 zł, co daje nam procentowy wzrost wynoszący 33 867,39%.

Przełomowa technologia blockchain zapewnia wiele wydajnych rozwiązań. Jednym z nich są „inteligentne umowy” z ang. *smart contracts*. Z definicji wynika, iż *smart contract* jest to protokół komputerowy przeznaczony do cyfrowego ułatwienia, weryfikacji, egzekwowania negocjacji lub wykonania umowy. Umowy są egzekwowane poprzez określony zestaw reguł bez ingerencji człowieka. Rozwiązanie te jest szybsze oraz mniej narażone na manipulowanie oraz fałszowanie niż umowy tradycyjne. *Smart contract* jest to program deterministyczny. Pozwala na zawieranie transakcji bez konieczności wzajemnego zaufania pomiędzy dwoma stronami, ponieważ w przypadku nie spełnienia warunków, umowa nie zostanie zapisana. Wykorzystując inteligentne kontrakty jesteśmy w stanie przeprowadzać wiarygodne transakcje wykluczając konieczność pośredników i instytucji centralnych.



Rysunek 5. Różnice pomiędzy a) transakcja tradycyjną b) *smart contract*

Źródło: Anna Buczak, „Smart Contract For Fintech - Definition And Use Cases”

FINANSOWANIE SPOŁECZNOŚCIOWE – ang. *CROWDFUNDING*

Według definicji crowdfunding to „rodzaj gromadzenia i alokacji kapitału przekazywanego na rzecz rozwoju określonego przedsięwzięcia w zamian za określone świadczenie zwrotne, który angażuje szerokie grono kapitałodawców, charakteryzuje się wykorzystaniem technologii teleinformatycznych oraz niższą barierą wejścia i lepszymi warunkami transakcyjnymi niż ogólnodostępne na rynku.”

Zatem w szerszym ujęciu finansowanie społecznościowe można interpretować jako pozyskiwanie środków finansowych od zgromadzonej społeczności poprzez sieć komputerową. Obecnie możemy wyróżnić podstawowe modele finansowania społecznościowego:

1. Model donacyjny (*donations model*), zwany charytatywnym, w tym:

1a. Model bez nagradzania uczestników (*non-rewards model*);

1b. Model z nagradzaniem uczestników (*reward-based model*), zwany

sponsorskim.

2. Model pożyczkowy (*lending model*), w tym:

2a. Model mikropożyczek (*microfinance, P2P microfinance*);

2b. Model pożyczek społecznościowych (*social lending, P2P lending*).

3. Model inwestycyjny, w tym:

3a. Model kolektywnego współfinansowania firm (*collective investment*);

3b. Model inwestycyjny – funduszu inwestycyjnego (*investment fund*);

3c. Model inwestycyjny – akcyjny (*securities model*).

4. Rozwiązania mieszane (*mixed solutions*).

W niniejszej pracy inżynierskiej został zastosowany model pożyczkowy łączący ze sobą model mikropożyczek (2a) wraz z modelem pożyczek społecznościowych (2b). W modelu tym gromadzenie funduszy odbywa się poprzez bezpośrednie pożyczanie środków finansowych pomiędzy osobami fizycznymi, z pominięciem tradycyjnych instytucji finansowych np. banków. Mikropożyczki to pomoc finansowa, oferowana w niewielkich kwotach najczęściej dla osób najuboższych, zaś pożyczki społeczne odnoszą się do wyższych kwot, które są pożyczane na określonych regulaminami warunkach. Idea polega na łączeniu ze sobą interesów dwóch stron – inwestorów, posiadających kapitał oraz pożyczkobiorców, którzy potrzebują wsparcia finansowego. Pożyczka odbywa się na zasadzie aukcji pożyczkowej, w ramach której pożyczkobiorca ustala warunki pożyczki i wystawia swoją ofertę. Z kolei pożyczkodawca przeglądając ogłoszenia podejmuję decyzję o okresowym przekazaniu określonej kwoty pieniędzy wybranej osobie.

Tabela 2. Statystyki serwisów pożyczkowych (30.04.2012)

Obraz zawierający stół

Opis wygenerowany automatycznie

Źródło: Dariusz Dziuba, „Rozwój systemów crowdfundingu – modele, oczekiwania i uwarunkowania”, str. 95

MEDIA SPOŁECZNOŚCIOWE – ang. *SOCIAL MEDIA*

Zgodnie z definicją zaproponowaną w 2010 roku przez Chris Treadaway i Mari Smith, media społecznościowe to „zbiór technologii służących do inicjowania komunikacji i przekazu treści między ludźmi, ich znajomymi i sieciami społecznościowymi, do których należą”.

Powołując się na powyższą definicję można wyróżnić poszczególne cechy, które posiadają *social media:*

* zrzeszają różnych użytkowników,
* pozwalają na tworzenie i publikację treści oraz ich ocenę przez użytkowników
* są interaktywne,
* są dostępne dla każdego zainteresowanego
* posiadają dowolną skala wykorzystania, gdyż treści rozprzestrzeniane są poprzez społeczną interakcję

Media tradycyjne w znacznym stopniu odbiegają od mediów społecznościowych. Główne różnice jakie można wskazać:

1. **Zasięg**– **Media tradycyjne posiadają ograniczony zasięg ze względu na aspekty technologiczne oraz kosztowe, gdzie w mediach społecznościowych możliwości te są niemalże nieograniczone i są w stanie dotrzeć do szerokiego grona odbiorców**
2. **Natychmiastowość**– informacje w mediach społecznościowych rozchodzą się znacznie szybciej niż w mediach tradycyjnych.
3. **Trwałość** – informacje, które już zostały udostępnione w mediach tradycyjnych są niemożliwe do zmiany. W mediach społecznościowych mogą one być na bieżąco aktualizowane lub usuwane oraz wielokrotnie przetwarzane.
4. **Dostępność**– w mediach tradycyjnych jednostki zarządzające decydują o formie udostępnienia informacji.. Media społecznościowe pomimo, że również posiadają jednostki zarządzające, są dostępne dla ogółu ludności i są w znacznym stopniu darmowe.
5. **Użyteczność**– media tradycyjne wymagają większego nakładu pracy oraz wyższych kompetencji, aby udostępnianą informację odpowiednio przygotować. Media społecznościowe umożliwiają rozprzestrzenianie informacji bez specjalnego przygotowania.

Media społecznościowe bazują w szczególności na aktywności swoich użytkowników. Zwiększająca się liczba aktywnych osób na portalach przekłada się na korzyści dla poszczególnych portali oraz pomaga w ich rozwoju. Poniższy wykres obrazuje to, jak ogromną liczbę stanowią użytkownicy mediów społecznościowych w dzisiejszych czasach.

Rysunek 6. Wykres przedstawiający media społecznościowe z największą ilością aktywnych użytkowników (w miliardach)

Źródło: opracowanie własne na podstawie danych pochodzących z Statista.com, stan na dzień 20.08.2021

### Terminologia

Student – osoba kształcąca się na uczelni wyższej

Pożyczkobiorca – osoba, która zaciąga pożyczki

Pożyczkodawca – osoba udzielająca pożyczki

Pożyczka – operacja polegająca na udzieleniu przez pożyczkodawce określonej kwoty środków pieniężnych do dyspozycji pożyczkobiorcy

Wniosek – formalna prośba o podjęcie określonych działań lub czynności

*Rating* – ocena wiarygodności pożyczkowej

*Social media* – z ang. media społecznościowe; rozwiązania wykorzystujące technologie internetowe i mobilne, umożliwiające tworzenie oraz wymianę treści przez użytkowników

*Crowdfunding* – z ang. finansowanie społecznościowe; metoda zdobywania środków finansowych od Internautów

*Social lending* – z ang. pożyczki społeczne; polegają na zawieraniu transakcji pomiędzy osobami fizycznymi przez serwisy internetowe

*Blockchain* – z ang. łańcucha bloków; technologia, która służy do przechowywania oraz przesyłania informacji o transakcjach zawartych w sieci

*Smart contracts* (inaczej *chaincode*)– z ang. inteligentne kontrakty; cyfrowe umowy funkcjonujące w formie samoobsługujących się aplikacji

*Ledger* – kluczowy element w Hyperledger Fabric; przechowuje informacje odnośnie obiektów biznesowych oraz historie bieżących, jak i zrealizowanych transakcji. Jest połączeniem *World State* oraz *blockchain* (w odniesieniu do terminologii Hyperledger Fabric)

*World State* – jest kolekcją zmiennych przechowująca rezultaty wykonanych transakcji

*Peer to peer (P2P) –* z ang. każdy z każdym; model komunikacji, w której każdy z węzłów jest równoważny, a więc każdy z nich może wysyłać jak i odbierać dane równorzędnie

Konsensus (ang. *consensus*) – mechanizm pozwalający użytkownikom lub maszynom koordynować ustawienia rozproszone; technika osiągania porozumienia w kwestii dodania nowych bloków.

## Cel pracy

Celem poniższej pracy inżynierskiej jest zaprojektowanie oraz implementacja fragmentów systemu umożliwiającego udzielania mikropożyczek. System będzie opierał się na modelu finansowania społecznościowego, co oznacza, że transakcje będą zawierane bezpośrednio pomiędzy użytkownikami systemu (P2P), wykluczając instytucje finansowe. Zastosowanie technologii *blockchain* ma na celu zapewnienie systemowi bezpieczeństwo, efektywność oraz transparentność. Media społecznościowe będą miały za zadanie weryfikację pożyczkobiorcy oraz ocenę ryzyka pożyczkowego. System ma ponadto umożliwiać promowanie wniosku wśród społeczności, wykorzystując *social-media*.

Praca inżynierska również ma zapewnić osiągniecie celów eksploracji takich jak lepsze zrozumienie dziedziny problemowej, określenie możliwych rozwiązań, a następnie wybór najlepszego z nich i uzasadnienie podjętej decyzji.

## Problemy inżynierskie

Aby osiągnąć cel pracy zostanie przeprowadzona analiza cech i porównanie właściwości dostępnych wariantów technologii łańcucha bloków w odniesieniu do projektowanego systemu. Zastosowanie *smart contracts* pozwoli na omijanie stron trzecich podczas transakcji, co dodatkowo skutkuje eliminacją jakichkolwiek prowizji. Ponadto pozwala to na zaoszczędzenie czasu i ograniczeniu ryzyka. Transakcje mogą być śledzone i nie można ich cofnąć, co sprzyja budowaniu zaufania między użytkownikami oraz zapewnia bezpieczeństwo.

Konieczne będzie zdefiniowanie zasady działania finansowania społecznościowego w systemie. Następnie zostanie określona rola mediów społecznościowych w odniesieniu do systemu. Ostatnim etapem będzie wybór portali społecznościowych, które będą wspomagały proces weryfikacji pożyczkobiorcy oraz promowania złożonego już wniosku.

Kluczowym aspektem będzie zaprojektowanie oraz implementacja systemu. Wyzwaniem trudnym do zrealizowania będzie stanowić integracja warstw systemu – prezentacji, logiki biznesowej oraz *blockchain*.

## Organizacja praktyki inżynierskiej

Aby rozwiązać zidentyfikowane problemy oraz zagwarantować osiągnięcie celu postawionego w pracy dyplomowej należy dokonać wyboru metodyki zarządzania projektem oraz wytwórczej, a ponadto technik i narzędzi. Odpowiednie zastosowanie metodyk wprowadza do naszych działań porządek, ład oraz dobre praktyki.

Do zarządzania projektem zostanie zastosowana metodyka PRINCE2. Z kolei aby określić sposób, w jaki wytwarzany będzie system zostanie wykorzystana metodyka AUP (*Agile Unified Process)*.

### Cykl życia systemu informatycznego

Model cyklu życia projektowanego systemu informatycznego określa jakie czynności realizowane są w poszczególnych fazach oraz ich kolejność. Pozwala to na uporządkowanie przebiegu oraz monitorowanie zadań. W celu wytwarzania oprogramowania został dobrany model spiralny. Składa się on z czterech etapów wykonywanych w iteracjach:

1. Planowanie
2. Analiza ryzyka
3. Weryfikacja
4. Konstrukcja

Został on wybrany ze względu na łatwość wprowadzania zmian oraz możliwość wykrywania błędów nawet we wczesnych etapach. Zaletą tego rozwiązania jest stała możliwość rozwijania oprogramowania.

Rysunek 7. Cykle życie oprogramowania w modelu spiralnym



Źródło: https://www.cri.agh.edu.pl/uczelnia/tad/PSI5/faq.html

### Metodyka zarządzania procesem wytwórczym

Do odpowiedniego sterowania etapami projektu kluczowe jest zastosowanie metodyki zarządczej. Dzięki skorzystaniu z metod zawartych w metodyce możliwe jest usystematyzowanie procesów towarzyszących zarządzaniu projektami, co przekłada się na zakończenie projektu z powodzeniem i osiągnięcie ustalonego celu.

PRINCE2 jest kompleksową metodyką zarządzania projektami, która prezentuje procesowe podejście do całego projektu. Kładzie główny nacisk na działaniach zarządczych i procesach decyzyjnych w projekcie. Projekt definiowany jest jako „tymczasowa organizacja, stworzona w celu dostarczenia jednego lub wielu produktów biznesowych według uzgodnionego uzasadnienia biznesowego".

Rysunek . Projekt według PRINCE2



Źródło: opracowanie własne na podstawie Wyrozębski P., Metodyka PRINCE2 [w] Metodyki zarządzania projektami, wyd. Bizarre, Warszawa 2011

PRINCE2 składa się z czterech powiązanych ze sobą elementów:

– 7 pryncypiów (zasad)

– 7 tematów

– 7 procesów

– środowiska projektów

Pryncypia lub inaczej zasady są fundamentalnymi elementami metodyki, które są niezmienne, a zatem uniwersalne dla wszystkich projektów niezależnie od branży czy rozmiaru. Zastosowanie ich świadczy o zarządzaniu projektem według Metodyki PRINCE2. Zostały one sformułowane z doświadczeń wynikających z tworzonych w przeszłości projektów. Są to:

1. Ciągła zasadność biznesowa
2. Korzystanie z doświadczeń
3. Zdefiniowane role i obowiązki
4. Zarządzanie etapowe
5. Zarządzanie z wykorzystaniem tolerancji
6. Koncentracja na produktach
7. Dostosowanie do warunków projektu

Kolejnym elementem metodyki są tak zwane tematy, czyli obszary, którymi należy zajmować się na poszczególnych etapach projektu. Ponadto mają one za zadanie wspieranie realizacje procesów. Wyróżniamy następujące etapy:

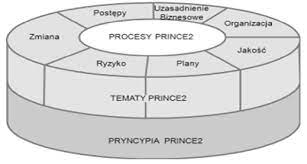
1. Uzasadnienie biznesowe,
2. Organizacja
3. Jakość
4. Plany
5. Ryzyko
6. Zmiana
7. Postępy

Główną częścią metodyki stanowią procesy. Zapewniają one prowadzenie projektu od początku do końca w sposób kontrolowany. Opisywana metodyka składa się z 7 procesów, które następnie dzielą się na podprocesy:

1. Strategiczne zarządzanie projektem
2. Uruchamianie projektu/przygotowanie założeń projektu
3. Inicjowanie projektu
4. Sterowanie etapem
5. Zarządzanie wytwarzaniem produktów
6. Zarządzanie zakresem etapu
7. Zamykanie projektu

Ostatnim elementem metodyki jest dostosowanie PRINCE2 do środowiska projektu. Aby nadmiernie nie przeciążać projektu czynnościami zarządczymi należy odpowiednio dostosować metodykę. Ma to na celu zachowanie odpowiedniego planowania oraz kontroli, zgodnie ze skalą oraz poziomem złożoności według ustalonych potrzeb.

Rysunek . Struktura metodyki PRINCE2



Źródło: Managing Successful Projects with PRINCE2 by OGC.

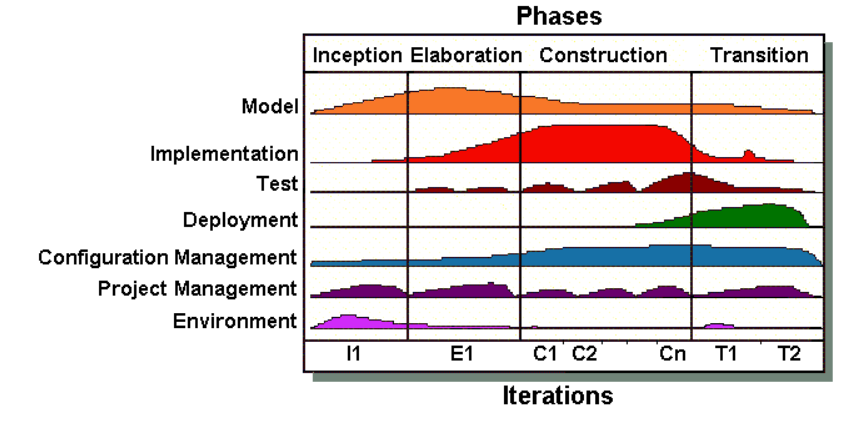
Metodyka PRINCE2 została wybrana do zarządzania procesem wytwórczym systemu z racji tego, że opiera się na zbiorze najlepszych praktyk, co skutkuje mniejszym prawdopodobieństwem popełnienia błędów. Jest skupiona na poprawnym zaplanowaniu czynności oraz monitorowaniu całego przebiegu wytwarzania produktu w każdym z etapów. Dzięki dostarczeniu kompletnego zestawu procesów zarządzania pozwala na spójną kontrolę nad pracą. Możliwość dostosowania metodyki do środowiska projektów świadczy o wysokim poziomie elastyczności.

### Metodyka wytwarzania systemu informatycznego

*Agile Unified Process* w skrócie AUP wywodzi się od *Rational Unified Process* i jest jej uproszczoną lekką wersją. Jest to zwinna metodyka wytwarzania systemu informatycznego, która trzyma się koncepcji RUP. Przedstawia ona proste do zrozumienia podejście do wytwarzania oprogramowania. W przeciwieństwie do RUP, w metodyce AUP wyróżniamy siedem dyscyplin.

1. Modelowanie
2. Implementacja
3. Testowanie
4. Wdrażanie
5. Zarządzanie konfiguracją
6. Zarządzanie projektem
7. Środowisko

Rysunek 10. Fazy *Agile Unified Process*



Źródło: Scott W. Ambler, 2005

Tworząc oprogramowanie według AUP opieramy się na następujących filozofiach.

1. Twoi pracownicy wiedzą, co robią
2. Prostota
3. Zwinność
4. Skoncentruj się na działaniach o wysokiej wartości
5. Niezależność narzędziowa
6. Dostosowanie produktu do własnych potrzeb

Stosując się do powyższych wytycznych nie skupiamy się na szczegółowej opisywaniu, a następnie czytaniu dokumentacji. Wszystko opisane jest prosto i zwięźle. AUP jest metodyką zwinną, a zatem oparta jest na iteracyjno-przyrostowej metodzie wytwarzania oprogramowania. Podczas tworzenia systemu koncentruje się na ważnych działaniach, odrzucając te mało istotne. Pozwala na użycie dowolnego zestawu narzędzi dając swobodę w procesie tworzenia systemu.

Wybierając AUP decydujemy się na wybór pomiędzy tradycyjną, ciężką metodyką, jaką jest RUP oraz metodykami zwinnymi, zachowując ich poszczególne zalety.

### Metody, narzędzia i techniki inżynierii systemów

Jakie informatyczne lub ilościowe metody / techniki / narzędzia dyplomant zamierza wykorzystać/zastosować w procesie rozwiązywania zidentyfikowanych problemów inżynierskich? I dlaczego dobrano takie a nie inne?

**IDE:** IntellIJ IDEA

**Warstwa logiki biznesowej aplikacji:** Java, Gradle

**Warstwa prezentacji:** HTML, CSS, JavaScript(React, Angular)

**Blockchain:** Hyperledger Fabric, IBM Blockchain Platform

**Wirtualizacja i rozproszenie:** Docker

**System kontroli wersji:** Git

**Integracja z mediami społecznościowymi:** Meta for Developers

**NARZĘDZIE DO MODELOWANIA**: Enterprise Architect

## Ocena piśmiennictwa

Przedstawiamy tu charakterystykę wykorzystanych w pracy rodzajów źródeł literaturowych oraz konkluzje z oceny (analiza krytyczna) specjalistycznej literatury dotyczącej wiedzy dziedzinowej dla danego problemu oraz wiedzy dotyczącej nauki (metodologii) oraz rzemiosła (praktyki inżynierskiej = metodyki, metody, narzędzia).

W pracy inżynierskiej należy wykorzystać kilkanaście pozycji literaturowych (nie więcej niż ok. 25 - 30) z przewagą źródeł książkowych, normatywnych i naukowych (czasopism naukowych - przy czym nie jest to computer world, harvard business review itp. - za to bogate zbiory dostępne za darmo dla studentów WAT znajdują się w e-Źródłach BG WAT - są to bazy publikacji m.in.: IEEE, Springer, Wiley).

Wykorzystane w pracy źródła literatury dotyczą takich dziedzin jak technologia *blockchain, crowdfunding, social media*. Każda z nich została sklasyfikowana według swojej dziedziny problemowej. Ich zakres obejmuje podstawowe definicje i pojęcia, istotę działania, właściwości, funkcje oraz klasyfikacje. Bez znajomości literatury specjalistycznej dotyczącej wiedzy dziedzinowej nie byłoby możliwe sformułowanie problemów. Ponadto skorzystano z literatury dotyczącej metodologii, która skupiona jest na praktyce inżynierskiej.

## Wnioski

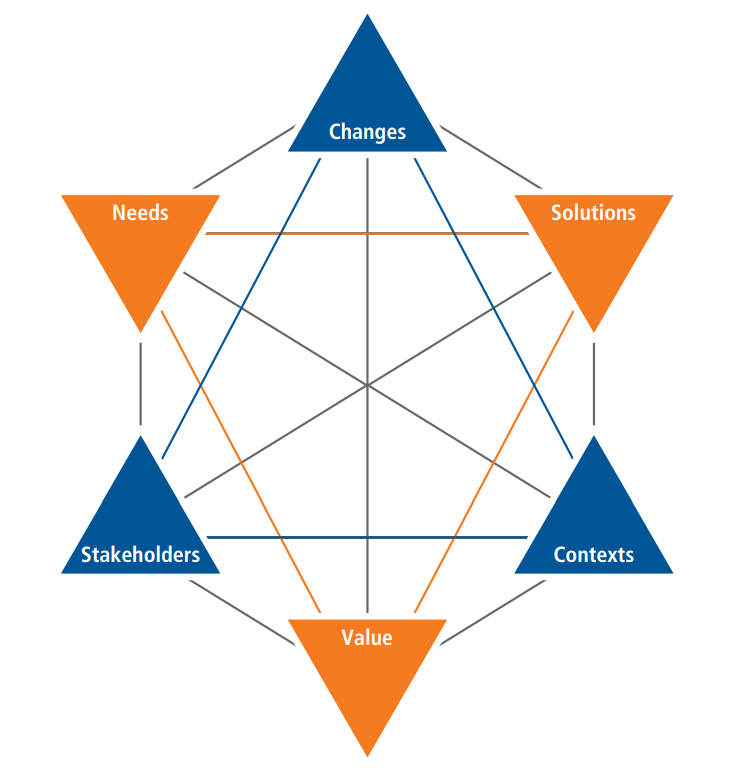
Rozwiązanie zaproponowane w poniższej pracy inżynierskiej dzięki zastosowaniu technologii *blockchain* niweluje zagrożenia oraz problemy związane z procesem obsługi mikropożyczek. Wykorzystanie łańcucha bloków, jako rozproszonego rejestru daje możliwość pominięcia banków oraz wszelkich pośredników, gwarantując transparentność, nieodwracalność i szybkość transakcji oraz jej automatyczne rozliczenie. Za *sprawą smart contracts* zapewniono wiarygodność oraz bezpieczeństwo, gdyż transakcje mają charakter warunkowy. Oznacza to, że operacje wykonują się po spełnieniu wcześniej ustalonych warunków. Cechy jakie oferuje technologia *blockchain* idealnie wpasowuje się w koncepcje modelu finansowania społecznościowego. W tradycyjnych umowach procedury są czasochłonne oraz kosztowne, przekładające się na dużą ilość papierowej dokumentacji, co często prowadzi do rezygnacji z usług przez klienta. W koncepcji, która została nakreślona nie ma na to miejsca.

# aNALIZA BIZNESOWA

## Wprowadzenie

Powołując się na formalną definicję *International Institute of Business Analysis* analiza biznesowa jest do dyscyplina zajmująca się “określaniem potrzeb i rekomendowaniem rozwiązań, które przynoszą wartość interesariuszom”. Analiza biznesowa umożliwia wyartykułowanie potrzeb oraz ich uzasadnienie. Prowadzi się ją w celu stworzenia modelu, pozwalającego na lepsze zrozumienie zasad jej działania oraz funkcjonowania.

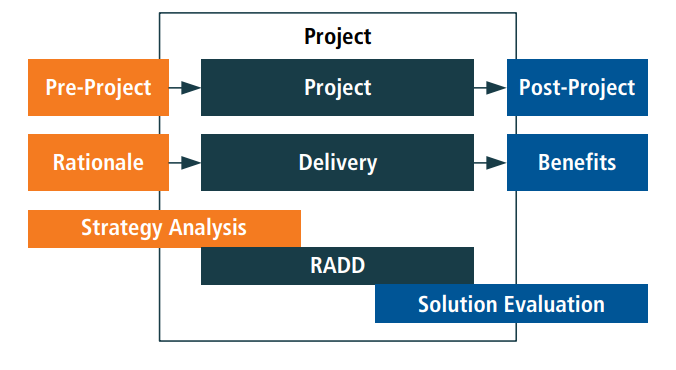
Rysunek 11. Podstawowy model konceptualny analizy biznesowej (The Business Analysis Core Concept Model™)



Źródło: A Guide to the Business Analysis Body of Knowledge®

Poniższy rysunek przedstawia jak obszary wiedzy wspierają dostarczanie wartości biznesowych w poszczególnych etapach cyklu projektu.

Rysunek 12. Analiza biznesowa (*Business Analysis Beyond Projects)* według BABOK®



Źródło: A Guide to the Business Analysis Body of Knowledge®

W niniejszym rozdziale zostanie opracowana analiza biznesowa uszczegóławiając analizę organizacji oraz interesariuszy. Zostanie określone otoczenie organizacji, czyli zbiór zewnętrznych podmiotów, oddziałujących na organizacje. Aby ustalić w jaki sposób ma działać organizacja zostaną zamodelowane procesy biznesowe oraz zdefiniowani aktorzy biznesowi i ich role.

## Analiza organizacji

Celem tego podrozdziału jest zdefiniowanie i przeanalizowanie organizacji, dla której zostanie stworzony system. Poniższe podpunkty zawierają:

* przeznaczenie systemu,
* charakterystykę organizacji,
* dziedzinę działalności,
* otoczenie organizacji

### Charakterystyka organizacji/przedsięwzięcia

Tworzony system przeznaczony jest dla rzeczywistej organizacji społecznej, zrzeszającej społeczność studencką. Jest to organizacja obejmująca cały obszar kraju. Organizacja prowadzi działalność o charakterze dydaktycznym i naukowo-badawczym. Głównym celem organizacji jest nauka – a dokładnie zdobycie wykształcenia wyższego.

### Dziedzina działalności organizacji

Dziedziną rozpatrywanej organizacji jest edukacja. Skupia się ona na zdobywaniu oraz poszerzaniu wiedzy z różnych dziedzin nauki. Ma za zadanie wyposażenie studenta w wiedzę teoretyczną oraz praktyczną, która pozwoli na uzyskanie zatrudnienia zgodnego z posiadanymi kwalifikacjami lub też kontynuację nauki.

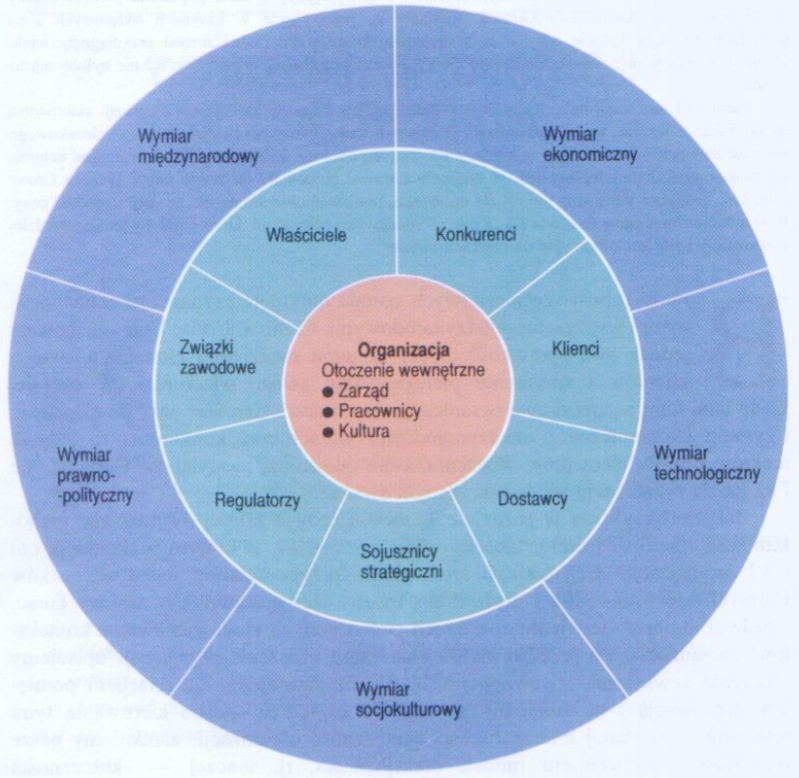
### Otoczenie organizacji

Otoczenie[[2]](#footnote-2) to zbiór różnego rodzaju uwarunkowań społeczno-politycznych, ekonomiczno-prawnych, organizacyjnych, techniczno-technologicznych, kulturowych oraz demograficznych, o charakterze krajowym i międzynarodowym, oddziaływujących w sposób bezpośredni i pośredni na zachowanie się podmiotów gospodarczych stwarzających z jednej strony szanse ich rozwoju, z drugiej zaś – narzucających pewne ograniczenia

Organizacje zawierają otoczenie wewnętrzne jak i zewnętrzne, które następnie dzieli się na dwie warstwy:

* otoczenia ogólnego
* otoczenia celowego

Rysunek 13. Elementy otoczenia organizacji



Żródło: R.W.Griffin, Podstawy zarządzania organizacjami, str. 46

Do zbadania makrootoczenia przedsiębiorstwa zostanie zastosowana analiza PEST, która zestawia ze sobą zewnętrzne czynniki organizacji:

1. polityczno-prawne
2. ekonomiczne
3. społeczno-kulturowe
4. technologiczne

Celem badania jest określenie podstawowych obszarów otoczenia, które mają kluczowy wpływ na funkcjonowanie organizacji. Ponadto dzięki przeprowadzonej analizie PEST ułatwia ona ułożenie przyszłej strategii działania, która z kolei może zapobiec wielu porażkom oraz uzmysłowić wszelkie zagrożenia. Składa się z trzech etapów:

1. Wyróżnienie istotnych czynników dotyczących poszczególnych segmentów otoczenia

2. Ustalenie wpływu każdego z czynników na funkcjonowanie organizacji

3. Określenie relacji między organizacją a makrootoczniem.

Tabela 3. Czynniki analizy PEST

|  |  |
| --- | --- |
| **Polityczno-prawne** | **Ekonomiczne** |
| * Kodeks cywilny * Prawo bankowe * Prawo finansowe * Umowa pożyczki * Ochrona konsumenta * Rozporządzenie o ochronie danych osobowych * Polityka prywatności * Polityka podatkowa * Ustawa o prawie autorskim | * Poziom inflacji * Poziom nakładów na naukę, kulturę i oświatę * Stopa bezrobocia * Stopy procentowe * Zdolność kredytowa * Średnie wynagrodzenie * Inwestycje * Oszczędności * PKB * Ceny produktów |
| **Społeczno-kulturowe** | **Technologiczne** |
| * Konsumpcja * Postęp cywilizacyjny * Poziom wykształcenia * Mobilność społeczna * Styl życia społeczeństwa * Faza rozwoju demograficznego społeczeństwa * Wymagania i oczekiwania konsumentów * Poziom nauki, kultury i sztuki * Etyka pracy | * Poziom technologiczny * Zgodność z normami jakości * Innowacyjność i nowoczesność * Nakłady państwa na badania i rozwój * Szybkość transferu nowoczesnych technologii * Poziom rozwoju transportu * Funkcjonowanie standardów jakościowych, dotyczących bezpieczeństwa i ochrony zdrowia |

1. **Czynniki polityczno-prawne:**

Tworzony system nie jest przeznaczony dla instytucji bankowej, a co za tym idzie nie podlega nadzorowi KNF - Komisji Nadzoru Finansowego. Na chwilę obecną nie ma żadnych regulacji prawnych, które opisują zasady funkcjonowania pożyczek społecznościowych. Pożyczki społecznościowe różnią się od tych tradycyjnych udzielanych przez banki tym, że w przypadku *social lending* obowiązują przepisy Kodeksu cywilnego, co oznacza, że podlegają pod przepisy Prawa bankowego oraz rekomendacje KNF dotyczące instytucji finansowych. Jednakże trzeba mieć na uwadze to, że w przyszłości mogą zostać wprowadzone przepisy regulujące pożyczki społecznościowe, które będą podlegały pod prawo bankowe czy też finansowe.

Decydująca rolę odgrywa szereg praw oraz rozporządzeń dotyczących wykorzystania oraz ochrony danych. Najważniejsze z nich stanowi **rozporządzenie o ochronie danych osobowych** – RODO. Zawiera ono szczegółowe wymogi dla organizacji, które dotyczą przechowywania oraz zarządzania danymi osobowymi w Unii Europejskiej. Ponadto rozporządzenie obejmuje wymagania dotyczące analizy ryzyka związanego z przetwarzaniem danych oraz uwzględnia prawa osób, których przetwarzane dane dotyczą.

W przypadku systemu dane osobowe gromadzone są na skutek procesu obsługi pożyczki. RODO narzuca wytyczne związane z koniecznością informowania klientów o sposobie przetwarzania ich danych osobowych. Dodatkowo istnieje obowiązek prowadzenia rejestru czynności przetwarzania, zawierający informacje na temat administratorów i innych osób, zajmujących się przetwarzaniem danych osobowych, sposoby zabezpieczenia danych osobowych oraz cel ich przetwarzania.

Skutkiem nieprzestrzegania postanowień zawartych w rozporządzeniu o ochronie danych osobowych są kary pieniężne. Dodatkowo organ ochrony danych może nakazać zaprzestania przetwarzania danych osobowych, co uniemożliwia działanie systemu.

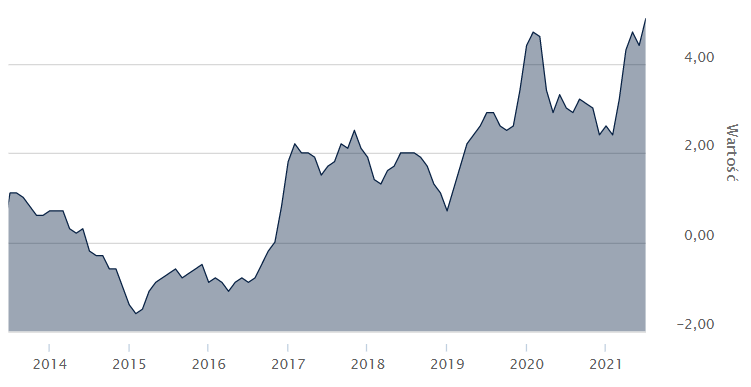
Czynnikiem prawnym, które ma wpływ na funkcjonowanie organizacji stanowi prawo autorskie. Oprogramowanie podlega ochronie jak utwory literackie. Warunkiem jest aby program komputerowy posiadał indywidualne, wyróżniające go cechy, świadczące o charakterze twórczym. Ma to pozytywny wpływ, ponieważ chroni własność intelektualną twórcy przez co pozwala na uniknięcie problemów związanych z naruszeniami praw autorskich oraz przywłaszczeniami autorstwa.

1. **Czynniki ekonomiczne**

Obszar ekonomiczny makrootoczenia ma znaczący wpływ na funkcjonowanie organizacji, ponieważ czynniki z tej strefy decydują o warunkach zaciągnięcia pożyczki. Jest on powiązany z sytuacją gospodarczą kraju. Zawiera informacje dotyczące między innymi średniego wynagrodzenia, poziomu inflacji, stopy bezrobocia oraz stopy procentowej banku centralnego.

Do inflacji dochodzi w momencie, gdy na rynku szybciej wzrasta ilość pieniędzy w stosunku co do produkcji, a co za tym idzie następuje wzrost cen w gospodarce, prowadzący do spadku siły nabywczej pieniądza. Ma to przełożenie na portfel konsumenta, gdyż za te samą sumę pieniędzy jesteśmy w stanie kupić coraz mniej produktów.

Rysunek 14. Poziom inflacji r/r w Polsce w latach 2014-2021



Źródło: <https://www.bankier.pl/gospodarka/wskazniki-makroekonomiczne/inflacja-rdr-pol>

Ponadto inflacja może wiązać się z podwyższeniem stóp procentowych, które ustalane są przez banki centralne. Ma to swoje konsekwencje dla ludzi, którzy posiadają aktywne zobowiązania finansowe, ponieważ wzrost stopy procentowej powoduje zwiększenie części odsetkowej raty.

Kolejnym ekonomicznym czynnikiem, mającym wpływ na organizację stanowi stopa bezrobocia. Jest to miara, która mówi jaki procent udziału w ilości ludności czynnej zawodowo stanowią bezrobotni. W raz jej wzrostem oddziałuje na stan gospodarki kraju, hamując jej rozwój, co za tym idzie wzrasta poziom inflacji. Tak jak w przypadku inflacji, wzrost bezrobocia również oddziałuje niekorzystnie na zainteresowanie pożyczkami. Nie zawsze status osoby bezrobotnej wiąże się z brakiem generowania jakichkolwiek przychodów, jednakże na ogół borykają się z trudną sytuacją finansową. Występuje wtedy niepokój o przyszłą stabilność dochodów, a zatem również obawa przed brakiem możliwości spłaty.

Zestawiając czynniki wchodzące w skład makrootoczenia ekonomicznego należy ponadto uwzględnić zdolność kredytową. Jest to zdolność do terminowej spłaty długu wraz z odsetkami i opłatami, określonymi w umowie. Jej wartość obliczana jest na podstawie wielu czynników takich jak uzyskiwane dochody, miesięczny koszt utrzymania, forma zatrudnienia, obecne zadłużenia, wiek, liczba osób na utrzymaniu, status majątkowy czy historia kredytowa. To na jej podstawie bank decyduje o przyznaniu bądź odmowie pomocy finansowej.

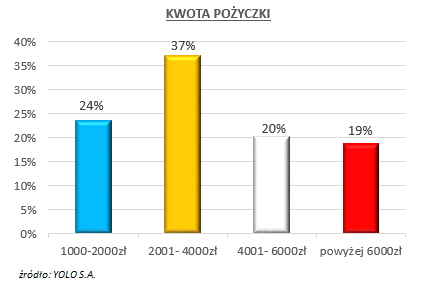
Zdolność kredytowa studentów zazwyczaj jest na niskim poziomie. Będąc młodym studentem banki mogą przestrzegać ich jako ryzykownych klientów ze względu na nieukształtowaną sytuację zawodową oraz brak stabilnego zatrudnienia. Ma na to wpływ między innymi to, że większość studentów nie posiada stałej pracy, a jedynie prace dorywcze na umowach cywilno-prawnych (umowy zlecenie, o dzieło). Znaczenie posiadają również cechy osobowe takie jak wiek, stan cywilny czy wiek. Ponadto zazwyczaj studenci nie posiadają żadnej historii kredytowej.

Zaletą rozwiązania jakim są pożyczki społecznościowe jest fakt, iż wzrost stopy procentowej nie ma znaczenia, ponieważ oprocentowanie nie jest ustalane przez bank centralny, a pożyczkobiorcę, Natomiast wzrost inflacji ze względu na wyższe ceny dóbr może powodować brak środków do życia, co z kolei przyczynia się do konieczności zaciągnięcia pożyczki. Osoby bezrobotne niechętnie będą decydowały się na zobowiązania finansowe, z racji na ich trudną sytuacje finansową. Z drugiej strony nagłe sytuacje mogą wymusić potrzebę zaciągnięcia pomocy finansowej. Kolejnym atutem zaproponowanej koncepcji systemu jest to, że jest dedykowany dla społeczności studenckiej, przez co zdolność kredytowa nie jest wyliczana tak jak w przypadku kredytów udzielanych w tradycyjnych bankach, gdzie student ma bardzo niskie szanse na otrzymanie pomocy finansowej.

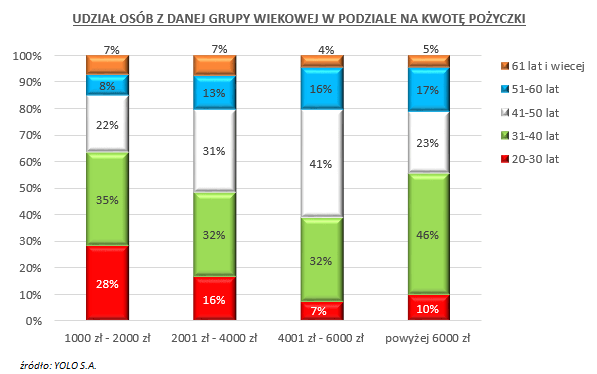
1. **Czynniki społeczno-kulturowe**

Otoczenie społeczno-kulturowe jest powiązane z czynnikiem ludzkim i obejmuje takie aspekty jak faza rozwoju demograficznego społeczeństwa, postęp cywilizacyjny, styl życia konsumentów, konsumpcjonizm, poziom wykształcenia oraz popyt na określone produkty i usługi.

Spółka YOLO S.A., zajmująca się pożyczkami konsumenckimi przeprowadziła badanie dotyczące kwot pożyczek, celów pożyczkowych oraz udział osób z danej grupy wiekowej.



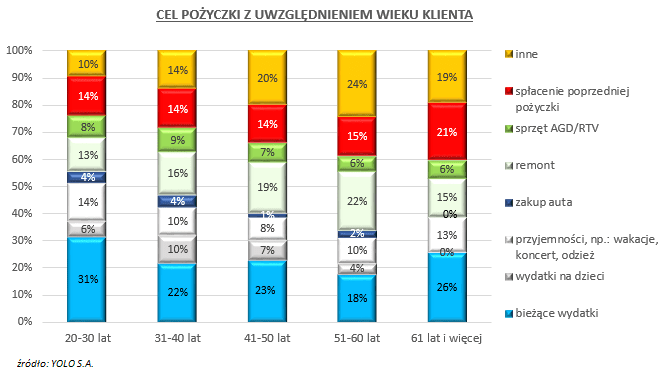
Według ankiety kwotą o jaką najczęściej wnioskują pożyczkobiorcy jest to przedział od 2 001 do 4 000 zł, który stanowi 37% udziału ankietowanych. Rozpatrując powyższy wykres, jednakże uwzględniający udział grup wiekowych nasuwają się pewne wnioski.



Dominujący przedział wiekowy stanowi 31-40 lat, gdzie jedynie przy kwocie 4001 zł – 6000 zł uległ on grupie obejmującej przedział 41-50 lat, która stanowiła 41% udziału. W pozostałych przedziałach kwotowych stanowi on największą część. Powodem tego zjawiska może być fakt, iż ludzie w tym wieku zazwyczaj są już stabilni finansowo i nie będą borykać się z problemem uregulowania długu.

Kolejnym wnioskiem wynikającym z przeprowadzonego badania jest to, że młodzi ludzie najchętniej zaciągają pożyczek o niskich kwotach (1000 – 2000 zł). Niekoniecznie wiąże się to z innymi potrzebami konsumenckimi, jednakże w przeciwieństwie do starszych grup wiekowych, brak stabilności finansowej młodego klienta może wiązać się z większym ryzykiem związanym z udzieleniem pożyczki, a zatem udzieleniem mu mniejszej kwoty.

W przedziałach wiekowych 51-60 lat oraz powyżej 61 lat obserwujemy znaczący spadek aktywności. Może to świadczyć o tym, że ludzie w podeszłym wieku niechętnie zaciągają pożyczek, ze względu na brak zaufania do takich usług.



Mogłoby się wydawać, że potrzeby pożyczkobiorców mogą diametralnie od siebie odbiegać, jednakże powyższy wykres przedstawiający cele pożyczkowe z uwzględnieniem wieku pokazuje, iż w każdej grupie wiekowej jednym z najczęstszych powodów zaciągnięcia pożyczki są bieżące wydatki. Spory udział procentowy również odnotował cel jakim jest remont. Na przyjemności takie jak wakacje, koncerty czy odzież najczęściej zadłużają się młodzi ludzie z przedziału 20-30 lat. Natomiast wydatki na dzieci odnotowały swój największy udział w grupie 31-40 lat. Jest to spowodowane tym, że ludzie w tym wieku najczęściej posiadają dzieci w wieku szkolnym, co wiąże się z licznymi wydatkami.

Powyższe badanie przeprowadzone przez spółkę YOLO S.A. brały pod uwagę odpowiedzi przez każdą grupę wiekową oraz nie uwzględniały statusu społecznego. W celu zestawienia ogólnych wyników z wynikami otrzymanymi od studentów zostanie wykorzystane badanie Barometr Providenta, które ma za zadanie lepsze zrozumienie decyzji finansowych konsumentów oraz ich zachowań. Badanie zostało zrealizowane przez firmę DANAE w dniach 11-15.09.2019 na ogólnopolskiej próbie n=920 studentów I i II stopnia.

Rozpoczęcie studiów wiąże się najczęściej z koniecznością opuszczenia domu rodzinnego i przeprowadzki do innego miasta. Dla wielu studentów jest to pierwsza styczność samodzielnego zarządzania budżetem, który jak wykazało badanie w 59% nie przekracza on 1500 zł netto. Średnia dochodów studenckich wynosi 1518, 72 zł.

Rysunek 15. Główne źródła dochodu studentów

Źródło: opracowanie własne na podstawie danych z https://media-provident.pl/pr/467816/barometr-providenta-studenci-najwiecej-wydaja-na-zakwaterowanie

Według przeprowadzonego badania najczęstszym źródłem dochodu u studentów jest praca dorywcza. Podejmuje się jej ponad połowa badanych – 53%. Następne z kolei źródło dochodów stanowi wsparcie od rodziny, które otrzymuje 45% studentów. Natomiast stałą pracę posiada 26% badanych. Forma zatrudnienia najczęściej zależy od trybu studiów. Studenci dzienni częściej podejmują się prac dorywczych, zaś studenci zaoczeni preferują stabilne wynagrodzenie. Badanie wykazało, że jednym z głównych źródeł dochodów jest również stypendium naukowe. Otrzymuje je 24% wyróżniających się studentów, którzy osiągają szczególne wyniki w nauce. Suma procentowa odpowiedzi przekracza 100%, ponieważ spora część studentów wskazała w odpowiedzi więcej niż jedno źródło dochodu.

W pytaniu odnośnie najważniejszych wydatków, jakimi są obciążeni studenci wskazywali najczęściej na koszty związane z wyżywieniem (średnio 804,60 zł), zakwaterowaniem (średnio 544,50 zł) oraz rachunkami (średnio 492,30 zł).

Rysunek 16. Najważniejsze wydatki studentów

Źródło: opracowanie własne na podstawie danych z https://media-provident.pl/pr/467816/barometr-providenta-studenci-najwiecej-wydaja-na-zakwaterowanie

Ponadto dodatkowymi wydatkami na jakie wydają pieniądze studenci najczęściej typowali samochód i jego utrzymanie, edukację, rozrywkę oraz ubrania.

Rysunek 17. Najczęściej wskazywane przez studentów dodatkowe wydatki

Źródło: opracowanie własne na podstawie danych z https://media-provident.pl/pr/467816/barometr-providenta-studenci-najwiecej-wydaja-na-zakwaterowanie

1. **Czynniki technologiczne**

Są to czynniki, które są narzucane przez współczesne innowacje. Mają na celu ułatwienie i usprawnienie pracy w przedsiębiorstwach. Postępujące zmiany technologiczne mogą przyczynić się zarówno do rozwoju nowych branż, jak i upadku. Technika stwarza kolejne możliwości, przez co determinuje do przeznaczania budżetu na jej rozwój i badania. W ciągu ostatnich lat mogliśmy zauważyć znaczący rozwój gospodarki opartej na technologiach cyfrowych.

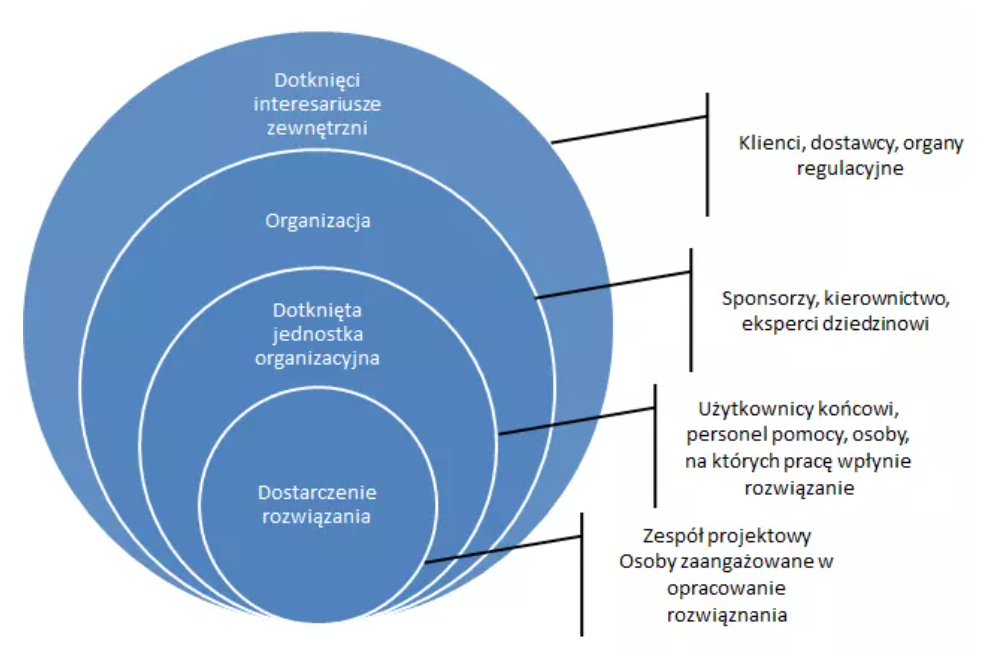
Olbrzymią rolę w kształtowaniu historii pożyczek odegrał Internet. Niegdyś aby otrzymać pomoc finansową była konieczność fizycznego stawiennictwa w instytucji finansowej. W dzisiejszych czasach dzięki rozwoju technologii możemy ubiegać się o dodatkowe pieniądze bez wychodzenia z domu. Zaletą takiego rozwiązania jest szybszy czas realizacji pożyczki oraz wygoda, przez co klienci cenią sobie komfort usługi.

## Analiza interesariuszy

Wnikliwe poznanie oraz zinterpretowanie otoczenia biznesowego wiążą się z dokonaniem odpowiedniej analizy przedsiębiorstwa. Jednym z wielu sposobów badania makrootoczenia jest przeprowadzenie prawidłowej analizy interesariuszy, która przedstawia poziom wpływów na dany projekt oraz w jakim stopniu wywierana jest zależność osób zintegrowanych z daną jednostką. Kształtowanie strategii przedsiębiorstwa powiązane jest z odpowiednią analizą poszczególnych grup interesariuszy. Jednym z głównych celów jest zdefiniowanie potencjalnych zagrożeń dla danego rynku oraz ich eliminacja.

Interesariusze – *stakeholders* – są to osoby, których interesy są w znacznym stopniu powiązane z przedsięwzięciem jednocześnie posiadając możliwość oddziaływania na główne procesy. Podstawowym podziałem są grupy interesariuszy zewnętrznych oraz wewnętrznych. Wyróżnia się również oddziaływanie pozytywne, które ma na celu popierać dany projekt jak i negatywne, które skutkuje uniemożliwieniem realizacji założeń. Zarządzanie projektem zależy od stopnia zaangażowania interesariuszy w projekt.

Rysunek 18. Diagram interesariuszy



*Źródło: BABOK® Guide, wersja 2.0*

W pierwszej kolejności określeni zostają interesariuszy wewnętrzni, którzy stanowią zespól projektowy i osoby zaangażowane w opracowywanie rozwiązania. Z racji tego, że system tworzony jest w ramach pracy inżynierskiej, zespół projektowy składa się z jednej osoby – autora pracy, który sprawuje w nim wszystkie role.

Przechodząc do ogółu, dotknięta jednostką organizacyjną stanowi społeczność studencka, ponieważ użytkownikami końcowymi są studenci zaciągający pomocy finansowej dla zaspokajania własnych potrzeb. Swój udział w ramach organizacji mogą odgrywać rolę samorządy studenckie, reprezentujące poszczególne uczelnie. Z ich strony istnieje możliwość pomocy rozwoju systemu poprzez rozpowszechnianie wniosków dla najbardziej potrzebujących wsparcia. Ponadto Państwo udziela uczelniom dotacji finansowej na pomoc materialną dla studentów. Decyzję do kogo trafią środki podejmuje rektor uczelni w porozumieniu z samorządem, zatem to też może mieć istotny wpływ na przyznawanie pożyczek w ramach systemu.

Z kolei interesariuszami zewnętrznymi będą instytucje finansowe, które stanowią konkurencję. Mogą oddziaływać negatywnie poprzez próby przedstawiania korzystniejszej oferty. W następnej kolejności występują organy regulacyjne. Istnieje ryzyko wprowadzenia nowych kwestii prawnych, które będą uniemożliwiały proces obsługi pożyczki, jaki założony był pierwotnie. Spowoduje to konieczność zmian, które mogą niekorzystnie wpłynąć na działanie systemu.

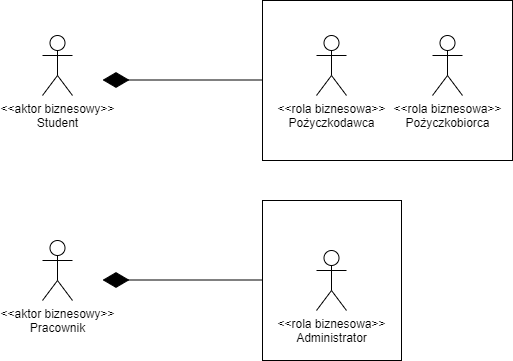
## Role biznesowe i aktorzy biznesowi

Aktorem biznesowym nazywamy podmiot, który wchodzi w interakcje z organizacją. Reprezentuje on rolę odgrywaną względem biznesu, przez co wskazuje jakie byty korzystają z funkcji, które dostarcza organizacja. Z kolei rolą biznesową jest odpowiedzialność za wywoływanie określonego zachowania, do której może być przypisany aktor biznesowy. Pozwala na kategoryzowanie aktorów biznesowych biorąc pod uwagę to, jaką zajmują oni pozycje w procesie biznesowym oraz jakie czynności w nim wykonują.

### Mapa aktorów biznesowych

Opracowywany system obsługi mikropożyczek cechuje się prostotą. Możemy wyróżnić jeden typ aktora biznesowego o nazwie „Student”. Przypisane zostały do niego dwie role biznesowe – „Pożyczkodawca” oraz „Pożyczkobiorca”. Wynika to z faktu, iż każdy student może pełnić obie role i wykonywać działania, które wchodzą w zakres roli pożyczkodawcy jak i pożyczkobiorcy.

Rysunek 19. Aktorzy oraz role biznesowe



Źródło: Opracowanie własne

### Katalog ról biznesowych i odpowiedzialności

Tabela 4. Katalog ról biznesowych

|  |  |  |
| --- | --- | --- |
| **ID** | **Nazwa** | **Opis** |
| ROL001 | Pożyczkodawca | Student upoważniony do udzielania pożyczki |
| ROL002 | Pożyczkobiorca | Student upoważniony do zaciągania pożyczki |

Źródło: opracowanie własne

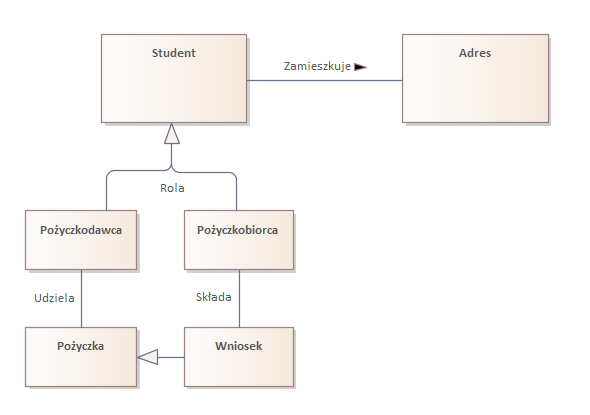
## Obiekty / encje biznesowe

W systemie zachodzi potrzeba przechowywania danych o pewnych obiektach, które charakteryzują się pewnymi cechami i właściwościami. Podstawowym obiektem stanowiącym źródło wszelkich procesów w systemie jest „Student”, który może pełnić rolę „Pożyczkodawca” lub „Pożyczkobiorca” oraz posiada dane adresowe zapisane w obiekcie „Adres”. System ponadto musi gromadzić dane dotyczące wniosków oraz pożyczek.

### Model konceptualny - diagramy klas poziom konceptualny

Poniższy diagram modelu konceptualnego tworzy podstawę schematu, na której będzie się opierać przetwarzanie danych. Obiekty reprezentowane są w uniwersalnym modelu niezależnym od modelu implementacyjnego. Na tym etapie koncepcje biznesowe oraz wymagania są rozważane bez uwzględnienia aspektów technologicznych.

Rysunek 20. Diagram klas – poziom konceptualny

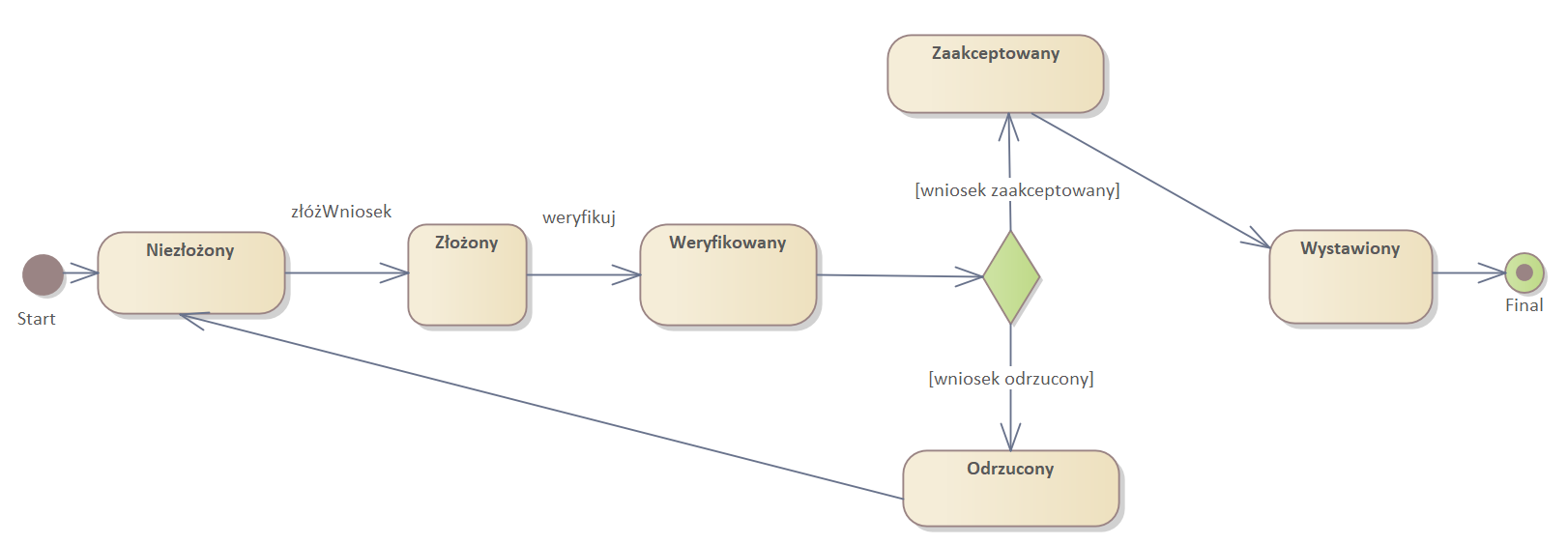


Źródło: Opracowanie własne

### Maszyny stanowe

Wyróżnione obiekty w systemie mogą być przestrzegane jako automat skończony, czyli inaczej jako maszyna stanowa. Jest to graf skierowany posiadający skończoną liczbę wierzchołków, które informują w jakim stanie aktualnie znajduje się obiekt. Z kolei łuki opisują przejścia pomiędzy stanami, wynikające z wystąpienia pewnych zdarzeń. Na diagramie poniżej przedstawiono maszynę stanową obiektu *Wniosek*. Zanim rozpocznie się proces składania wniosku o pożyczkę *Wniosek* przyjmuje stan *Niezłożony*. Po dokonaniu złożenia wniosku przechodzi w stan *Złożony*, a następnie jest on weryfikowany, zatem znajduję się w stanie *Weryfikowany*. Następnie umieszczony jest węzeł wyboru, który posiada dwa wyjścia. W przypadku zaakceptowania wniosku przechodzi on w stan *Zaakceptowany*, natomiast jeśli wniosek został odrzucony przyjmuje stan *Odrzucony*. Odrzucony wniosek wymaga jego ponownego złożenia, tym samym przechodzi w stan *Niezłożony*. Zaakceptowany wniosek po chwili pojawia się jako dostępny do udzielenia pożyczki przez pożyczkodawcę i przyjmuje stan *Wystawiony.*

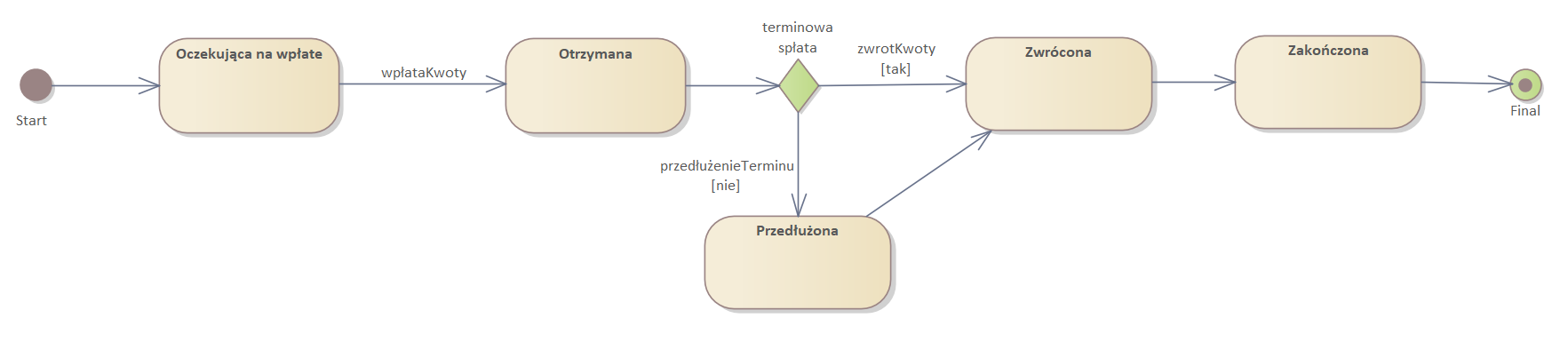
Rysunek 21. Maszyna stanowa obiektu *Wniosek*



Źródło: opracowanie własne

Druga maszyna stanowa została opracowana dla obiektu *Pożyczka*. Po ówczesnym wystawieniu wniosku i wybraniu konkretnej oferty przez pożyczkodawcę *Pożyczka* znajduję się w stanie *Oczekująca na wpłatę*. Gdy następuje wpłata ustalonej kwoty, pożyczkobiorca otrzymuje pożyczkę, dlatego też przechodzi w stan *Otrzymana*. Każda zawarta pożyczka posiada określony termin zwrotu zawarty w umowie, jednakże może dojść do sytuacji, kiedy pożyczkobiorca nie będzie w stanie spłacić zobowiązań na czas. W takim wypadku może poprosić pożyczkodawcę o przedłużenie terminu. Jeśli otrzyma na to zgodę, *Pożyczka* przejdzie w stan *Przedłużona*. Po dokonaniu zwrotu *Pożyczka* znajduje się w stanie *Zwrócona*, a następnie proces jest finalizowany, toteż obiekt występuje w stanie *Zakończona.*

Rysunek 22. Maszyna stanowa obiektu *Pożyczka*



Źródło: opracowanie własne

## Procesy biznesowe

Procesy biznesowe stanowią ciąg uporządkowanych oraz powiązanych ze sobą działań realizowanych w organizacji, które prowadzą do osiągnięcia wyznaczonego celu biznesowego. Wynika on z potrzeb klientów, a jego rezultatem jest ich zaspokojenie. Proces biznesowy można podzielić na podprocesy o własnych atrybutach, które mają wpływ na wynik procesu nadrzędnego.

### Mapa procesów biznesowych

Rysunek 23. Procesy główne

Obraz zawierający tekst

Opis wygenerowany automatycznie

Źródło: Opracowanie własne

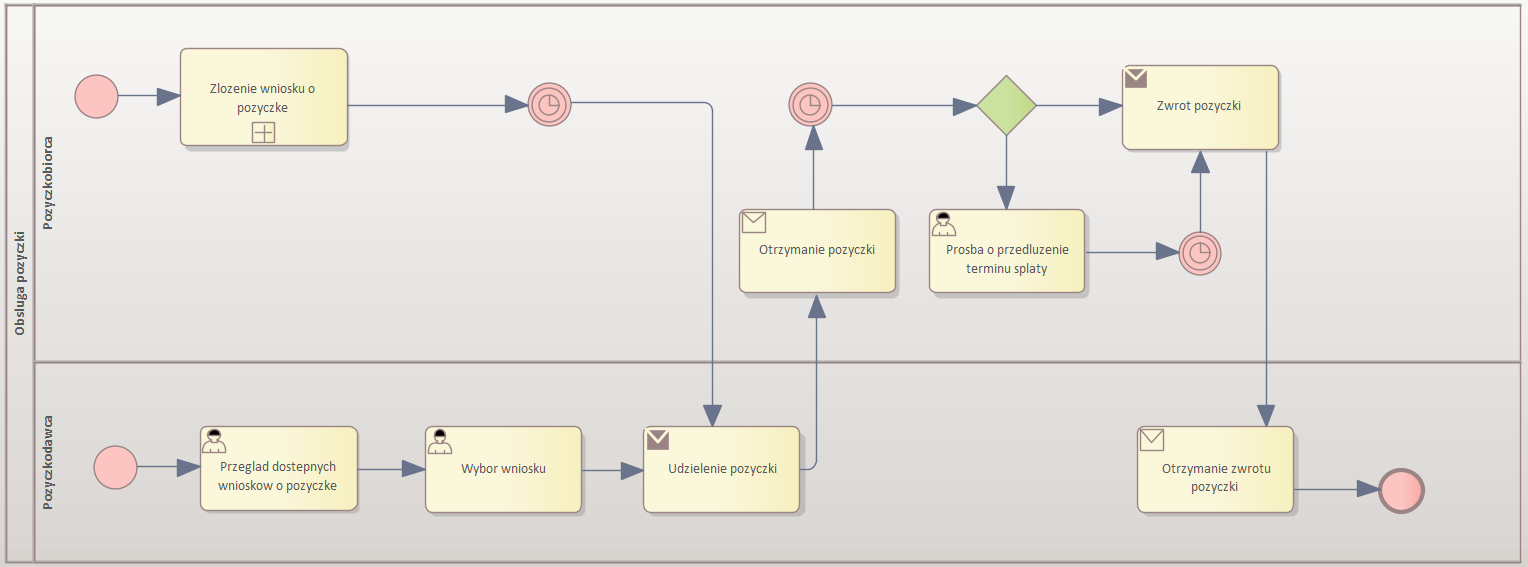
### Macierz CRUD dla relacji PROCES-OBIEKT/ENCJA

### Macierz RACI dla relacji ROLA BIZNESOWA-PROCES BIZNESOWY

### Proces - *obsługa pożyczki*

Poniższy diagram przedstawia jeden z głównych procesów biznesowych występujących w systemie obsługi mikropożyczek. Jest to ogólny, uporządkowany przepływ działań, które prowadzą do osiągnięcia określonego celu – zrealizowanie obsługi pożyczki. Proces ten zawiera podprocesy, które zostały zdekomponowane, w celu przedstawienia szczegółów realizacji danych czynności.

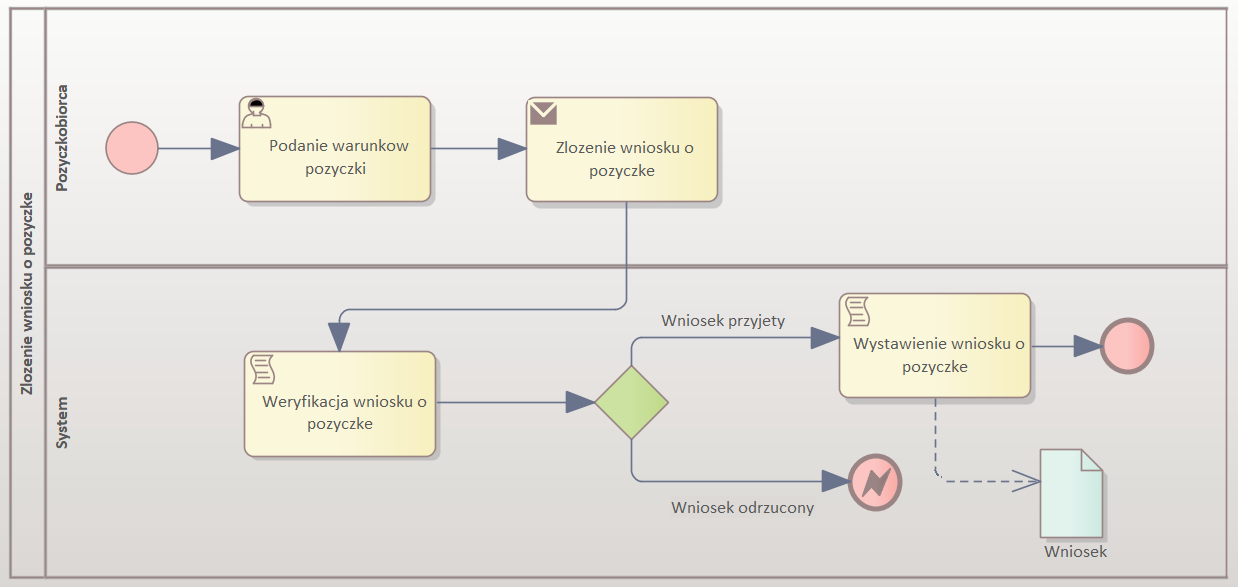
Rysunek 24. Diagram procesu biznesowego - *obsługa pożyczki*



Źródło: Opracowanie własne

### Podproces – *Złożenie wniosku o pożyczkę*

Proces obsługi pożyczki został zdekomponowany na podprocesy. Poniżej przedstawiono diagram 1 poziomu podprocesu biznesowego - *Złożenie wniosku o pożyczkę*

Rysunek 25. Diagram podprocesu biznesowego – *Złożenie wniosku o pożyczkę*

Źródło: Opracowanie własne

## Reguły biznesowe

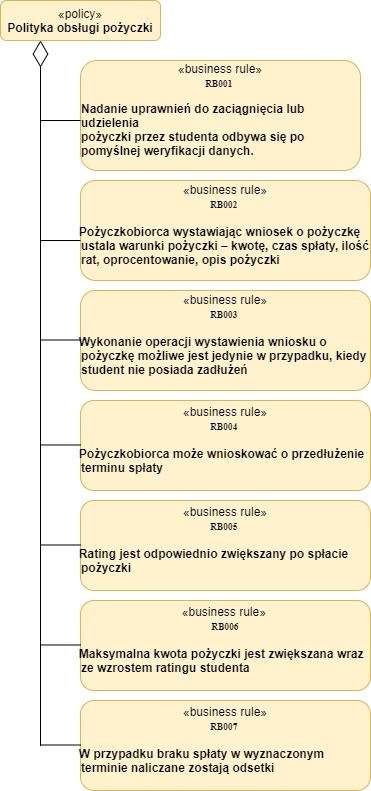
Reguły biznesowe (ang. *Business rules*) stanowią ograniczenia specyficzne, które dotyczą funkcjonowania całego obszaru organizacji. Mają wpływ na zachowanie organizacji poprzez definiowanie lub ograniczanie pewnych aspektów biznesu. Celem reguł biznesowych jest zdefiniowanie struktury biznesowej lub kontrolowanie wpływu na zachowania biznesowe. Ponadto pozwalają zapobiegać niepożądanym przez organizacje sytuacjom praz zapewniają spójność decyzji biznesowych w ramach podjętych działań w organizacji.

Tabela 5. Reguły biznesowe w zakresie polityki obsługi pożyczki

|  |  |
| --- | --- |
| **Polityka obsługi pożyczki** | |
| **ID** | **Opis** |
| RB001 | Nadanie uprawnień do zaciągnięcia lub udzielenia pożyczki przez studenta odbywa się po pomyślnej weryfikacji danych |
| RB002 | Pożyczkobiorca wystawiając wniosek o pożyczkę ustala warunki pożyczki – kwotę, czas spłaty, ilość rat, oprocentowanie, opis pożyczki |
| RB003 | Wykonanie operacji wystawienia wniosku o pożyczkę możliwe jest jedynie w przypadku, kiedy student nie posiada zadłużeń |
| RB004 | Pożyczkobiorca może wnioskować o przedłużenie terminu spłaty |
| RB005 | Rating studenta jest odpowiednio zwiększany po spłacie pożyczki |
| RB006 | Maksymalna kwota pożyczki jest zwiększana wraz ze wzrostem ratingu studenta |
| RB007 | W przypadku braku spłaty w wyznaczonym terminie naliczane zostają odsetki |

Źródło: opracowanie własne

Rysunek 26. Reguły biznesowe wchodzące w zakres polityki obsługi pożyczki

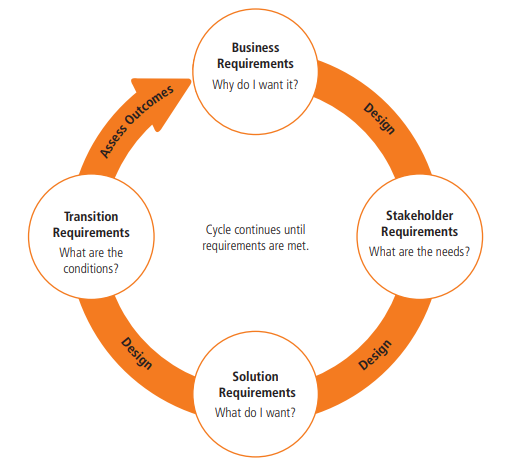


Źródło: Opracowanie własne

## Potrzeby biznesowe

Aby prawidłowo określić i opisać wymagania należy dogłębnie zrozumieć stojące za nimi potrzeby. Wyrażają cel, który musi być osiągnięty i uzasadniają podjęte czynności. Potrzeby biznesowe określają wymagania związane z realizacją celu biznesowego na wysokim poziomie abstrakcji. Mogą powodować zmiany poprzez motywowanie interesariuszy do działania. Specyfikacja potrzeb biznesowych zawiera określenie wymagań, przez co staje się jednym z artefaktów analizy biznesowej.

Rysunek . Cykl specyfikacji potrzeb biznesowych



Źródło: A Guide to the Business Analysis Body of Knowledge® v3

W tym przypadku potrzebą jest utworzenie systemu obsługi mikropożyczek, który zapewni bezpieczną, szybką obsługę pożyczek bez dodatkowych opłat. Jest to cel nadrzędny projektu, który ma zostać osiągnięty. Mając zdefiniowaną potrzebę możemy wówczas przejść do określania poszczególnych wymagań.

### Wymagania biznesowe

[[3]](#footnote-3)Wymagania biznesowe (ang. Business requirements) definiują oczekiwania i rezultaty, których zrealizowanie przekłada się na osiągnięcie celów biznesowych założonych przez organizację. Mogą dotyczyć całości przedsiębiorstwa, obszaru biznesowego lub konkretnej inicjatywy.

Tabela 6. Wymagania biznesowe

|  |  |  |
| --- | --- | --- |
| Typ | Wymaganie | ID |
| «Wymaganie biznesowe» | WB001. Ograniczenie czynności związanych z procesem udzielania pożyczki | WB-001 |
| «Wymaganie biznesowe» | WB002. Zapewnienie możliwości zaciągnięcia jak i udzielenia pożyczki, eliminując strony trzecie. | WB-002 |
| «Wymaganie biznesowe» | WB003. Ograniczenie fizycznego kontaktu z pracownikiem bankowym lub inną osobą fizyczną, będącą pożyczkobiorcą tudzież pożyczkodawcą | WB-003 |
| «Wymaganie biznesowe» | WB004. Ograniczenie czynności związanych z tworzeniem papierowej dokumentacji | WB-004 |

Źródło: Opracowanie własne

### Wymagania interesariuszy

Wymagania interesariuszy (ang. Stakeholder requirements) opisują potrzeby konkretnych grup, które muszą być dostarczone interesariuszom w celu spełnienia wymagań biznesowych.

Tabela 7. Wymagania interesariuszy

|  |  |  |
| --- | --- | --- |
| Typ | Wymaganie | ID |
| «Wymagania interesariuszy» | Rozwiązanie powinno dokonywać weryfikacji w celu sprawdzenia prawidłowości podanych danych osobowych | WI-001 |
| «Wymagania interesariuszy» | Rozwiązanie powinno zapewnić możliwość ponownego wykorzystania danych osobowych studenta gromadzonych w przetwarzanych zbiorach danych | WI-002 |
| «Wymagania interesariuszy» | Rozwiązanie powinno uniemożliwić utworzenie nowego wniosku w przypadku niespełnionych wymagań. | WI-003 |
| «Wymagania interesariuszy» | Rozwiązanie powinno eliminować koszty, które poniósłby student korzystając z usług instytucji finansowych | WI-004 |
| «Wymagania interesariuszy» | Rozwiązanie powinno umożliwić przegląd historii pożyczek innego studenta | WI-005 |
| «Wymagania interesariuszy» | Rozwiązanie powinno umożliwiać przegląd profilu użytkowników | WI-006 |
| «Wymagania interesariuszy» | Rozwiązanie powinno umożliwić przegląd dostępnych wniosków | WI-007 |

Źródło: opracowanie własne

## Diagramy wymagań

Rysunek 28. Diagram wymagań biznesowych

Obraz zawierający tekst

Opis wygenerowany automatycznie

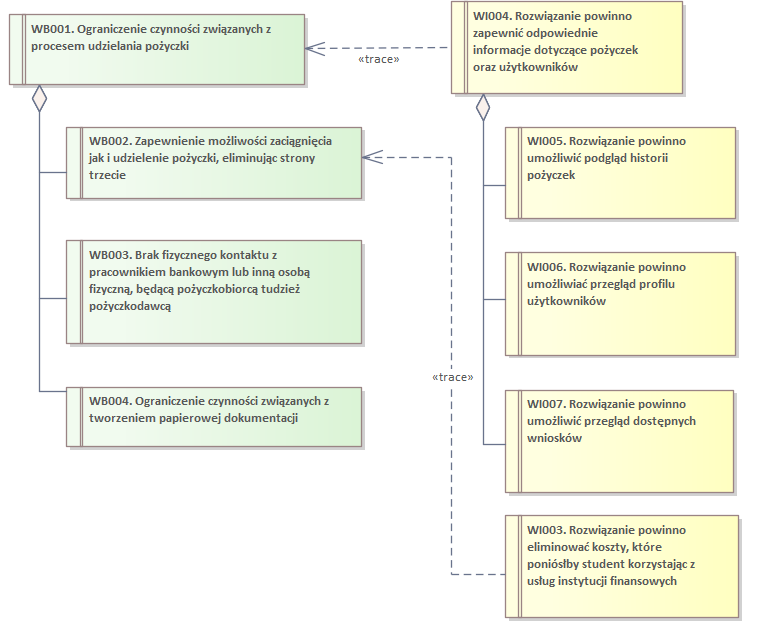
Źródło: Opracowanie własne

Rysunek 29. Diagram wymagań interesariuszy



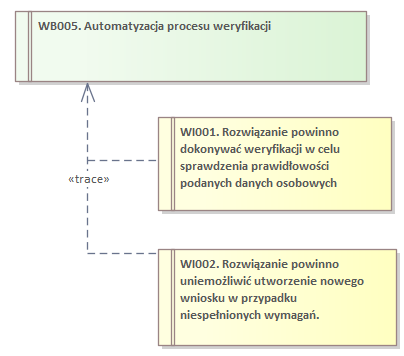
Źródło: opracowanie własne

Rysunek 30. Diagram mapowania wymagań nr 1



Źródło: Opracowanie własne

Rysunek 31. Diagram mapowania wymagań nr 2

  
Źródło: Opracowanie własne

## Wnioski

Analiza biznesowa jest to obszerna, wieloetapowa oraz wieloaspektowa dziedzina. Dzięki przeprowadzonej analizie biznesowej została określona potrzeba. Mając zdefiniowaną potrzebę, na jej podstawie zostały opracowane wymagania, które pozwalają na jej zaspokojenie. Opierając się na analizie otoczenia oraz interesariuszy uzyskano motywację biznesową dla poszczególnych działań organizacji. Wdrożone zostały reguły biznesowe, które wprowadzają pewne ograniczenia oraz zapobiegają niechcianym przez organizacje sytuacjom.

# ANALIZA SYSTEMU INFORMATYCZNEGO

## Wprowadzenie

Analiza systemu informatycznego jest kluczowym etapem w cyklu wytwarzania oprogramowania. Polega na systematycznym badaniu systemu w celu określenia wymagań dotyczących informacji oraz procesów, jakie zachodzą w systemie oraz określenie między nimi relacji. Prowadzi do specyfikacji obserwowalnego zachowania systemu oraz określa co system ma robić, aby spełniać potrzeby użytkownika. Podczas analizy definiowane są wymagania systemowe zgodnie z postawionymi celami oraz zakres funkcjonalności systemu.

## Wymagania funkcjonalne

Wymagania funkcjonalne opisują jakie funkcje ma oferować system oraz jak system ma reagować na poszczególne zdarzenia czy dane wejściowe. Podczas analizy systemu informatycznego określają czego potrzebuje organizacja, aby osiągnąć swoje cele biznesowe, wynikające z wymagań biznesowych. Ponadto definiują czego potrzebuje użytkownik, co z kolei wynika z wymagań użytkownika.

### Obszary funkcjonalne systemu

Wymagania funkcjonalne mogą dotyczyć różnych aspektów systemu. Na skutek tego w systemie obsługi mikropożyczek zostały wyróżnione 3 obszary funkcjonalne:

## zarządzanie kontem

## obsługa pożyczki

## zarządzanie wnioskiem o pożyczkę

Tabela 8. Obszary funkcjonalne systemu

|  |  |
| --- | --- |
| **Obszar** | **Nazwa obszaru funkcjonalnego** |
| A | zarządzanie kontem |
| B | obsługa pożyczki |
| C | zarządzanie wnioskiem o pożyczkę |

Źródło: opracowanie własne

### Wymagania funkcjonalne w obszarze A

Wymagania funkcjonalne w obszarze obsługi pożyczki obejmują:

Tabela 9. Wymagania funkcjonalne w obszarze obsługi pożyczki

|  |  |  |  |
| --- | --- | --- | --- |
| Numer | Typ | Wymaganie | ID |
| 1 | «Funkcjonalne» | System dokonuje obsługi pożyczki | WF-001 |
| 1.1 | «Funkcjonalne» | System umożliwia przegląd dostępnych wniosków o pożyczkę | WF-002 |
| 1.2 | «Funkcjonalne» | System umożliwia złożenie wniosku o pożyczkę | WF-003 |
| 1.2.1 | «Funkcjonalne» | System wymaga podania warunków pożyczki - kwota, czas, oprocentowanie, rodzaj spłaty, opis wniosku | WF-004 |
| 1.2.2 | «Funkcjonalne» | System dokonuje weryfikacji spełnienia warunków przyznania pożyczki | WF-005 |
| 1.3 | «Funkcjonalne» | System umożliwia proces udzielenia pożyczki | WF-006 |
| 1.4 | «Funkcjonalne» | System umożliwia proces otrzymania pożyczki | WF-007 |
| 1.5 | «Funkcjonalne» | System umożliwia prośbę o przedłużenie terminu spłaty | WF-008 |
| 1.6 | «Funkcjonalne» | System umożliwia proces zwrotu pożyczki | WF-009 |

Źródło: opracowanie własne

### Wymagania funkcjonalne w obszarze B

Tabela 10. Wymagania funkcjonalne w obszarze zarządzania kontem

|  |  |  |  |
| --- | --- | --- | --- |
| Numer | Typ | Wymaganie | ID |
| 2 | «Funkcjonalne» | System umożliwia zarządzanie kontem | WF-010 |
| 2.1 | «Funkcjonalne» | System umożliwia rejestracje konta | WF-011 |
| 2.2 | «Funkcjonalne» | System umożliwia logowanie do konta | WF-012 |
| 2.3 | «Funkcjonalne» | System umożliwia edycje profilu | WF-013 |
| 2.4 | «Funkcjonalne» | System dokonuje weryfikacji danych | WF-014 |
| 2.5 | «Funkcjonalne» | System umożliwia przegląd historii pożyczek | WF-015 |
| 2.6 | «Funkcjonalne» | System umożliwia przegląd profilu | WF-016 |
| 2.7 | «Funkcjonalne» | System umożliwia integrację profilu z zewnętrznym serwisem społecznościowym | … |

Źródło: opracowanie własne

### Wymagania funkcjonalne w obszarze C

Tabela 11. Wymagania funkcjonalne w obszarze zarządzanie wnioskiem o pożyczkę

|  |  |  |  |
| --- | --- | --- | --- |
| Numer | Typ | Wymaganie | ID |
| 3 | «Funkcjonalne» | System umożliwia zarządzanie wnioskiem o pożyczkę | WF-017 |
| 3.1 | «Funkcjonalne» | System umożliwia usunięcie wniosku o pożyczkę | WF-018 |
| 3.2 | «Funkcjonalne» | System umożliwia udostępnienie wniosku o pożyczkę wykorzystując media społecznościowe | WF-019 |
| 3.3 | «Funkcjonalne» | System umożliwia edycje opisu wniosku | WF-020 |

Źródło: opracowanie własne

## Wymagania niefunkcjonalne

Wymagania niefunkcjonalne definiują ograniczenia, przy których system powinien realizować swoje funkcje oraz jak system powinien realizować swoje zadania. Określają pożądane cechy (np. wydajność, rozmiar, niezawodność, bezpieczeństwo, łatwość użytkowania) tworzonego systemu, przez co mają wpływ na typ architektury systemu.

### Wydajność

Tabela 12. Wymagania niefunkcjonalne dotyczące wydajności

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numer | Typ | Cecha | Wymaganie | ID |
| 1 | «Niefunkcjonalne» | Wydajność | Czas reakcji na zdarzenie wywołane przez użytkownika nie powinien przekroczyć 2 sekund | WNF-001 |
| 2 | «Niefunkcjonalne» | Wydajność | Ilość przetworzonych transakcji powinna wynosić co najmniej 10 na sekundę | WNF-002 |
| 3 | «Niefunkcjonalne» | Wydajność | System powinien umożliwiać na korzystanie z niej przez 1000 osób równocześnie | WNF-003 |
| 4 | «Niefunkcjonalne» | Wydajność | System będzie dostępny dla użytkowników 24/7/365 średnio 99,5% czasu | WNF-004 |

Źródło: opracowanie własne

### Rozmiar

Tabela 13. Wymagania niefunkcjonalne dotyczące rozmiaru

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numer | Typ | Cecha | Wymaganie | ID |
| 5 | «Niefunkcjonalne» | Rozmiar | Wymagana pamięć RAM – 1 GB | WNF-005 |
| 6 | «Niefunkcjonalne» | Rozmiar | Struktura systemu wraz z danymi nie powinna przekraczać 500 GB | WNF-006 |

Źródło: opracowanie własne

### Niezawodność

Tabela 14. Wymagania niefunkcjonalne dotyczące niezawodności

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numer | Typ | Cecha | Wymaganie | ID |
| 7 | «Niefunkcjonalne» | Niezawodność | Prawdopodobieństwo błędnego wykonania podczas realizacji transakcji nie powinno przekroczyć 0.5% | WNF-007 |
| 8 | «Niefunkcjonalne» | Niezawodność | Prawdopodobieństwo utraty danych w przypadku awarii nie powinno przekroczyć 0.2 % | WNF-008 |
| 9 | «Niefunkcjonalne» | Niezawodność | Błędy krytyczne systemu powinny zostać naprawione w przeciągu 48 godzin | WNF-009 |

Źródło: opracowanie własne

### Bezpieczeństwo

Tabela 15. Wymagania niefunkcjonalne dotyczące bezpieczeństwa

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numer | Typ | Cecha | Wymaganie | ID |
|  | «Niefunkcjonalne» | Bezpieczeństwo | System ma wykorzystywać technologie łańcucha bloków w celu zabezpieczenia przebiegu transakcji |  |
|  | «Niefunkcjonalne» | Bezpieczeństwo | Transakcje mają wykonywać się tylko w przypadku spełnienia warunków |  |
| 10 | «Niefunkcjonalne» | Bezpieczeństwo | System musi spełniać wszystkie wymagania wynikające z RODO | WNF-010 |
| 11 | «Niefunkcjonalne» | Bezpieczeństwo | Logowanie do konta zabezpieczone dwuetapowym uwierzytelnianiem | WNF-011 |
| 12 | «Niefunkcjonalne» | Bezpieczeństwo | Odporność na ataki Denial of Service – DDOS | WNF-012 |
| 13 | «Niefunkcjonalne» | Bezpieczeństwo | System powinien być odporny na błędy użytkownika – odpowiednia walidacja | WNF-013 |
| 14 | «Niefunkcjonalne» | Bezpieczeństwo | System nie powinien ujawniać żadnych niepożądanych danych osobowych użytkowników | WNF-014 |

Źródło: opracowanie własne

### Łatwość użytkowania

Tabela 16. Wymagania niefunkcjonalne dotyczące łatwości użytkowania

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numer | Typ | Cecha | Wymaganie | ID |
| 15 | «Niefunkcjonalne» | Łatwość użytkowania | Liczba stron dokumentacji powinna wynosić maksymalnie 5 stron. | WNF-015 |
| 16 | «Niefunkcjonalne» | Łatwość użytkowania | Czas do opanowania obsługi systemu nie powinien przekroczyć 2 godzin | WNF-016 |
| 17 | «Niefunkcjonalne» | Łatwość użytkowania | System powinien zawierać instrukcję dla użytkownika | WNF-017 |
| 18 | «Niefunkcjonalne» | Łatwość użytkowania | Interfejs powinien być prosty w obsłudze przez użytkownika | WNF-018 |

Źródło: opracowanie własne

### Inne

Tabela 17. Inne wymagania niefunkcjonalne

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numer | Typ | Cecha | Wymaganie | ID |
| 19 | «Niefunkcjonalne» | Inne | System działa poprzez przeglądarkę internetową | WNF-019 |
| 20 | «Niefunkcjonalne» | Inne | Użytkownicy mogą zgłaszać błędy na dedykowany adres email | WNF-020 |
| 21 | «Niefunkcjonalne» | Inne | Strona powinna być przystosowana do różnych wielkości ekranu | WNF-021 |
| 22 | «Niefunkcjonalne» | Inne | Kod źródłowy napisany w języku angielskim | WNF-022 |

Źródło: opracowanie własne

## Założenia i ograniczenia

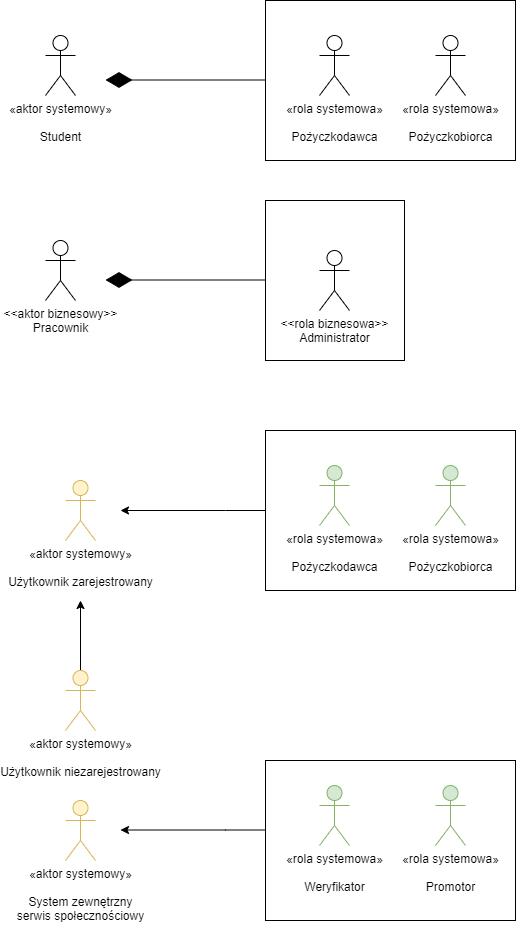
### Założenia

### Ograniczenia

## Aktorzy i role systemowe

Aktor jest rolą, którą pełni użytkownik w stosunku do systemu oraz przypadków użycia. W systemie występuje aktor systemowy *„użytkownik niezarejestrowany”*, który po dokonaniu rejestracji zmienia się w *„użytkownika zarejestrowanego”*. Po przejściu procesu rejestracji, aktor systemowy pełni role systemową „*pożyczkodawca”* lub *„pożyczkobiorca”*. Aktor systemowy nie jest to zawsze obiekt fizyczny. Może być reprezentowany również poprzez zewnętrzne systemy. W przypadku opracowywanego systemu obsługi mikropożyczek ma on wykorzystywać media społecznościowe. Stanowią one aktora systemowego „serwis społecznościowy”, który pełni role systemowe takie jak „weryfikator” oraz „promotor”.

Rysunek 32. Diagram aktorów i ról systemowych



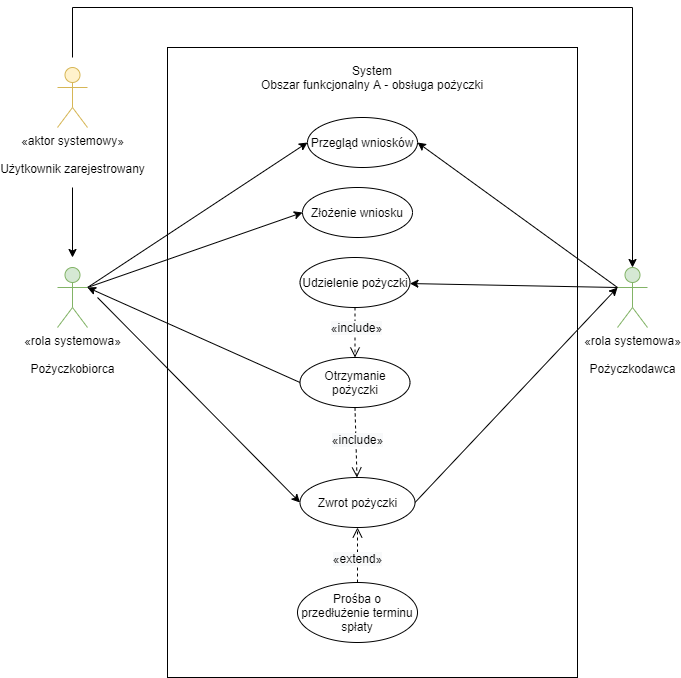
Źródło: Opracowanie własne

#TODO ADMIN

## Przypadki użycia

Dzięki przypadkom użycia zostały przedstawione interakcje pomiędzy aktorami systemowymi oraz systemem, które prowadzą do uzyskania konkretnego rezultatu. Biorąc pod uwagę system tworzony na potrzeby pracy inżynierskiej zostały zdefiniowane 3 obszary funkcjonalne, które posiadają poszczególne przypadki użycia.

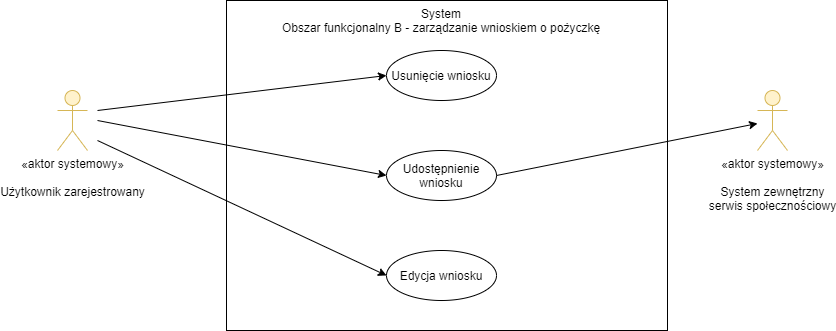
### Przypadki użycia w obszarze funkcjonalnym A

Rysunek 33. Diagram przypadków użycia w obszarze funkcjonalnym obsługi pożyczki

Źródło: Opracowanie własne

### Przypadki użycia w obszarze funkcjonalnym B

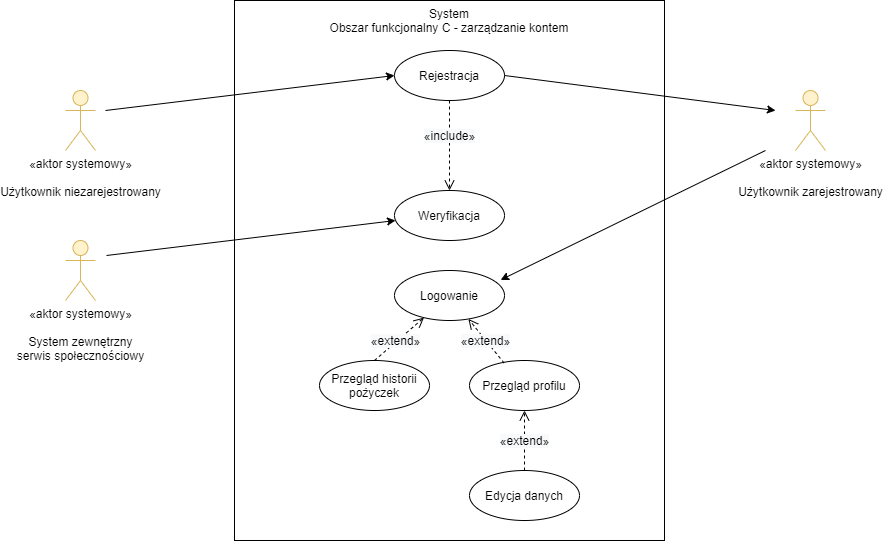
Rysunek . Diagram przypadków użycia w obszarze funkcjonalnym zarządzanie wnioskiem o pożyczkę

****

Źródło: Opracowanie własne

### Przypadki użycia w obszarze funkcjonalnym C

Rysunek 35. Diagram przypadków użycia w obszarze funkcjonalnym zarządzania kontem



Źródło: Opracowanie własne

## Reguły systemowe

Tabela 18. Tabela zestawiająca reguły systemowe

|  |  |
| --- | --- |
| **ID** | **Opis** |
| RS001 | Użytkownik otrzymuje 30 pkt. *rating* za obowiązkowe weryfikacje do uzyskania pożyczki. |
| RS002 | Pożyczkobiorca otrzymuje 20 pkt. *rating* za terminową spłatę pożyczki, a w przypadku spłaty w przedłużonym terminie 5 pkt. *rating* |
| RS003 | Kwota pożyczki wzrasta odpowiednio wraz ze wzrostem ratingu |
| RS004 | Oprocentowanie w skali roku może wynosić maksymalnie 9% |
| RS005 | Po upływie terminu spłaty pożyczki naliczone zostają odsetki |
| RS006 | Oprocentowanie odsetek wzrasta wraz z upływem dni po terminie spłaty pożyczki |
| RS007 | Oprocentowanie odsetek może wynosić maksymalnie 7% |
| RS008 | Pożyczkobiorca otrzymuje powiadomienie 3 dni przed terminem spłaty pożyczki o konieczności uregulowania zobowiązania |

Źródło: Opracowanie własne

### Diagram reguł decyzyjnych

### Tablice decyzyjne

Poniżej znajduje się tabela decyzyjna dla reguł systemowych – RS001 oraz RS002. Znajdują się w niej zasady na jakich przyznawane są pkt. *rating* dla pożyczkobiorców. Powołując się na rys. 37 - drzewo decyzyjne dotyczące reguły systemowej RS003 wynika fakt, iż aby móc zaciągnąć pożyczkę potrzebne jest co najmniej 30 pkt *rating*. Liczba ta przyznawana jest za dokonanie niezbędnych weryfikacji danych użytkownika.

Tabela 19. Tabela decyzyjna dla reguły systemowej RS00, RS002

|  |  |
| --- | --- |
|  | **Przyznane pkt *rating*** |
| **Dokonanie niezbędnych weryfikacji** | + 30 |
| **Spłata pożyczki w określonym terminie** | + 20 |
| **Spłata pożyczki w przedłużonym terminie** | + 5 |

Źródło: opracowanie własne

Tabela 20. Tabela decyzyjna dla reguły systemowej RS006, RS007 oraz RS008

|  |  |
| --- | --- |
| **Ilość dni po terminie spłaty pożyczki** | **Oprocentowanie odsetek** |
| <10 | 2% |
| <10:60> | 5% |
| >60 | 7% |

Źródło: opracowanie własne

Odsetki wyliczane są w następujący sposób:

gdzie:

K - całość kwoty wraz z naliczonym oprocentowaniem ustalonym na początku umowy

D – liczba dni zwłoki

O – wysokość oprocentowania odsetek

R – liczba dni w roku (zawsze przyjęta jako 365 dni)

Przykład:

K – 4200 zł

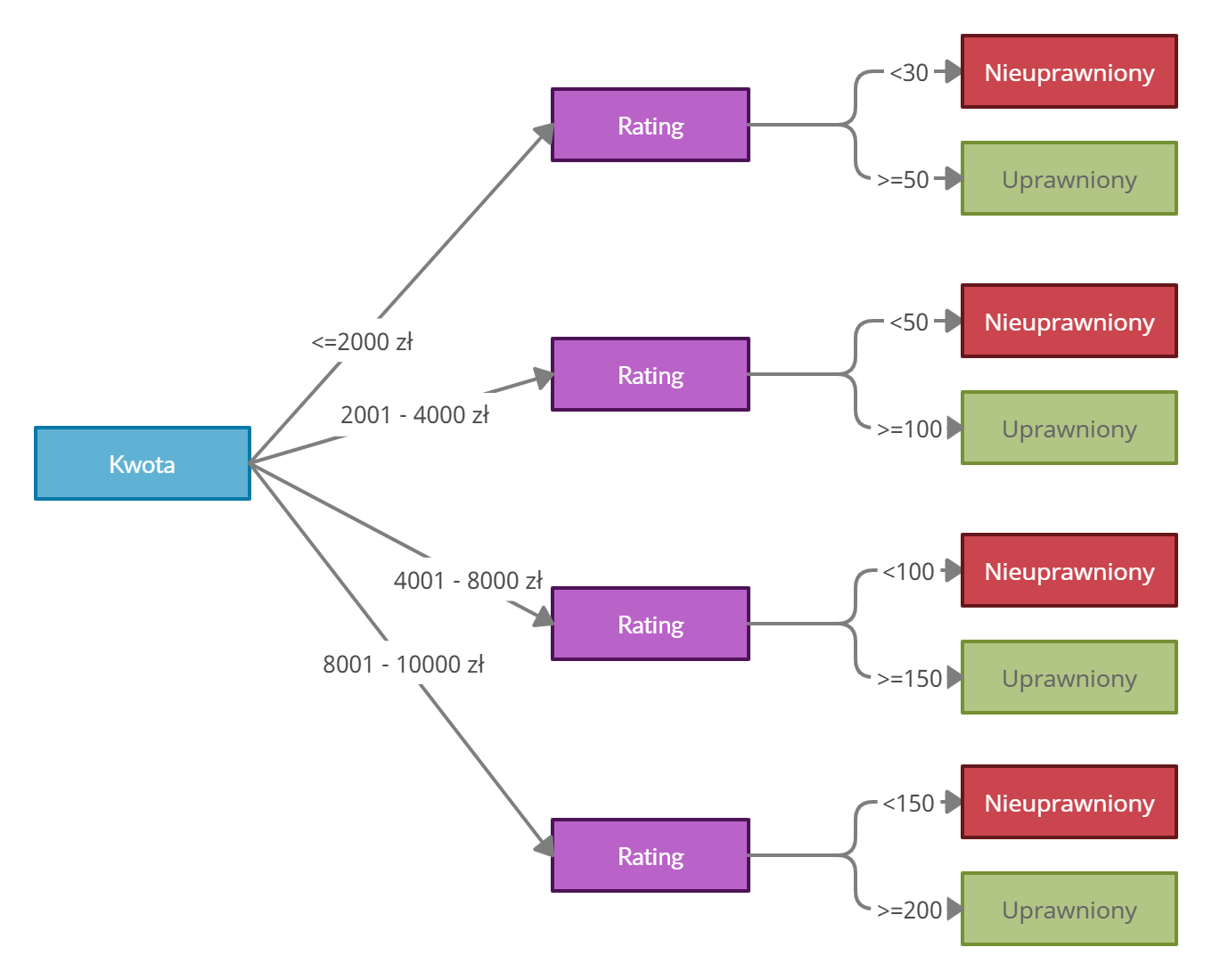
D – 15

Zatem odsetki po upływie 15 dni od terminu spłaty pożyczki określonym w umowie wynoszą 8,63 zł a cała kwota do spłaty wynosi 4208,63 zł. Wzrost oprocentowania ma za zadanie zmobilizowanie do spłaty dłużnika, gdyż z każdym upływającym dniem kwota do spłaty staje się większa. Ponadto ma to na celu dążenie do jak największej spłacalności pożyczek i ograniczeniu ryzyka podjętego przez pożyczkodawców.

### Drzewa decyzyjne

### 

Rysunek 36. Drzewo decyzyjne dotyczące reguły systemowej RS003



Źródło: Opracowanie własne

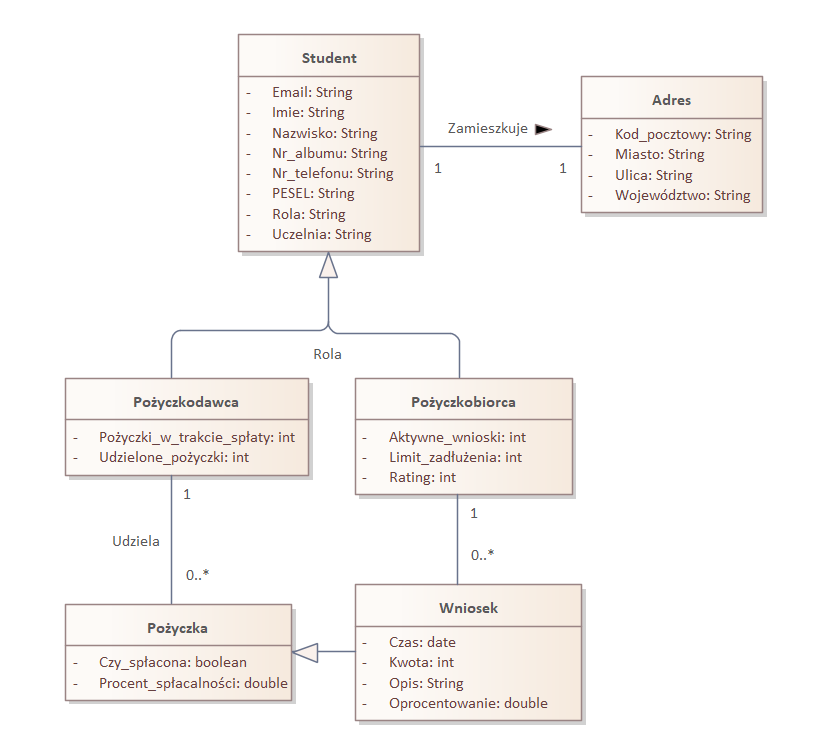
## Model dziedziny systemu

Model dziedziny systemu przedstawia kolekcje obiektów reprezentujących modelowany problem. Stanowi on wewnętrzną strukturę oraz opis mechanizmu działania systemu, jak również przedstawia powiązania pomiędzy elementami. Elementy systemu zwane obiektami posiadają swoją nazwę oraz cechy zwane atrybutami. Cechy są ukryte wewnątrz nich, zatem nie są przedmiotem współdziałania i nie wpływają na komunikację pomiędzy obiektami.

### Diagramy klas - poziom logiczny (atrybuty)

Poniższy diagram przedstawia klasy na poziomie logicznym. Wymagania biznesowe zostały przekształcone w opis możliwy do implementacji. Mimo to nie jest to opis szczegółowy, definiujący wszelkie aspekty technologiczne. Model konceptualnym przekształcił się i został rozszerzony atrybuty klas o ogólnym typie oraz liczebności w związkach pomiędzy nimi.

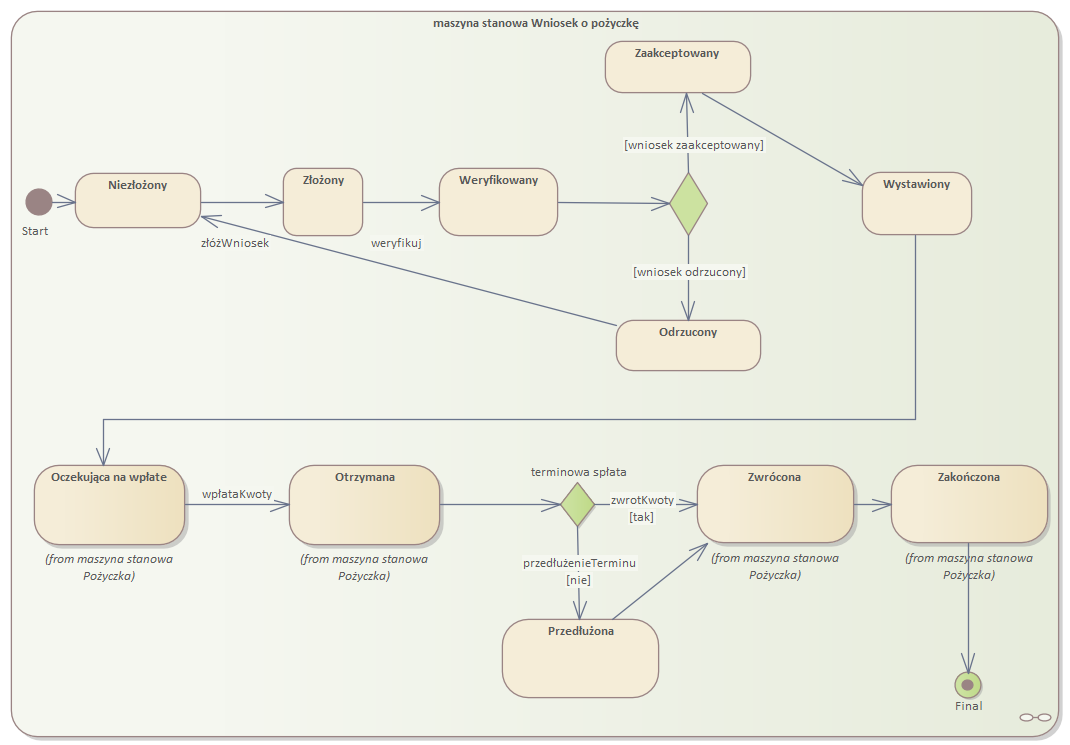
Rysunek 37. Diagram klas – poziom logiczny



Źródło: Opracowanie własne

### Diagramy maszyny stanowej

Poniższy diagram stanowi połączenie maszyny stanowej obiektu *Wniosek* oraz *Pożyczka*. Wynika to z tego, iż wniosek jest składany o pożyczkę. W sytuacji kiedy *Wniosek* znajduje się w stanie *Wystawiony*, pożyczkodawca wybierając go oraz deklarując się na udzielenie pożyczki wywołuje tym samym stan *Oczekująca na wpłatę* obiektu *Pożyczka*.

Rysunek . Diagram maszyny stanowej *wniosek o pożyczkę*

Źródło: opracowanie własne

### Diagramy sekwencji - poziom konceptualny

Pozyczkodawca

przeglądajWnioski() [user] -> wyświetlWnioski() [system] -> wybierzWniosek() [user]-> wyswietlWniosek() [system] -> udzielPozyczki() [pozyczka]-> obciąż(Kwota)[Rachunek\_user] -> wyswietlPotwierdzenie() [system]

Pożyczkobiorca

dodajWniosek() -> wprowadzDane() -> weryfikuj() [system] -> wystawWniosek() [system/dodany do BD] ->

statusWniosku()

## Wnioski

# PROJEKT systemu informatycznego

## Wprowadzenie

W niniejszym rozdziale zostanie opracowana strategia projektowania systemu informatycznego, która pozwala na przedstawienie koncepcji wykonania systemu w różnych wariantach, uwzględniając przy tym przeprowadzoną analizę biznesową oraz techniczną. Proces projektowania systemu informatycznego polega na określeniu architektury sprzętowej i oprogramowania, a ponadto części składowych takich jak komponenty, interfejsy, mechanizmy bezpieczeństwa oraz algorytmy. W etapie projektu systemu informatycznego następuje przejście od modeli konceptualnych do modeli implementacyjnych.

## Strategia projektowania systemu informatycznego

Strategia projektowania systemu informatycznego pozwala przedstawić koncepcję wykonania różnych wariantów oprogramowania na podstawie przeprowadzonej analizy (biznesowej i systemowej). Wyróżniamy 3 warianty:

1. Minimalny
2. Maksymalny
3. Realny

### Wariant minimalny

Wariant minimalny to wariant, w którym istnieje sieć łańcucha bloków, składająca się z węzłów. Sieć wchodzi w interakcje z aplikacją, która składa się jedynie z warstwy logiki biznesowej. Do sieci został wdrożony *chaincode*.

### Wariant maksymalny

Wariant maksymalny obejmuje utworzenie sieci łańcucha bloków, która wchodzi w interakcje z aplikacją, składającej się z warstwy logiki biznesowej oraz prezentacji. Został wdrożony do sieci *chaincode*. Baza danych została w pełni zintegrowana z węzłami. System jest dostępny dla klientów poprzez przeglądarkę. Wszystkie zdefiniowane wymagania co do systemu zostały spełnione. Przebieg procesów biznesowych jest realizowany dokładnie tak jak zostało to zamodelowane.. Zaimplementowane zostały wszystkie zaprojektowane algorytmy, mechanizmy bezpieczeństwa oraz sposoby monitorowania systemu.

### Wariant realny

Wariant realny obejmuje utworzenie sieci łańcucha bloków, która wchodzi w interakcje z aplikacją, składającej się z warstwy logiki biznesowej. Został wdrożony do sieci *chaincode*. Została utworzona baza danych. Zdefiniowane wymagania co do systemu zostały spełnione w 70%. Procesy biznesowe realizowane są w znacznym uproszczeniu. Wprowadzono mechanizmy bezpieczeństwa oraz algorytmy dotyczące przebiegu transakcji.

### Wybór wariantu projektowania systemu informatycznego

A bo ja wiem

## Architektura wysokopoziomowa systemu

„Architektura [systemów] określa podział oprogramowania na komponenty oraz definiuje funkcje tych komponentów i występujące między nimi relacje. Opis architektury oprogramowania pełni w projekcie rolę schematu jego budowy. Poza zakresem projektu architektury oprogramowania pozostaje określenie sposobu realizacji komponentów, nazywane niekiedy projektowaniem szczegółowym. Opracowanie architektury oprogramowania jest zadaniem twórczym, którego treścią jest definiowanie komponentów, których współdziałanie zapewni realizację wymaganego zachowania systemu.” – K.Sacha Inżynieria oprogramowania.

### Decyzje architektoniczne

Specyfikacja wszystkich kluczowych decyzji podjętych na etapie projektowania systemu i wpływających na jego architekturę.

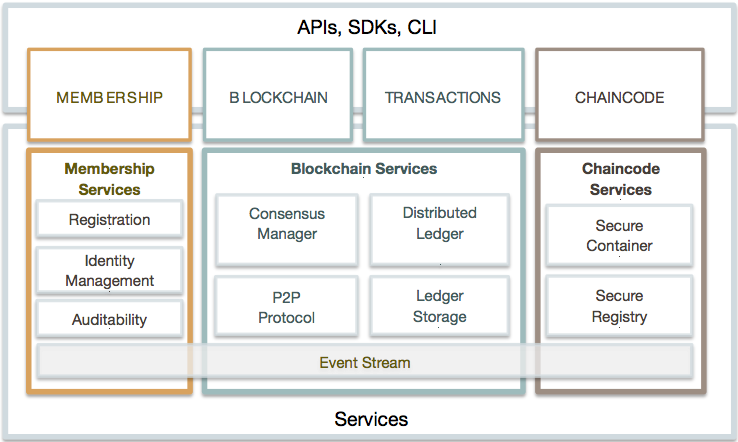
### Wzorce architektoniczne

Wzorce architektoniczne (ang. *architectural pattern*) uznany i sprawdzony sposób rozwiązania danego problemu z zakresu architektury oprogramowania. Określają one ogólną strukturę systemu informatycznego danego typu, elementy z jakich się składa, zakres funkcji realizowany przez dany element jak również zasady komunikacji pomiędzy poszczególnymi elementami. np. architektura wielowarstwowa, architektura klient-serwer, architektura zorientowana usługowo, architektura P2P (peer-to-peer), itd.

### Podział warstwowy i układ warstw systemu

Warstwy są strukturami logicznymi, co za tym idzie nie stanowią fizycznych komponentów. System jest podzielony na 3 warstwy:

1. Warstwa członkostwa (*membership)*
2. Warstwa łańcucha bloków (*blockchain*)
3. Warstwa inteligentnych umów (*smart contract)*

Rysunek 39. Podział warstwowy oraz układ warstw systemu

Źródło: <https://openblockchain.readthedocs.io/en/latest/protocol-spec/#2.1-Architecture>

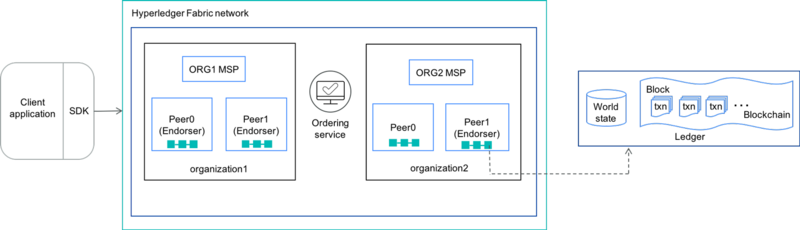
Warstwa członkostwa (ang. *membership*) zapewnia zarządzanie uczestnikami sieci. Jest odpowiedzialna za kontrolowanie prywatności i poufności w utworzonej sieci oraz rejestrowanie tożsamości jej członków. Podmioty rejestrują się w celu uzyskania certyfikatów tożsamości, dzięki czemu mogą być rozróżniane. Certyfikaty są przechowywane w łańcuchu bloków. Ponadto służą do autoryzacji składanych transakcji. Dzięki przyznanym poświadczeniom transakcje mogą być odpowiednie grupowane.

Warstwa łańcucha bloków (ang. *blockchain*) pozwala na zarządzanie rozproszonym rejestrem za pośrednictwem protokołu P2P, opartego na HTTP/2. Połączone ze sobą bloki zawierają dane o bieżących oraz już wykonanych transakcjach. Struktury danych w tej warstwie powiązane są ze sobą dzięki wykorzystania funkcji skrótu. Konfigurowany jest tutaj algorytm konsensusu, czyli technika osiągania porozumienia pomiędzy węzłami sieci.

Warstwa inteligentnych umów (ang. *smart-contract*) zawiera implementacje logiki biznesowej. Zapewnia ona bezpieczny sposób wykonania kodu zawartego w *smart-contract*, ponieważ warstwa jest zamkniętym i zabezpieczonym odizolowanym kontenerem, dzięki czemu nie jest możliwy bezpośredni dostęp do kodu tej warstwy. Inteligentne umowy inicjują nowe transakcje oraz za ich pośrednictwem zmieniają stan rozproszonego rejestru, który znajduje się w warstwie *blockchain*.

### Komponenty systemu

Rysunek 40. Komponenty systemu informatycznego opartego na *Hyperledger Fabric*



Źródło: <https://developpaper.com/basic-knowledge-of-hyperledger-fabric/>

Sieć zbudowana poprzez Hyperledger Fabric zawiera 3 podstawowe komponenty, które odgrywają unikalne role:

* Węzły członków(*peer nodes*) - każdy członek sieci stanowi węzeł, który przechowuje swoją kopię rozproszonego rejestru (tzw. *Ledger* – został on opisany w podrozdziale 4.5.1.) oraz jest odpowiedzialny za uruchomienie inteligentnej umowy zawartej w *chaincode*. Wyróżniamy dwa typy *peer node*: *endorsing peer* oraz *commiting peer*. Każdy węzeł jest *commiting peer*, jednakże niektóre z nich posiadają specjalną rolę *endorsing peer*, która jest odpowiedzialna za zatwierdzanie transakcji.
* Węzły zleceniodawcy (*orderer nodes, ordering service*) – jest to komponent sieci, który ma za zadanie weryfikacje certyfikatów tożsamości, odbieranie i zatwierdzanie transakcji, tworzenie nowych bloków oraz dodawania ich do rejestru, a następnie dostarczanie ich do odpowiednich węzłów.
* Dostawca usług członkowskich (*Membership Service Provider,* MSP) – komponent oferujący operacje dotyczące członkostwa. Z racji tego, iż Hyperledger Fabric jest prywatną siecią *blockchain* z upoważnieniami, jej użytkownicy muszą potwierdzać swoje certyfikaty tożsamości w celu uczestniczenia w sieci.

Każda organizacja uczestnicząca w sieci będzie posiadać węzły oraz nadane im certyfikaty tożsamości. Węzły członków będą odgrywać role, które zostaną zdefiniowane poprzez odpowiednie certyfikaty tożsamości – pożyczkodawcy lub pożyczkobiorcy. Ponadto konieczne jest, aby w każdej sieci znajdował się przynajmniej jeden serwis zleceniodawcy (*ordering service*), który może zostać rozdysponowany między wiele organizacji. Wprowadza on konsensus w celu uporządkowania przesłanych przez członków transakcji.

Komponent systemu, który komunikuje się z zewnątrz z siecią utworzoną poprzez *Hyperledger Fabric* jest *SDK Java*. Umożliwia on zarządzanie siecią, przez co zapewnia:

* wdrażanie oraz wywoływanie *chaincode*,
* tworzenie kanałów,
* rejestrowanie użytkowników,
* tworzenie zapytań dotyczących transakcji i bloków,
* monitorowanie zdarzeń w sieci.

### Interfejsy

Podstawowym interfejsem *Hyperledger Fabric* jest REST API. Umożliwia aplikacjom rejestrowanie użytkowników, dokonywanie transakcji oraz wysyłanie zapytań pod konkretne punkty końcowe (ang. *endpoint*). Składa się z następujących kategorii:

1. *Chaincode* API – umożliwia wdrożenie nowego *chaincode* lub wysłanie zapytania pod istniejący; zawiera parametry takie jak *chaincodeID*, typ, nazwe
2. *Block* API – umożliwia uzyskanie danych odnośnie zawartości bloków
3. *Blockchain* API – umożliwia uzyskanie danych o obecnym stanie łańcucha bloków takie jak liczba bloków, obecny oraz poprzedni *hash*
4. *Network* API – umożliwia uzyskanie danych o węzłach znajdujących się w łańcuchu bloków takie jak liczba węzłów, ID, adres, typ
5. *Registra*r API – umożliwia zarządzanie użytkownikami oraz ich certyfikatami tożsamości.
6. *Transactions* API – umożliwia uzyskanie danych o transakcjach takie jak unikalny identyfikator, typ, *chaincodeID*, znacznik czasowy, certyfikat.

Ponadto podobne zastosowanie posiada wbudowany interfejs wiersza poleceń (ang. *Command Line Interface)*, dzięki któremu możemy administrować siecią *blockchain*.Umożliwia szybkie testowanie i debugowanie kodu oraz formułowanie zapytań o status transakcji. Jest to pomocny interfejs podczas tworzenia aplikacji, ponieważ pozwala na modyfikacje łańcucha bloków bez tworzenia kompletnej sieci, co w znaczącym stopniu wspomaga testowanie. Wykorzystując CLI możemy na przykład dokonywać operacji związanych z węzłami, zalogować się jako dany użytkownik, wdrożyć *chaincode* oraz wysłać pod niego zapytanie.

*Membership Service Provider* (MSP) jest interfejsem zarządzającym warstwą członkostwa (ang. *membership*), regulujący reguły, uprawnienia i role węzłów w sieci. Każdy z nich posiada swój Certyfikat Tożsamości, który został nadany dzięki MSP, w celu wzajemnej komunikacji oraz weryfikacji. Domyślną implementację interfejsu stanowi *Fabric-CA* (*Certificate Authority*). Zapewnia on:

* Rejestrowanie oraz definiowanie tożsamości użytkowników
* Wydawanie certyfikatów
* Weryfikacje certyfikatów
* Odwołanie oraz odnawianie certyfikatów

Innym interfejsem nie wchodzącym w skład *Hyperledger Fabric* jest *Facebook* *Social Graph (Graph API)*, pochodzący z platformy *Meta for Developers*. Jest to interfejs programowania aplikacji oparty na protokole komunikacji HTTP, korzystający z rozwiązania architektonicznego REST, który umożliwia pobieranie danych z platformy Facebook. Wykorzystując *Graph API* możliwe jest tworzenie zapytań oraz przesyłanie treści i zarządzanie nią. Informacje, które umieszczone są w serwisie społecznościowym reprezentowane są jako:

* węzły,
* krawędzie,
* pola

Aby otrzymać dane o wybranym obiekcie należy użyć węzłów, które posiadają swój unikalny identyfikator. Przykładami węzłów mogą być takie obiekty jak osoby, strony, grupy, posty oraz komentarze.

Pola zawierają szczegółowe dane o węzłach. Dla obiektu osoba wyszczególnimy np. pola takie jak: id, imię, nazwisko, email.

Obiekty (węzły) mogą posiadać między sobą relacje, zatem połączone są krawędziami. Krawędzie przechowują dane świadczące o istniejącym połączeniu. Dla przykładu obiekt osoba udostępnia post zawierający hiperłącze do swojego wniosku o pożyczkę. W tym przypadku istnieje krawędź łączące te obiekty, która zawiera np. takie dane jak data opublikowania posta.

Wszystkie z powyższych składowych interfejsu stanowią punkty końcowe (ang. *endpoints*), które wymagają żeton dostępu (ang. *access token*). Mają one za zadanie umożliwić dostęp do informacji użytkownik. Przykładowo użytkownik logując się za pomocą swojego profilu *Facebook* oraz zezwalając na udostępnienie aplikacji swojego adresu email daje dostęp do odczytu tych danych. *Access token* zawiera takie informacje jak nazwę aplikacji, nazwę użytkownika, status, czas wygaśnięcia (żetony tracą swoją ważność po upłynięciu około godziny).

Rysunek 41. Informacje zawarte w *access token*

Obraz zawierający tekst

Opis wygenerowany automatycznie

Źródło: opracowanie własne

Po otrzymaniu *access token* w zależności od przyznanych uprawnień posiadamy dostęp do danych użytkownika

Rysunek 42. Przykładowe żądanie GET w *Graph API*

Obraz zawierający tekst

Opis wygenerowany automatycznie

Źródło: Opracowanie własne

## Mechanizmy bezpieczeństwa

System informatyczny jest bezpieczny, jeśli jego użytkownik może na nim polegać, a zainstalowane oprogramowanie działa zgodnie ze swoją specyfikacją.. Aby zapewnić bezpieczeństwo systemu informatycznego zalecane jest stosowanie pewnych mechanizmów bezpieczeństwa, mających na celu zapewnienie poufność, nienaruszalność oraz dostępność informacji. Należy zniwelować prawdopodobieństwo na wystąpienie niepożądanych zdarzeń takich jak nieuprawniony dostęp do danych, czy nieautoryzowane modyfikacje. W tym celu w systemie zostały skonfigurowane następujące mechanizmy bezpieczeństwa:

1. Wykorzystanie funkcji skrótu w celu powiązania ze sobą bloków zawierających transakcje.
2. Transakcje wykonywane przez *smart-contract*
3. Algorytm konsensusu
4. Mechanizm kontroli prywatności skonfigurowany poprzez Hyperledger
5. Integracja konta z serwisami społecznościowymi (?)

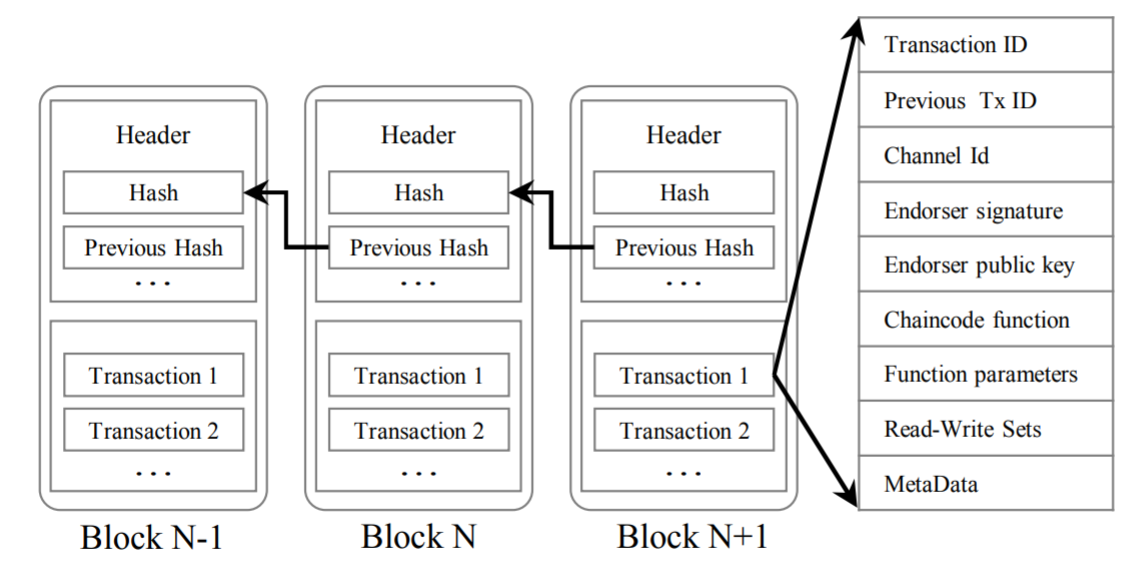
### Mechanizm bezpieczeństwa zabezpieczający transakcje

Kluczowym zadaniem systemu jest zabezpieczenie procesu obsługi pożyczki. Transakcja ma być całkowicie bezpieczna oraz poufna. W tym celu należy zagwarantować, aby żadna nieuprawiona osoba nie miała możliwości na fałszowanie oraz manipulowanie danymi zawartymi w transakcji. Wszystkie próby nieautoryzowanych transakcji muszą zostać wykryte, a następnie odrzucone. Udzielona pożyczka ma trafić bezpośrednio od pożyczkodawcy do pożyczkobiorcy pomijając strony trzecie. Ponadto ustalone uprzednio warunki przez obie strony nie mogą zostać zmienione. Kwota pożyczki nie może ulec zmianie po wykonaniu transakcji, a pożyczkobiorca ma otrzymać jej całość.

Wykorzystanie funkcji skrótu celem powiązania ze sobą bloków zawierających transakcje.

Aby spełnić powyżej wymienione wymagania transakcje umieszczane są w łańcuchu bloków. Ogólna architektura i działanie zostało opisane w podrozdziale 1.2.1. oraz rys. 3, jednakże każda technologia wykorzystuje inne koncepcje, różniące się zawartością bloków oraz ich sposobem połączenia. Występują różne funkcje skrótu, a zatem odmienny jest sposób generowania *hash*, stanowiący powiązanie pomiędzy blokami.

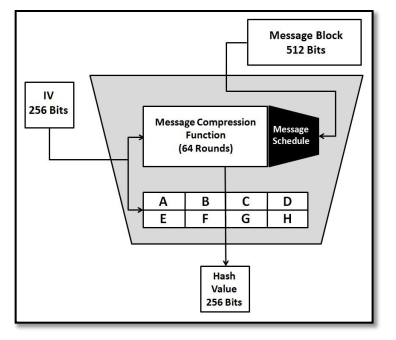
Rysunek 43. Struktura bloków Hyperledger Fabric



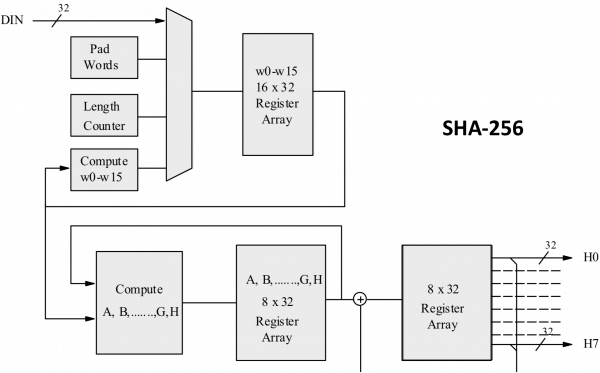
Źródło: Ence Zhou, Haoli Sun, Bingfeng Pi, Jun Sun, Kazuhiro Yamashita, Yoshihide Nomura „Ledgerdata Refiner: A Powerful Ledger Data Query Platform for Hyperledger Fabric”

Platforma Hyperledger Fabric w tym celu wykorzystuje funkcję skrótu SHA256, która stanowi jeden z sześciu algorytmów z rodziny jednokierunkowych *Secure Hash Algorithm 2* (SHA-2). Algorytm jest używany do ochrony kryptograficznej. Jest to proces, w którym zostaje wygenerowany niepowtarzalny zestaw danych o stałym rozmiarze 256 bitów dla dowolnego ciągu znaków.

Rysunek 44. Funkcja skrótu SHA256



Źródłó: Raful P. Naik, „*Optimising the SHA256 Hashing Algorithm for Faster and More Efficient Bitcoin Mining”*

Źródło: <https://www.cast-inc.com/security/encryption-primitives/sha-256/>

Należy mieć na uwadze, iż algorytm ten jest deterministyczny, co oznacza, że dopóki dane wejściowe nie ulegną zmianie, algorytm zwróci zawsze ten sam *hash*. Ich jakakolwiek zmiana spowoduje, że SHA256 wygeneruje kompletnie odmienny wynik. Funkcja jest jednokierunkowa oraz nieodwracalna, zatem nie jest możliwe odkodowanie danych wejściowych.

Tabela 21. Tabela przedstawiająca działanie algorytmu SHA-256

|  |  |
| --- | --- |
| Dane wejściowe | Dane wyjściowe (SHA-256) |
| Jakub Kupis | dd4c75f5c175849406671b2195c49e3e2ef6f1be885e6d2abb494f42f5172a5c |
| jakub Kupis | c8bd010587a36cf766cdc65b08711e4ebecfd6a5ca09b77d7f6a9ead60b41723 |

źródło: opracowanie własne

System informatyczny, wykorzystujący technologie łańcucha bloków posiada cechy:

1. Niezmienność i niezaprzeczalność – transakcja wprowadzana do łańcucha bloków jest niemożliwa do edycji, z racji tego, że każdy węzeł uczestniczący otrzymuje jej kopie
2. Odporność na fałszerstwa – niemożliwe jest dokonanie zmian danych zawartych w bloku z uwagi na zastosowane funkcje kryptograficzne
3. Odporny na usterki – usterka pojedynczego węzła nie powinna wpływać na działanie całej sieci, ponieważ transakcje zawarte w obrębie pojedynczych bloków pozostają niezmienne w przypadku utraty połączenia sieciowego
4. Gotowy do audytu – wszystkie transakcje, które miały miejsce przechowywane są w niezmienionej postaci

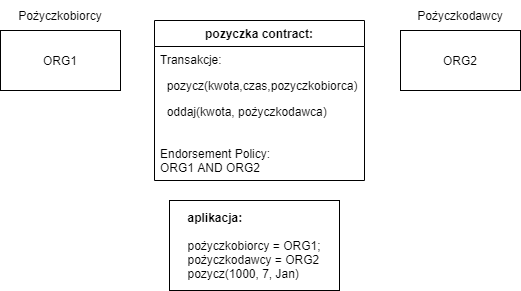
Transakcje wykonywane przez smart-contract / Algorytm konsensusu / Mechanizm kontroli prywatności skonfigurowany poprzez Hyperledger

Kolejnym mechanizmem bezpieczeństwa zabezpieczającym proces obsługi pożyczki jest zastosowanie w systemie inteligentnych umów z ang. *smart contract*. Czym są inteligentne umowy zostało opisane w podrozdziale 1.2.1. Platforma *Hyperledger Fabric* oferuje swoje rozwiązanie, które w terminologii określane jest jako *chaincode*.

W inteligentnych umowach definiowana jest logika biznesowa, która generuje nowe zdarzenia dodawane do rejestru, znajdującego się w łańcuchu bloków. Aby doszło do transakcji należy zdefiniować zbiór warunków, które są niezbędne do jej wykonania. Umowy zawarte w *chaincode* stanowią model biznesowy, regulujący interakcje pomiędzy stronami transakcji. Wykorzystując sieć *blockchain* kod przekształcany jest w wykonywalny program.

W poniższym diagramie możemy wyróżnić dwie organizacje, *ORG1* i *ORG2* oraz zawarty pomiędzy nimi kontrakt o nazwie *pozyczka*. Członkowie organizacji mogą wywołać uruchomienie kontraktu, który umożliwia zarówno udzielenie pożyczki jak i jej zwrot. Każdy kontrakt umieszczony w *chaincode* powiązany jest z polityką aprobaty z ang. *endorsement policy*. Jest to bardzo ważny element modelu, ponieważ definiuje, które z organizacji muszą potwierdzić wykonanie transakcji, aby ta została zatwierdzona. W przypadku poniższego modelu biznesowego, aby transakcja doszła do skutku wymagane jest potwierdzenie ze strony organizacji *ORG1* oraz *ORG2*. Po spełnieniu warunków organizacje osiągają konsensus, a następnie transakcja zostaje dodana do łańcucha bloków. Brak potwierdzenia z jakiejkolwiek strony skutkuje uznaniem transakcji za nieważną.

Rysunek 45. Diagram przedstawiający przykładowy model biznesowy zawarty w *Chaincode*



Źródło: opracowanie własne

Inteligentne umowy zapewniają bezpieczny przebieg procesu obsługi pożyczki, ponieważ są:

1. Niezmienne - kodu zawartego w umowie nie można zmienić po jego uruchomieniu, przez co jest on odporny na modyfikacje.
2. Zdecentralizowane - kontrakty nie znajdują się na scentralizowanym serwerze tylko podlegają dystrybucji na wszystkie węzły obecne w sieci, dzięki temu niemożliwa jest manipulacja danymi ze strony jednostki centralnej. Ponadto dzięki rozproszeniu system odporny jest na przerwy w działaniu serwerów.
3. Deterministyczne – wykonują jedynie te działania, które zostały zaimplementowane, po spełnieniu zapisanych w nich wymagań.
4. Autonomiczne – w momencie spełnienia warunków umowy transakcja zostaje wykonana automatycznie i jest zapisywana w łańcuchu bloków
5. Niewymagające zaufania – z racji właściwości łańcucha bloków możemy dokonywać transakcji bez konieczności korzystania z pośredników. Co więcej nie ma konieczności darzenia zaufaniem nawet drugiej strony transakcji

### Mechanizm bezpieczeństwa zabezpieczający logowanie

Proces logowania jest zabezpieczany poprzez wykorzystanie mechanizmu uwierzytelnienia dwuetapowego za pomocą narzędzia *Google Authenticator*. Jest to narzędzie wprowadzające dodatkowe warstwy ochronne. Wprowadzenie uwierzytelnienia dwuetapowego ma za zadanie zabezpieczenie poświadczeń użytkownika. Aplikacja wykorzystuje algorytm TOTP (*Time-based-one-time-password)*, który generuje jednorazowy kod o skończonej żywotności – 30 sekund. Po upływie czasu generowany jest nowy ciąg liczb, stanowiący kod dostępu.

Pierwszym etapem dostępu do konta jest logowanie za pomocą loginu oraz hasła. Drugi etap stanowi wprowadzenie jednorazowego kodu autoryzacji, który został wygenerowany poprzez aplikację mobilną w telefonie.

W przypadku sytuacji, kiedy osoba nieupoważniona poznała nasz login oraz hasło, nie będzie w stanie otrzymać dostępu do zasobów bez fizycznego dostępu do naszego telefonu i użycia aplikacji.

O ile sam w sobie Google Authenticator wprowadza bezpośrednio drugą warstwę ochrony danych, tak w wielu przypadkach wprowadza kolejne pośrednio. Wynika to z faktu, iż aby wygenerować kod, należy uzyskać dostęp do telefonu, który ma uruchomione następne mechanizmy bezpieczeństwa takie jak PIN, wzór odblokowania, odcisk palca, *FaceID.*

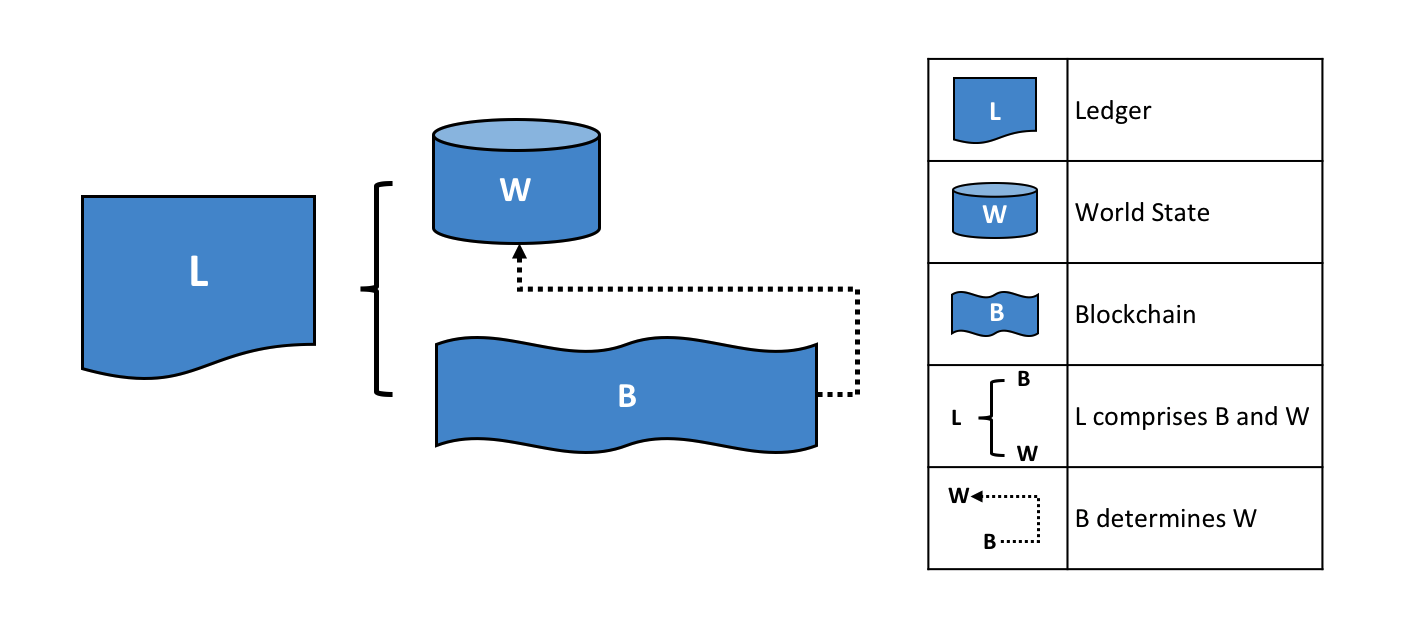
Aby zaimplementować mechanizm w systemie należy:

1. Zaciągnąć zależności *Maven* do projektu: *totp, commons-codec*
2. Zaimplementować metody z wykorzystaniem powyższych bibliotek
3. Pobrać oraz zainstalować aplikacje *Google Authenticator* na smartphone
4. Dodać nowe konto do aplikacji za pomocą wygenerowanego sekretnego klucza

## Logiczny model danych

### Model bazy danych

Platforma Hyperledger Fabric przechowuje dane w rozproszonym rejestrze zwanym *Ledger*, który stanowi połączenie struktur danych *World State* oraz *Blockchain.* Rejestruje on fakty dotyczące aktualnego stanu obiektów biznesowych oraz historie zrealizowanych transakcji, które doprowadziły do obecnego stanu. Każdy z węzłów znajdujących się w zdecentralizowanej sieci posiada własną kopie *Ledger*.

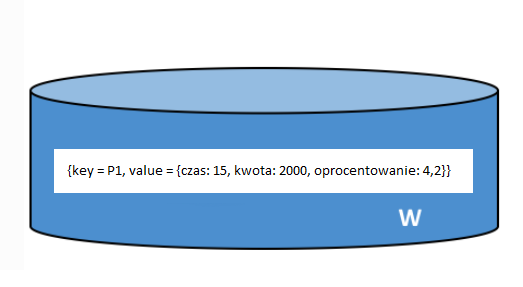
**

Rysunek 46. Model struktury danych Hyperledger Fabric

Źródło: <https://hyperledger-fabric.readthedocs.io/en/latest/ledger/ledger.html>

*World State* gromadzi dane odnośnie wartości obecnego stanu obiektów biznesowych, dzięki czemu nie ma konieczności przechodzenia przez cały rejestr transakcji (np. w celu obliczenia ilości udzielonych pożyczek). Stany w tej strukturze danych regularne ulegają zmianie, z powodu czego przechowywane są na zasadzie pary klucz-wartość.

Rysunek 47. Przykładowe wartości stanu pożyczki P1 w strukturze danych *World State*

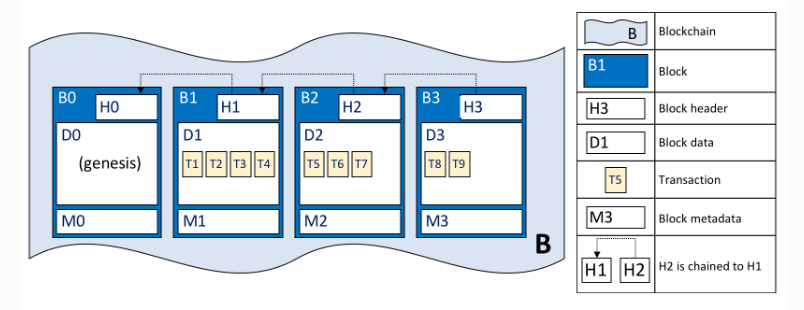


Źródło: opracowanie własne na podstawie <https://hyperledger-fabric.readthedocs.io/en/latest/ledger/ledger.html>

### Model struktur danych przechowywanych poza bazą danych

*Blockchain* stanowi rejestr historii wszystkich transakcji, mających wpływ na obecny stan *World State*. Transakcje znajdują się w blokach, które zostają dołączane do łańcucha. W przeciwieństwie do *World State*, strukturajest niezmienna a rekordy, zawierające transakcje ułożone są sekwencyjnie.

Rysunek 48. Model struktury danych zawartych w *Blockchain*



Źródło: <https://hyperledger-fabric.readthedocs.io/en/latest/ledger/ledger.html>

## Algorytmy

W systemie zastosowano następujące algorytmy:

1. Przebieg transakcji w systemie
2. Weryfikacja z wykorzystaniem mediów społecznościowych
3. Przyznawanie punktów *rating* na podstawie modelu wykorzystującego media społecznościowe

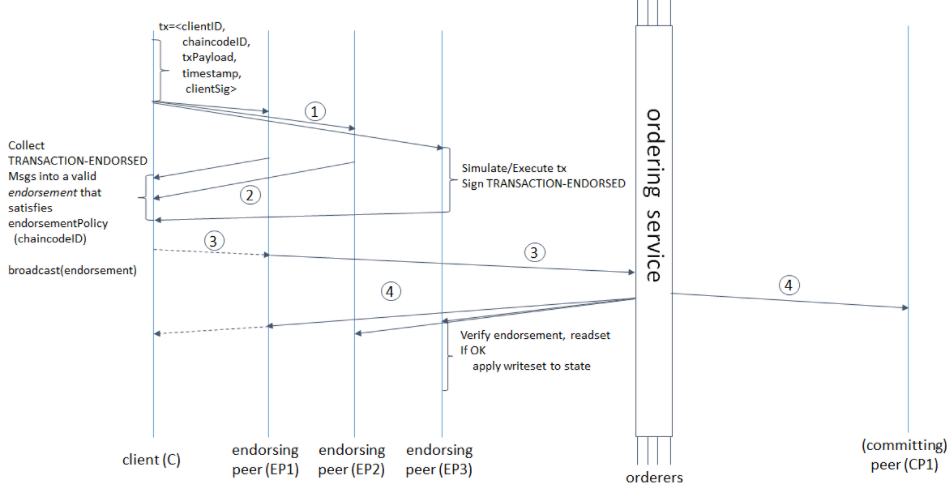
Algorytm przebiegu transakcji został przedstawiony jako diagram sekwencji, który jest implementacją *Hyperledger Fabric*. Algorytm nr 2 i 3 wykorzystuje media społecznościowe, których integracja zapewniona jest poprzez narzędzia pochodzące z platformy *Meta for Developers*. Działanie owych algorytmów oparte jest na oficjalnych patentach, jednakże zostały dostosowane pod potrzeby opracowywanego systemu informatycznego.

### Algorytm przebiegu transakcji

Przepływ transakcji:

1. Klient tworzy transakcje, a następnie przesyła do *endorsing peer*
2. *Endorsing peer* symuluje transakcje i podpisuje sygnaturą
3. Klient otrzymuje adnotację transakcji i nadaje ją do *ordering service*
4. *Ordering service* weryfikuje transakcje pod względem certyfikatów tożsamości, tworzy bloki, a następnie dostarcza do *comitting peer*

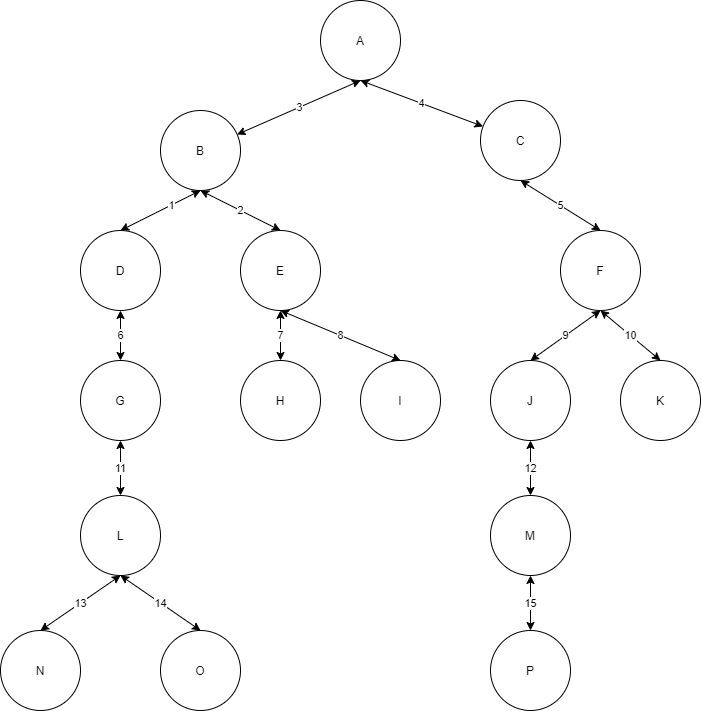
Rysunek 49. Przykładowy przepływ transakcji

  
źródło: <https://hyperledger-fabric.readthedocs.io/en/release-1.4/arch-deep-dive.html>

### Algorytm weryfikacji przyznania pożyczki

W celu zmniejszenia ryzyka i ustaleniu wiarygodności pożyczkobiorcy przy procesie obsługi pożyczki skorzystano z algorytmu weryfikacyjnego, opatentowanego przez *Facebook* w 2004 roku w dokumencie *„Authorization and authentication based on an individual's social network”*, który został zmodyfikowany pod potrzeby opracowywanego systemu. Algorytm w dużej mierze opiera się o teorii grafów i sieci. Wykorzystując serwis społecznościowy, studenci będący użytkownikami serwisów społecznościowych są ze sobą powiązani, tworząc sieć społecznościową.

Rysunek . Reprezentacja grafowa przykładowej sieci społecznościowej



Źródło: Opracowanie własne

Rysunek nr 48 przedstawia graf G = <W, K, R>, gdzie:

* W – węzły
* K - krawędzie
* R ⊂ W × K × W - relacje

Reprezentuje on przykładową sieć społecznościową. Każdy z jej członków będzie stanowić węzły (A-P), natomiast krawędzie (1-15) łączące węzły są to relacje zawarte między członkami sieci. Minimalna liczba kroków potrzebna do przebycia pomiędzy węzłami definiuje stopień separacji, który stanowi miarę pokrewieństwa pomiędzy członkami. Stopień separacji jest wartością indywidualną dla każdego z węzłów. Wraz ze wzrostem stopnia separacji liczba węzłów znajdujących się w sieci zaczyna rosnąc w coraz to szybszym tempie.

Zakładając, że sieć skupiona jest na węźle B ma on następujące stopnie separacji:

1 stopień – D, E, A

2 stopnie – G , H, I, C

3 stopnie – L, F

itd.

Algorytm sporządza dla każdego z węzłów 3 rodzaje list, na podstawie których podejmowana jest decyzja o przyznaniu pożyczki:

* Czarna lista - tworzona i aktualizowana jest na podstawie działań, jakie podejmują powiązane węzły. Przykładowo, jeśli węzeł B otrzyma niepożądane komunikaty (np. SPAM, hiperłącza zawierające szkodliwe oprogramowanie) lub napotka nieodpowiednie treści (znajdujące się w poniższym wykazie) opublikowane w sieci przez węzeł M i G, zostają one dodane do czarnej listy węzła B. Węzeł M i G nie jest uprawniony do dokonywania transakcji z węzłem B. W przypadku dokonania takich działań poprzez osoby nie znajdujące się w sieci węzła B, do czarnej listy dodawane są dane umożliwiające identyfikację np. adres e-mail.
* Szara lista - wywodzi się z listy czarnej. W jej skład wchodzą węzły, które są odseparowane jeden stopień od węzłów znajdujących się na czarnej liście węzła B. Kontynuując powyższy przykład, jeśli węzeł C oraz G został umieszczony na czarnej liście, do szarej listy węzła B zostaną dodane węzły D, L oraz J, P.
* Biała lista – znajdują się na niej pozostali członkowie sieci niewchodzący w skład listy czarnej i szarej z pewnym wyjątkiem. Węzły dodawane są do listy na podstawie algorytmu przeszukiwania wszerz (*breadth-first search*, BFS). W przypadku napotkania węzła nie należącego do białej listy, algorytm przeszukiwania odcina daną gałęź. Przykładowo napotykając węzeł D, należący do szarej listy węzła B, węzły G, L, N, O nie są dalej brane pod uwagę. Zatem na białej liście węzła B znajdować będą się węzły: A, E, C, H, I, F, K

Tabela 22. Wykaz treści niepożądanych kwalifikujących do umieszczenia użytkownika na czarnej liście

|  |  |
| --- | --- |
| **Lp.** | **Treść** |
| 1. | treści obrazujące przemoc, obrażenia fizyczne, okrucieństwo bądź śmierć, |
| 2. | treści nawołujące do samookolaczeń, samobójstw, zachowań szkodliwych dla zdrowia, |
| 3. | treści dyskryminacyjne, promujące wrogość, nienawiść, rasizm |
| 4. | treści pornograficzne dostępne bez żadnego ostrzeżenia (w szczególności pornografia dziecięca) |
| 5. | relacje na żywo prezentujące zachowania określane i postrzegane jako patologiczne (tzw. *patostream*) |

Źródło: opracowanie własne na podstawie <https://www.gov.pl/web/niezagubdzieckawsieci/szkodliwe-tresci-w-internecie-nie-akceptuje-reaguje>

Algorytm weryfikacji zaczyna się żądaniem węzła B o otrzymanie pożyczki. Następnie otrzymywana jest reprezentacja jego sieci społecznościowej, z której kolejno tworzone są poszczególne listy. Po otrzymaniu list określony zostaje obliczony ważony średni *rating*. W zależności od miary pokrewieństwa pomiędzy węzłami przyjmujemy odpowiednią wagę, według następującego wzoru:

gdzie:

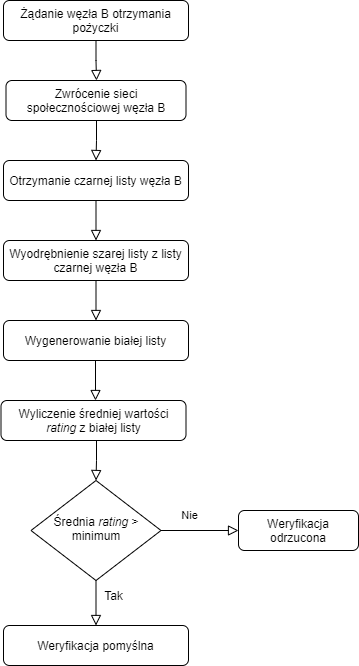
W – waga,

L - liczba stopni separacji pomiędzy węzłami,

M – maksymalna liczba stopnia separacji danego węzła

Finalnie na podstawie wyliczonej średniej ważonej *rating* zostaje podejmowana decyzja o przyznaniu pożyczki.

Rysunek . Algorytm weryfikacji przyznania pożyczki



Źródło: Opracowanie własne na podstawie *„Authorization and authentication based on an individual's social network”*

Tak jak zostało to zdefiniowane w podrozdziale 2.7.4., pożyczkodawca dokonuje decyzji komu zostanie udzielona pożyczka wybierając spośród dostępnych ofert (wystawionych wniosków). Z racji tego iż, powyższy diagram algorytmu weryfikacji przyznania pożyczki przedstawiony został od strony pożyczkobiorcy, nie został uwzględniony w nim przypadek, w którym pożyczkodawca chciałby udzielić pożyczki węzłowi B, jednakże ten znajdowałby się na jego czarnej liście. W tym przypadku nie jest możliwe udzielenie pożyczki.

### Algorytm przyznawania punktów *rating*

Kolejnym z algorytmów zastosowanych w systemie jest określenie początkowych punktów *rating* użytkownika, wykorzystując media społecznościowe. Algorytm bazuje na patencie „*Socioeconomic group classification based on user features*” opracowanego przez *Facebook*.

Użytkownik poproszony jest o zalogowanie się do systemu za pomocą wybranego serwisu społecznościowego i w ten sposób następuje integracja profilu z zewnętrznym systemem.

Rysunek 52. Integracja z profilem społecznościowym *Facebook*

Obraz zawierający tekst

Opis wygenerowany automatycznie

źródło: opracowanie własne

Po dokonaniu integracji, system otrzymuje dostęp do danych udostępnionych przez użytkownika w serwisie społecznościowym. Ponadto uwzględniane są również wszelkie działania jakie podjął użytkownik. Dzięki nim jesteśmy w stanie w pewnym stopniu oszacować do jakiej klasy społecznej należy użytkownik oraz jakie generuje przychody, co przekłada się na maksymalna kwotę pożyczki, jaką może otrzymać oraz jej prawdopodobieństwo spłaty.

Tabela 23. Dane mogące mieć wpływ na oszacowanie należności do danej klasy społecznej

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Demografia | Posiadane urządzenia podłączone do sieci | Użycie Internetu | Historia podróży | Gospodarstwo domowe |
| Wiek,  Płeć,  Rejon Zamieszkania, Wykształcenie | Liczba urządzeń,  Typ urządzeń,  Model urządzenia | Szybkość łącza, Zużycie łącza, Pora dnia użycia | Kierunek podróży,  Typ podróży, Rodzaj komunikacji | Dom (lokalizacja, wyposażenie),  Samochód |

źródło: opracowanie własne

Z racji tego, iż opracowywany system jest przeznaczony dla studentów dane demograficzne nie będą uwzględniane z tego względu, że w większości przypadków będą w podobnym wieku oraz będą mieć ten sam poziom wykształcenia.

Na potrzeby użycia algorytmu zostaną zadeklarowane trzy klasy (L – *lower*, M – *middle*, U – *upper*), do jakich zostaną przydzieleni użytkownicy. Każdej klasie został przypisany następujący *rating*:

Tabela 24. Punkty *rating* przyznane klasom

|  |  |  |  |
| --- | --- | --- | --- |
| **Klasa** | L | M | U |
| ***Rating*** | 30 | 100 | 150 |

źródło: opracowanie własne

Algorytm działa na zasadzie wykorzystania drzewa decyzyjnego… . W przypadku braku lub niewystarczających danych nie zostają przyznane żadne punkty. Oczywiście istnieje inny sposób zwiększenia swojej zdolności pożyczkowej – poprzez regularne, terminowe spłaty (patrz tabela nr 19.) #TODO drzewo decyzyjne

Ilość urzadzen -> rok wydania -> iOS / Android / Windows

Internet -> szybkość lacza

Ilosc podrozy -> kierunek podrozy (Ameryka, Europa, Azja)

Dom -> akademik / mieszkanie

## Wnioski

W tym rozdziale dokonano podziału systemu na warstwy, zawierające pewne komponenty, które realizują odmienne zadania. Zostały przedstawione zarówno interfejsy oparte na *Hyperledger Fabric* jak i *Meta for Developers*. Bezpieczeństwo systemu zostało zapewnione poprzez zastosowanie licznych mechanizmów zabezpieczenia dotyczących procesu transakcji (szyfrowanie, *smart contract*, konsensus, polityka prywatności) oraz logowania. Zdefiniowano sposób przechowywania danych w systemie. Przedstawiono model struktury danych oraz jego podział. Na koniec zostało opisane, w jaki sposób przebiega transakcja oraz jak odbywają się weryfikację za pomocą mediów społecznościowych.

# IMPLEMENTaCJA I TESTY systemu informatycznego

## Wprowadzenie

W niniejszym rozdziale został opisany jeden z etapów cyklu życia systemu, polegający na procesie tworzenia kodu źródłowego w konkretnym środowisku programistycznym, dostosowaniu oprogramowania do postawionych wymagań oraz uruchomieniu systemu informatycznego. Opis systemu zdefiniowany w poprzednich rozdziałach został przekształcony na obiekt fizyczny, jakim jest system informatyczny. Implementacja ma na celu przeprowadzenie wdrożenia oraz integracje systemu z procesami zachodzącymi w organizacji. Końcowym etapem było przeprowadzenie testów w celu weryfikacji poprawności działania systemu informatycznego.

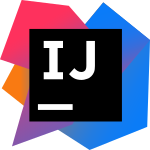
## Charakterystyka środowiska wytwórczego

Na środowisko wytwórcze składa się zbiór narzędzi, języków programowania, baz danych, bibliotek itd. wykorzystanych do wytwarzania oprogramowania. Podczas tego procesu skorzystano wyłącznie z oprogramowania typu *open source*. Główną część środowiska wytwórczego stanowi platforma *Hyperledger Fabric* oraz technologie, przez nią wykorzystywane (*Java*, *CouchDB*). Za konteneryzacje sieci odpowiada *Docker*. W celu integracji mediów społecznościowych z systemem skorzystano z narzędzia *Meta for Developers*. Cały proces wytwarzania oprogramowania został kontrolowany poprzez system *Git*.

### Narzędzia / środowisko programistyczne



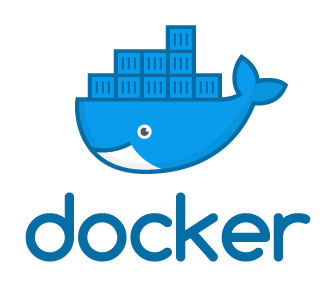
**Hyperledger Fabric** jest to platforma typu *open source* prowadzona przez Linux Foundation, która obsługuje technologię rozproszonego rejestru (ang. *Distributed Ledger Technology, DLT*). Została wyposażona w zaawansowane mechanizmy kontroli ochrony prywatności. *Blockchain* zbudowany poprzez Hyperledger Fabric jest siecią z uprawnieniami, co oznacza, iż jej użytkownicy mogą współużytkować jedynie te dane, do których mają zapewniony dostęp. Modułowa i konfigurowalna architektura korzysta z gotowych komponentów, zapewniając innowacyjność oraz wszechstronność, co przekłada się na szerokie możliwości zastosowań. Jako pierwsza z platform DLT umożliwiła tworzenie inteligentnych kontraktów w językach ogólnego przeznaczenia takich jak Java, Go oraz Node.js, dzięki czemu programiści są w stanie implementować *smart contracts* bez konieczności nauki nowego języka programowania.



**IntelIJ IDEA** jest to zintegrowane środowisko programistyczne, którego autorem jest firma *JetBrains*. Jego pierwsza wersja pojawiła się w 2001 roku. Dostępne jest zarówno dla rodziny systemów operacyjnych Windows, Linux jak i macOS. Występują dwie wersje licencji: *Community Edition: Apache 2.0* oraz *Ultimate Edition: Trialware*.

IDEA umożliwia tworzenie aplikacji między innymi w językach programowania takich jak *Java*, *Kotlin*, *Scala*, *Groovy*. Obsługuje liczne technologie np. *Spring*, *Gradle*, *Maven*. Wspiera następujące bazy danych: *PostgreSQL, SQLite, Oracle, Microsoft SQL*. Zapewnia również inne funkcjonalności takie jak integracja z systemami kontroli wersji np. *Git*

Kodowanie w owym środowisku jest ułatwione poprzez funkcję uzupełniania kodu, nawigację, umożliwiającą przechodzenie do deklaracji w kodzie, oraz jego refaktoryzacje. *IntellIJ* obsługuje również wtyczki (ang. *plugins*) za pomocą których możemy zapewnić dodatkowe funkcjonalności. Na dzień dzisiejszy repozytorium liczy ponad 3000 wtyczek.



**Docker** jest to platforma *open source*, służąca do tworzenia, wdrażania oraz zarządzana aplikacjami kontenerowymi. Został napisany w języku programowania *Go* oraz korzysta z funkcji, pochodzących z jądra systemu operacyjnego Linux. Kontenery są to wykonywalne komponenty łączące kod źródłowy aplikacji z zależnościami i bibliotekami systemu operacyjnego, które są wymagane do uruchomienia kodu w dowolnym środowisku. Docker umożliwia oddzielenie aplikacji od infrastruktury, co znacznie przyśpiesza proces wytwarzana oprogramowania. Wykorzystując kontenery programiści są w stanie dostarczać rozproszone aplikacje.



**Gradle** jest narzędziem *open source* służącym do budowania projektów na licencji *Apache License 2.0*. Jego pierwsze wydanie pojawiło się w 2008 roku. Pozwala na zautomatyzowanie procesu budowania aplikacji przy użyciu języka domenowego DSL (ang. *Domain Specific Language*). Obsługuje takie języki jak *Java, Kotlin, Groovy, Scala, C/C*++ oraz *JavaScript*. *IntelliJ IDEA* domyślnie wspiera projekty budowane przy pomocy *Gradle*. Umożliwia między innymi:

* Kompilowanie kodu źródłowego
* Tworzenie pliku ze skompilowanymi klasami (np. JAR – Java Archive)
* Zarządzanie zależnościami, wtyczkami oraz bibliotekami
* Testowanie oraz generowanie raportów

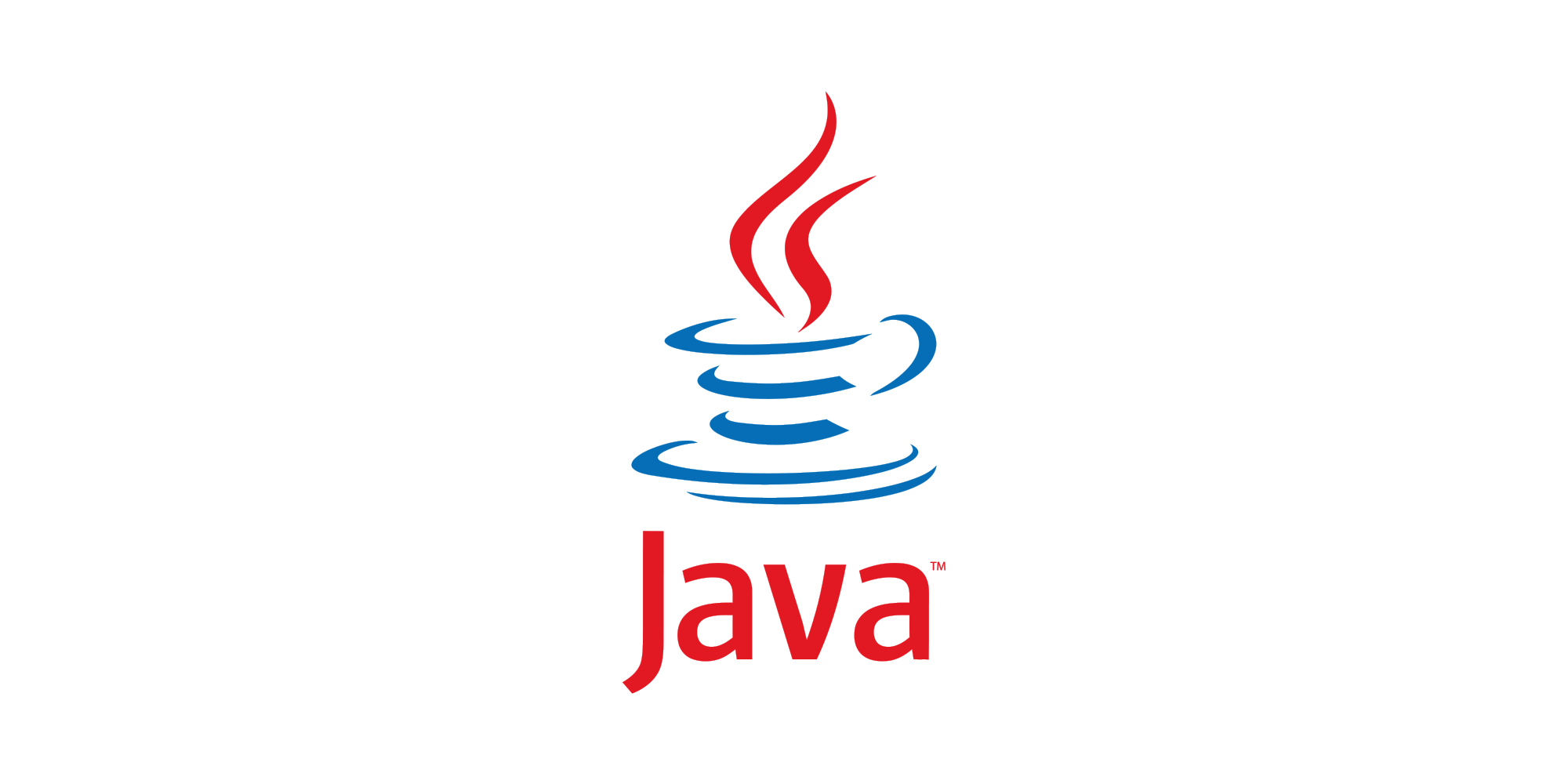


***Meta for Developers*** jest to zbiór platform, umożliwiający połączenie klientów z aplikacją wykorzystując ich produkty takie jak:

* *Messenger Platform*
* *Facebook Platform*
* *Instagram Platform*
* *WhatsApp Business API*

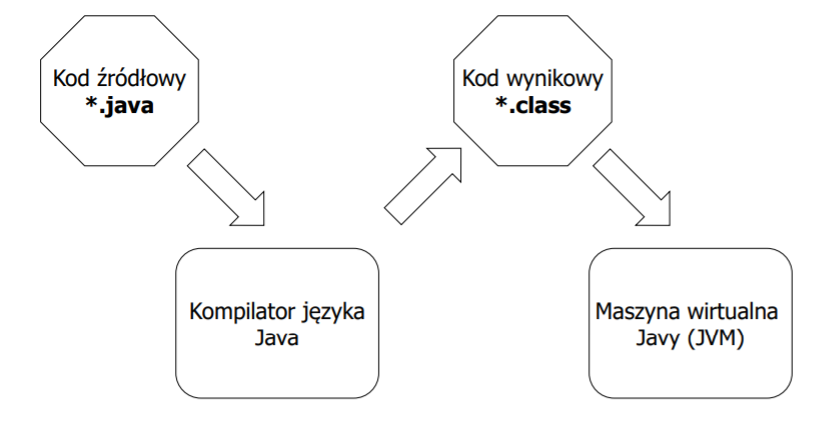
**Facebook SDK** jest zestawem narzędzi pochodzącym z *Meta for Developers* zawierającym między innymi *Facebook Login*, *Graph API* i *Social Plugins*. Za pomocą *Facebook Login* możliwa jest integracja aplikacji z serwisem społecznościowym. Gwarantuje szybki i wygodny sposób logowania do aplikacji, zapewaniający prywatność oraz bezpieczeństwo. Ponadto umożliwia uzyskanie uprawnień do dostępu danych umieszczonych w serwisie społecznościowym. Kolejnym wykorzystanym narzędziem pochodzącym również z Facebook SDK jest *Facebook Social Graph (Graph API*), które zostało opisane w podrozdziale 4.3.5. Wymienione narzędzia zostały wykorzystane w algorytmie weryfikacji przyznania pożyczki (rozdział 4.6.2.) oraz algorytmie przydzielania punktów *rating* (rozdział 4.6.1.).

### Języki programowania



**Java** to wysokopoziomowy, dynamiczny język programowania ogólnego zastosowania, który został opracowany przez firmę Sun Microsystems. Jest językiem tworzenia programów źródłowych kompilowanych do kodu bajtowego, który jest intepretowany poprzez wieloplatformową maszynę wirtualną Java (*Java Virutal Machine*, JVM). Z założenia pozwala to na uruchomienie aplikacji na różnych maszynach.

Rysunek . Kompilacja programów w języku Java

  
źródło: dr hab. inż. Marek Wojciechowski, „Język programowania Java”

Język programowania Java ma szerokie zastosowanie szerokie zastosowanie w tworzeniu oprogramowania. Najczęstszym wykorzystaniem są systemy aplikacji internetowych. Sprawdza się również w tworzeniu oprogramowania desktopowego. Ponadto system Android działa w środowisku Javy i również jest możliwość tworzenia aplikacji mobilnych w tym języku. W przypadku opracowywanego systemu zarówno aplikacja jak i *chaincode* została stworzona w tym języku.

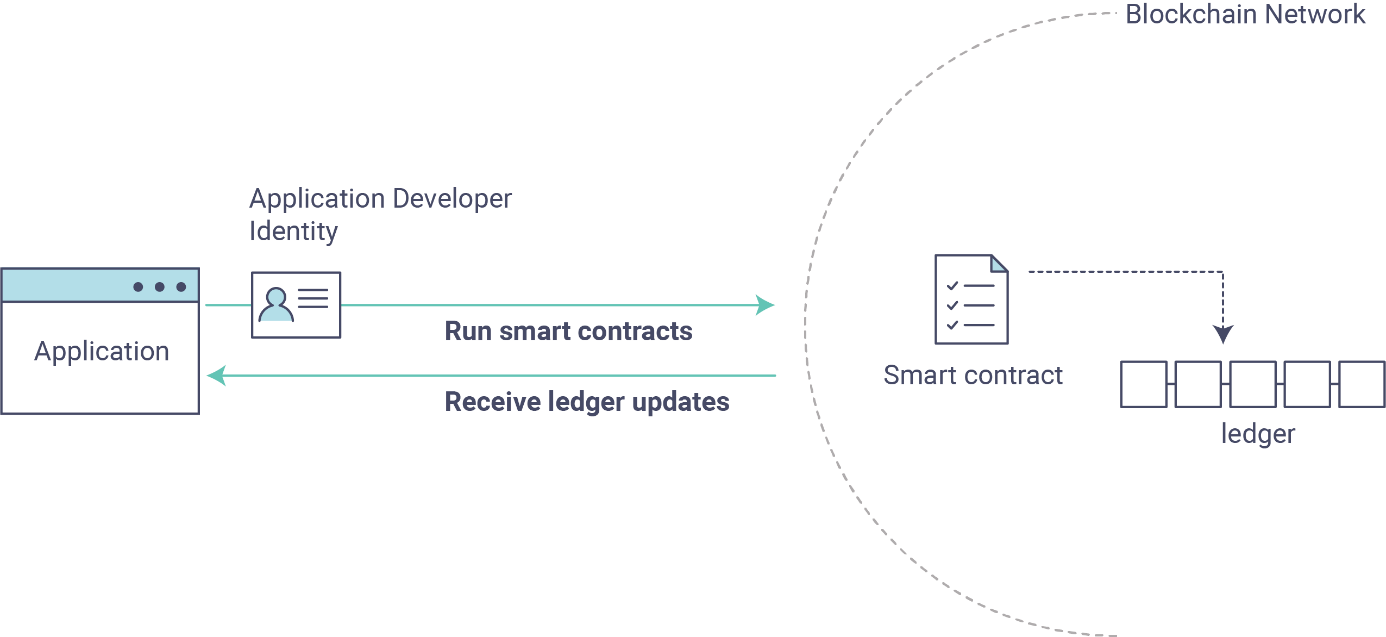
Cechy języka programowania Java jakie możemy wyróżnić to:

* Wydajny
* Kompilowany i interpretowany
* Zorientowany obiektowo
* Dynamiczny
* Niezależny od architektury

### Szkielet budowy aplikacji // Sposób działania aplikacji // Struktura aplikacji

Aplikacja, składająca się z warstwy prezentacji oraz logiki biznesowej potrzebuje sieci, z którą może wchodzić w interakcje. Za jej pomocą, zostaje wdrożony lub uruchomiony *smart contract* (inaczej *chaincode*), który znajduję się w sieci *blockchain* wraz z rejestrem zwanym *Ledger*. Uruchomienie *smart contract* oznacza dokonanie transakcji zawartych w nim. Wykonane transakcje zapisywane są w rejestrze, a następnie zwracany jest ich status do aplikacji.

Rysunek 54. Szkielet budowy aplikacji



Źródło: <https://hyperledger-fabric.readthedocs.io/en/latest/write_first_app.html>

Rysunek 66. Struktura *chaincode* w środowisku programistycznym IntelliJ IDEA

Obraz zawierający tekst, zrzut ekranu, monitor, czarny

Opis wygenerowany automatycznie

Źródło: Opracowanie własne

Rysunek 67. Struktura aplikacji *Java SDK* w środowisku programistycznym IntelliJ IDEA

Obraz zawierający tekst, monitor, ekran, zrzut ekranu

Opis wygenerowany automatycznie

Źródło: Opracowanie własne

### Biblioteki i komponenty

Opisać syntetycznie w formie wykazu wszystkie gotowe biblioteki i komponenty programistyczne, które zostały wykorzystane do budowy systemu informatycznego.

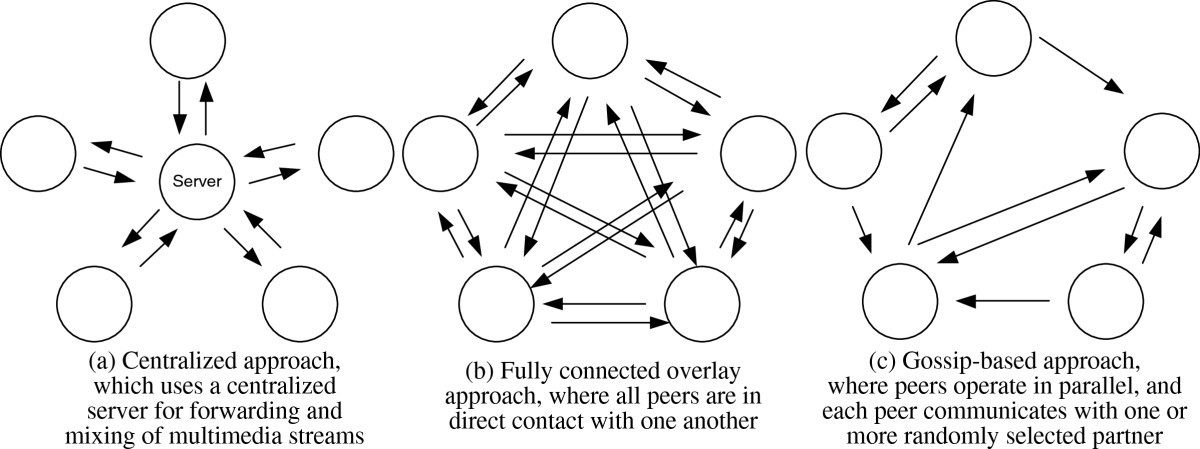
|  |  |  |
| --- | --- | --- |
| **Lp.** | **Nazwa** | **Opis** |
| 1 | Hyperledger Fabric Samples | Stanowi zbiór gotowych aplikacji, obrazów Docker oraz *chaincode*, służących do demonstracji możliwości i lepszego zrozumienia działania technologii oraz testowania sieci. |
| 3 | Hyperledger Docker Image | Jest to obraz |

### Protokoły komunikacyjne// Oprogramowanie pośredniczące // Komunikacja w sieci

Sieć zbudowana poprzez Hyperledger Fabric zapewnia bezpieczeństwo oraz skalowalność poprzez rozdzielenie operacji na rozproszone w niej węzły, które przechowują każdą transakcję, jaka wystąpiła w sieci. W celu zapewnienia integralności oraz spójności danych został wdrożył protokół plotek (z ang. *gossip protocol*). Jest to protokół komunikacji działający na zasadzie P2P.

Przepływ transakcji został opisany w podrozdziale 4.6.1.. W ostatnim etapie, gdzie *orderer service* dostarcza blok do wybranego węzła, *gossip* *protocol* rozpoczyna swoje działanie. Blok ten musi trafić do każdego z węzłów w sieci. Następuje proces zwany „plotkowaniem”. Węzeł, który otrzymał transakcje od *ordering serivce* przesyła ją do innego, losowo wybranego węzła (lub grupie węzłów), który z kolei powtarza te czynność. Proces ten trwa dopóty, dopóki wszystkie węzły otrzymują blok. Każda z przesłanych przez węzły wiadomości jest przez nie podpisywana, w celu zapobiegania przed manipulacją danych czy nieuprawnionym dostępem. Węzły, które z pewnych powodów nie otrzymają danych (np. opóźnienia, błędy sieciowe), po ponownym podłączeniu do sieci, zostaną zsynchronizowane z pozostałymi węzłami, które posiadają brakujące dane.

Rysunek 55. Topologia komunikacji opartej na protokole plotkowania



Źródło: Vincent Wing-Hei Luk, Albert Kai-Sun Wong, Chin-Tau Lea & Robin Wentao Ouyang, „*RRG: redundancy reduced gossip protocol for real-time N-to-N dynamic group communication”*

Zastosowany w sieci *gossip protocol* spełnia 3 podstawowe funkcje:

1. Stale identyfikuje dostępność węzłów oraz wykrywa te, które są niedostępne
2. Rozpowszechnia dane pomiędzy węzłami (a w przypadku węzła, który stracił dostęp i nie otrzymał danych - synchronizuje je).
3. Przyśpiesza aktualizacje danych dla nowo dołączonych węzłów w sieci.

### Bazy danych

CouchDB to nierelacyjna baza danych (noSQL) typu *open source* rozwijana poprzez Fundację Apache, która gromadzi i przechowuje dane w formacie dokumentów JSON (ang. *JavaScript Object Notation*). Językiem jaki został użyty do napisania CouchDB jest Erlang.

Dostęp do systemu realizuje się za pomocą interfejsu programowania aplikacji (API), korzystającego z mechanizmu REST. Przechowywane dokumenty zawierają klucze oraz odpowiadające im wartości. Klucz jest to łańcuch znaków (tzw. *String*), który jest unikalny w ramach jednego dokumentu.

Baza danych CouchDB została zastosowana z racji tego, iż jest jedną z dwóch baz danych (razem z LevelDB) wspieranych przez platformę Hyperledger Fabric. Stanowi strukturę danych określoną jako *World State,* opisaną w podrozdziale 4.5.1. Obsługuje ona podstawowe operacje na *chaincode,* a ponadto umożliwia wdrażanie indeksów, przez co zwiększa wydajność bazy danych i skraca czas wykonywania zapytań.

Rysunek 56. Operacje pomiędzy *chaincode* a bazą danych

Obraz zawierający strzałka

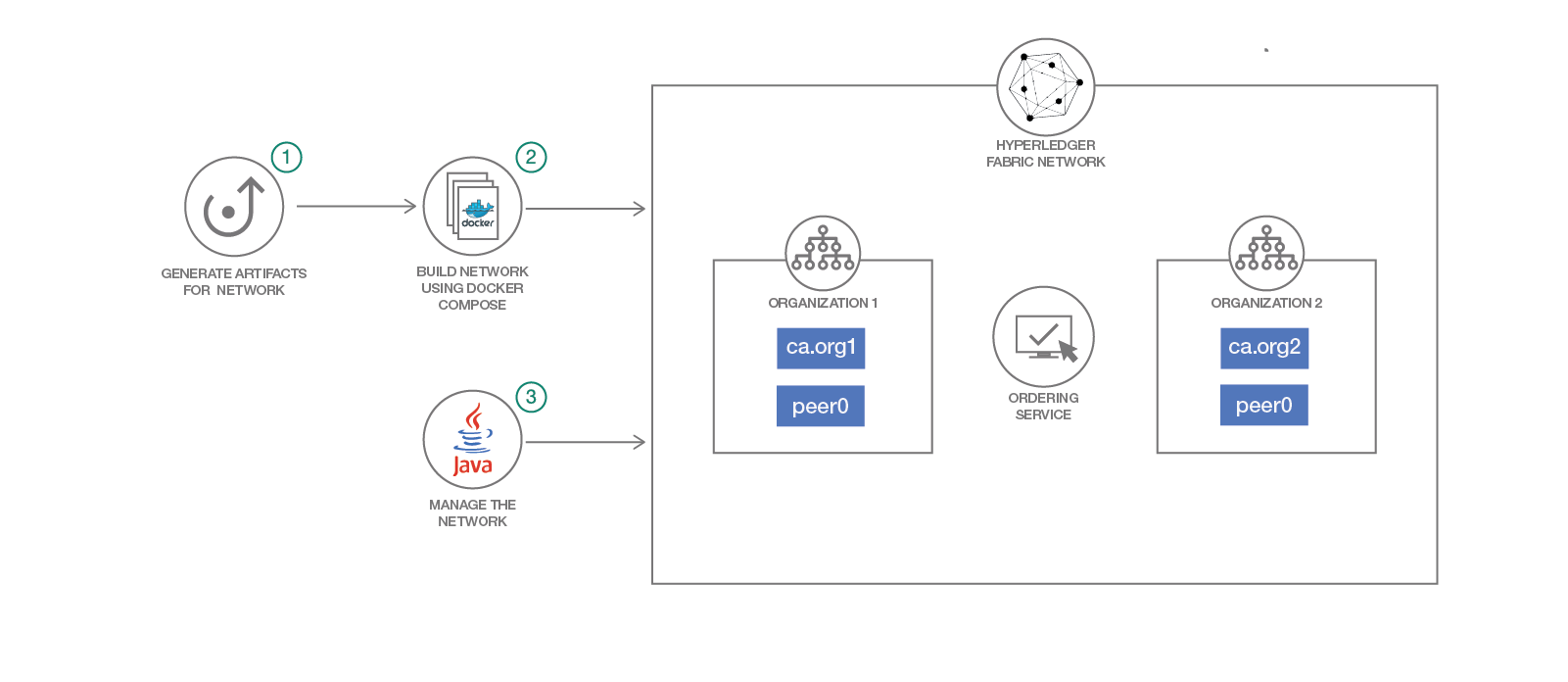
Opis wygenerowany automatycznieźródło:https://kctheservant.medium.com/putstate-and-getstate-the-api-in-chaincode-dealing-with-the-state-in-the-ledger-part-1-2008d6a86f98

CouchDB stanowi osobny węzeł w sieci, który jest uruchomiony jako kontener. Aby skonfigurować bazę danych należy wskazać ją w pliku konfiguracyjnym *core.yaml* i ustawić odpowiednie parametry takie jak *stateDatabase*, *couchDBConfig, couchDBAddress* itd.

### Inne oprogramowanie

## Charakterystyka środowiska uruchomieniowego

Rysunek 57. Elementy środowiska uruchomieniowego



Źródło: <https://github.com/IBM/blockchain-application-using-fabric-java-sdk>

W celu uruchomienia całego środowiska systemu należy podjąć odpowiednie kroki.

Sieć *blockchain:*

1. Wygenerowanie artefaktów oraz plików konfiguracyjnych sieci
2. Uruchomienie sieci wykorzystując Docker (utworzenie węzłów)
3. Utworzenie oraz inicjalizacja kanału w sieci
4. Wdrożenie *chaincode* na kanał
5. Inicjacja *chaincode* na węzły

Rysunek 58. Utworzenie kontenerów (węzłów) wykorzystując Docker

Obraz zawierający tekst

Opis wygenerowany automatycznie

Źródło: Opracowanie własne

Rysunek 59 Utworzenie kanału i przydzielenie do niego węzłów



Źródło: Opracowanie własne

Rysunek 60 Wdrożenie *chaincode* na kanał

Obraz zawierający tekst

Opis wygenerowany automatycznie

Źródło: Opracowanie własne

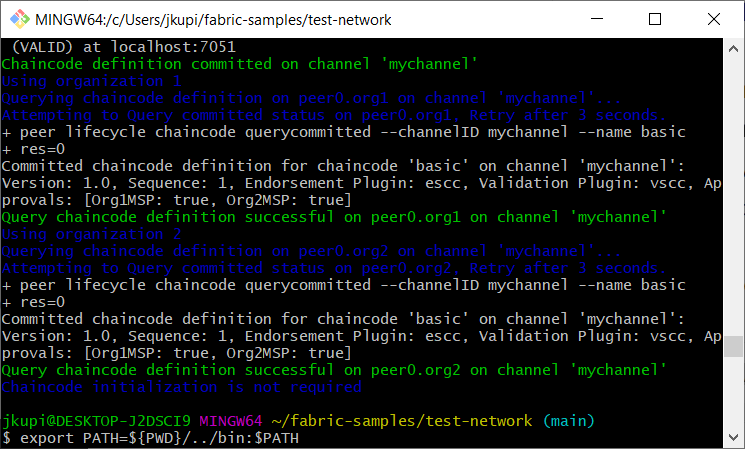
Rysunek Kompilacja *chaincode*

Obraz zawierający tekst

Opis wygenerowany automatycznie

Źródło: Opracowanie własne

Rysunek 62. Inicjacja *chaincode* na węzłach



Źródło: Opracowanie własne

Aplikacja (*SDK Java*):

1. Rejestracja administratora oraz konfiguracja Certyfikatu Tożsamości

Obraz zawierający tekst

Opis wygenerowany automatycznie

1. Rejestracja użytkowników oraz przydzielenie ich do organizacji

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

1. Połączenie aplikacji z utworzonym kanałem



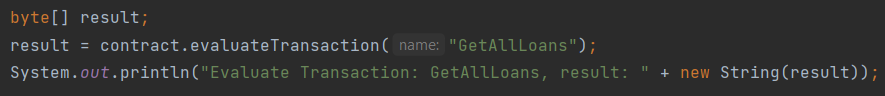
1. Zadeklarowanie *chaincode*, który zostanie wykonany



1. Inicjalizacja *Ledger*’a



1. Wykonywanie transakcji zawartych w *chaincode*



### Konfiguracja sprzętowa

Aby uruchomić sieć Hyperledger Fabric na lokalnej maszynie należy spełnić pewne wymogi, Zalecana jest instalacja poniższego oprogramowania w ostatnio wydanej wersji:

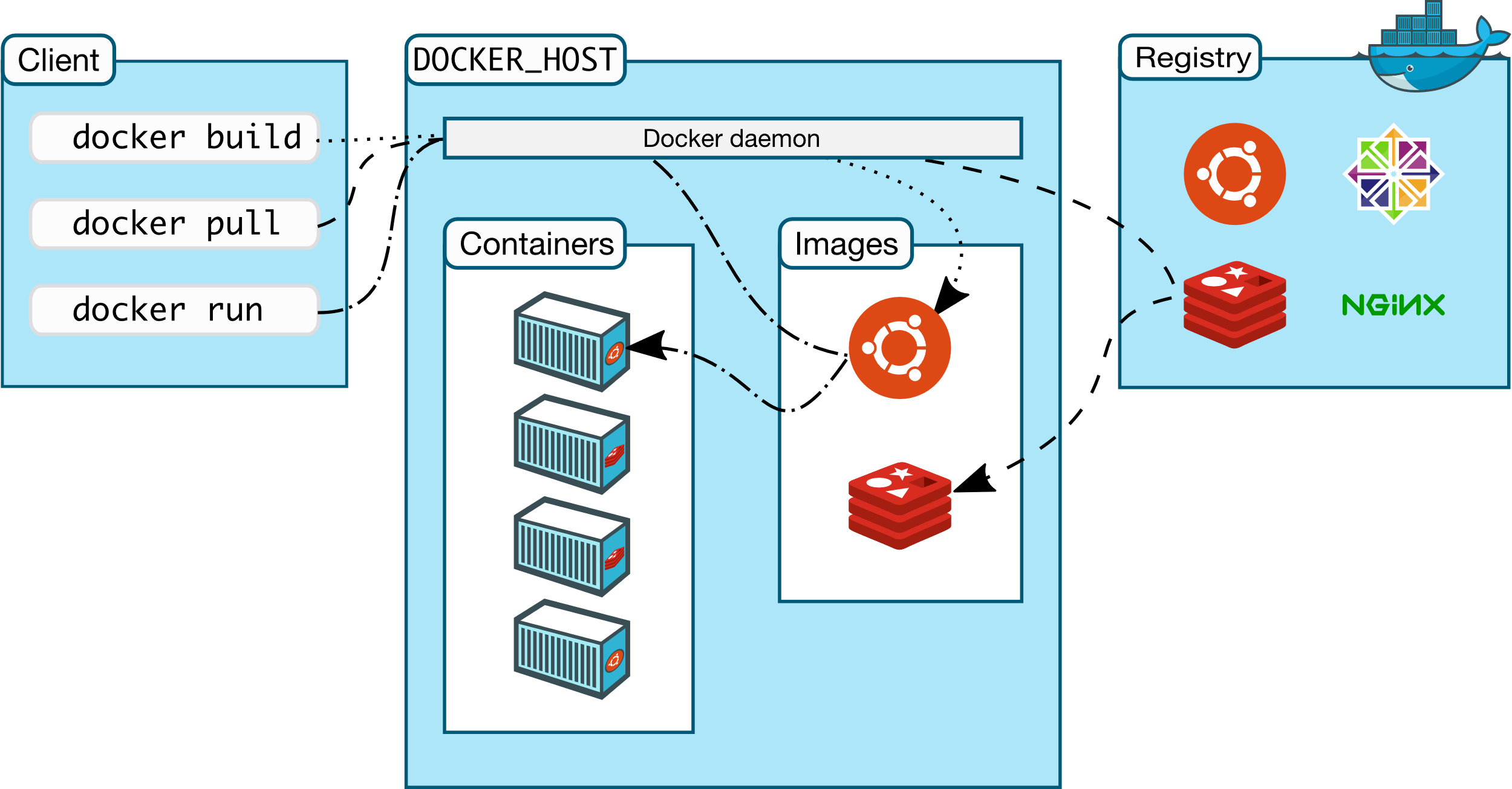
Git – aby używać skryptów pochodzących z repozytorium *Hyperledger* *Fabric samples*, w celu umożliwienia zastosowania nieznanych przez system operacyjny komend w CLI.

Docker - utworzenie sieci kontenerów, będących węzłami

Gradle – narzędzie do automatyzacji kompilacji

Java SDK Client – interakcja z siecią *blockchain* oraz *chaincode*

### Konteneryzacja //Warstwa wirtualizacji

Docker korzysta z architektury typu klient-serwer. Klient porozumiewa się poprzez REST API z Docker *deamon*, który zajmuje się budowaniem oraz uruchomianiem kontenerów. Ponadto zarządza obrazami, sieciami oraz woluminami. Obrazy przechowywane są w rejestrach, a w przypadku użycia komend przez Klienta zostają stamtąd pobierane.

Rysunek 63. Architektura platformy Docker

Źródło: <https://docs.docker.com/get-started/overview/>

Wykorzystując platformę Docker zostaje zbudowana sieć łańcucha bloków. Został zastosowany do tego przeznaczony obraz Hyperledger Fabric Docker (hyperledger/fabric-peer). Sieć zostaje stworzona poprzez skrypt, który generuje niezbędne artefakty(np. tak zwany *genesis block*), powołuje węzły do życia za pomocą *docker compose*, tworzy kanał oraz przydziela do niego węzły. Każdy z węzłów w sieci stanowi oddzielny kontener, który znajduje się na określonym porcie. Węzeł *peer0.borrowers* pochodzi z organizacji *Pozyczkobiorcy*, natomiast *peer0.lenders* – *Pozyczkodawcy*. Przechowują one kopie *Ledger* oraz uruchamiane są na nich *chaincodes*.

Rysunek 64. Uruchomione kontenery stanowiące węzły w sieci

Obraz zawierający tekst

Opis wygenerowany automatycznie

Źródło: opracowanie własne

### Systemy operacyjne

ŁYNDOWS

### Wymagane licencje

Napisałem ze open source………

### Sposób monitorowania systemu

System jest możliwy do monitorowa poprzez Serwisy Operacyjne (ang. *The Operations Service*), które są dostępne dla każdego węzła w sieci Hyperledger Fabric. Serwis Operacyjny stanowi serwer HTTP, oferujący operacje *RESTful API* takie jak:

1. Zarządzanie logami
2. Stan zdrowia (ang. *health checks*)
3. Metryki
4. Informacja wersji

Aby uruchomić na węźle *Operation Service* należy uprzednio skonfigurować plik *core.yaml*. W tym celu trzeba ustawić odpowiedni adres oraz port, a następnie należy użyć wygenerowanego certyfikatu oraz klucza w celu autoryzacji.

*Health checks* stanowi zasób REST. Wysyłając żądanie GET pod *endpoint* */healthz* serwis dostarcza informacji o stanie węzłów. W przypadku kiedy jest prawidłowy, otrzymujemy odpowiedź w postaci dokumentu JSON:

{

"status": "OK",

"time": "2021-12-10T23:00:00Z"

}

W przeciwnym przypadku, jeśli któryś z węzłów zwróci błąd, serwis wskaże wykryte nieprawidłowości, które mogą dotyczyć np. stanu *Docker deamon* oraz bazy danych *CouchDB*. Poniżej przykładowa odpowiedź serwisu w postaci dokumentu JSON:

{

"status": "Service Unavailable",

"time": "2021-12-10T23:00:00Z",

"failed\_checks": [

{

"component": "docker",

"reason": "failed to connect to Docker daemon: invalid endpoint"

}

]

}

Sposobem monitorowania, który umożliwia najszerszy przegląd systemu stanowią metryki. Zawierają one informację, które umożliwiają wgląd w jego zachowanie. Metryki zostały zaimplementowane w systemie poprzez Prometheus. Jest to darmowa aplikacja, służąca do monitorowania oraz rejestrowania zdarzeń w czasie rzeczywistym, oparta na modelu HTTP.

Obraz zawierający stół

Opis wygenerowany automatycznie

W celu wdrożenia oprogramowania *Prometheus* do systemu należy:

1. Zainicjować zmienne środowiskowe w pliku *Dockerfile*
2. Przypisać odpowiednie porty w powyższych zmiennych środowiskowych
3. Skonfigurować plik o nazwie *prometheus.yml*
4. Skonfigurować plik ­*docker-compose.yml*
5. Uruchomić sieć *blockchain* (podrozdział 5.3.)
6. Uruchomić *Prometheus* poprzez wprowadzenie komendy „*docker-compose up -d*”

Po odpowiedniej konfiguracji metryki dostępne są pod punktem końcowym */metrics*. Zakres udostępnionych informacji poprzez metryki jest bardzo szeroki. Obejmuje on między innymi informacje dotyczące *chaincode* takie jak liczba jego wykonań, czas jego uruchomienia, liczba niepowodzeń jego uruchomienia, liczba jego uruchomień, które przekroczyły dopuszczalny czas. Metryki mogą dotyczyć również bloków znajdujących się w sieci. Dane jakie możemy uzyskać to przykładowo czas przetwarzania bloku, ilość bloków, czas zatwierdzenia bloku. Ponadto możemy otrzymać informacje dotyczące transakcji, bazy danych, węzłów, logów.

Następnym sposobem monitorowania systemu jest wykorzystanie pakietu *common/flogging*. Jest to pakiet, umożliwiający kontrole logów pochodzących od węzłów, tworzący swego rodzaju rejestr zdarzeń. Logi stanowią ciąg znaków (tzw. *String*), który zawiera czas utworzenia, nazwę węzła z którego pochodzi log, rodzaj oraz treść. Są one klasyfikowane ze względu na rodzaj zdarzenia.

Wyróżniamy następujące rodzaje logów:

* *FATAL*
* *PANIC*
* *ERROR*
* *WARNING*
* *INFO*
* *DEBUG*

Ponadto system może być również monitorowany za pomocą *Command Line Interface*, który został opisany w podrozdziale 4.3.5.

## Charakterystyka repozytorium kodu

Efektywny proces wytwarzania oprogramowania powinien być wspierany przez system kontroli wersji. Dzięki owemu systemowi możliwe jest śledzenie zmian w kodzie źródłowym oraz zachowaniu spójności repozytorium. Ze względu na to, iż nowoczesna kultura pracy programistów charakteryzuje się decentralizacją zasobów ludzkich, występuje potrzeba zastosowania systemu, który pozwoli na wspólny dostęp do plików.

Repozytorium jest to rodzaj biblioteki cyfrowej, w której przechowywane są dokumenty. W celu ustrukturyzowania plików oraz projektów zostało utworzone repozytorium na platformie GitHub. Jest to platforma hostingowa typu *open source* oparta na systemie GIT, stworzona w 2008 roku. Z korzyści jakie przynosi nam korzystanie z repozytorium możemy wyróżnić:

* Przede wszystkim automatyczne wykrywanie zmian w plikach i automatyczne dokonywanie aktualizacji kodu. Wszelkie działania w obrębie repozytorium są dokumentowane tworząc historię zmian. Dzięki temu możliwe jest zidentyfikowanie autora, a w przypadku wystąpienia problemów z kodem przywrócenie poprzednich stanów.
* Odpowiednio skonfigurowane repozytorium zajmuje o wiele mniej miejsca niż cały projekt.
* W przypadku pracy w zespole zasoby są zunifikowane, a każdy z członków posiada do nich dostęp.
* Możliwe jest ujednolicenie zasad współpracy, co pomaga w rozwiązywaniu kolizji oraz konfliktów

### Struktura repozytorium

W repozytorium zostały wyodrębnione komponenty systemu, artefakty oraz dokumentacja. Struktura jest następująca:

* *application-java* (folder, w którym znajduje się *Client SDK Java*)
* *chaincode-java* (folder, w którym znajdują się pliki *chaincode* napisanego w języku *Java*)
* *network* (folder, w którym znajdują się obrazy Docker, pliki konfiguracyjne oraz skrypty niezbędne do utworzenia sieci *blockchain*)
* *README.md* (plik tekstowy, zawierający informacje o projekcie)
* *enterprise\_architect\_projects.eapx* (plik zawierający projekty, pochodzące z *Enterprise Architect*)

Rysunek 65. Struktura repozytorium na platformie GitHub

Obraz zawierający tekst, zrzut ekranu, monitor, ekran

Opis wygenerowany automatycznie

Źródło: Opracowanie własne

### Standardy tworzenia kodu źródłowego

Aby kod źródłowy był czytelny oraz zunifikowany zastosowano pewien zestaw reguł, określających jak powinien wyglądać. Dotyczą one komentowania kodu, konwencji nazewniczych, formatowania kodu oraz konstrukcji programistycznych.

Powodem dla którego należy stosować komentarze w kodzie jest uczynienie go czytelniejszym dla siebie, swoich współpracowników oraz innych osób, które będą miały z nim styczność. Komentarze nie powinny być długie, pisane pełnymi zdaniami. Najlepszą formą są krótkie i zwięzłe notatki (najlepiej w formie wypunktowanej), dostarczające najważniejszych informacji w celu zrozumieniu kodu. W kodzie źródłowym występują komentarze obejmujące pojedynczą linię kodu zaczynające się od znaków „**//**” oraz wieloliniowe. Wieloliniowe możemy podzielić na dwa typy:

1. użyte w celu dokumentacji, zaczynające się od znaków „***/\*\****” i kończące „***\*/***”
2. użyte w przypadku, kiedy kod przestał być odpowiedni i nie chcemy go kompilować, jednakże też go usuwać. Zaczynają się od znaków „***/\****”, a kończą „***\*/***”

Kolejnym sposobem na zwiększenie czytelności kodu jest stosowanie wcięć oraz paragrafów. Paragrafy stosuje się w celu grupowania powiązanych ze sobą linii kodu i odseparowania ich od ciągu dłuższego kodu. Wykonuje się je poprzez dodanie pustej linii (tzw. *whiteline*) pomiędzy blokami. Deklaracje klas oraz metod powinny zaczynać się w pierwszej kolumnie. Ze względu na to, że szerokość monitora jest ograniczona przyjęto maksymalną liczbę znaków w linii wynoszącą 120.

### Standardy nazewnicze

Nazwy zostały zainicjowane w języku angielskim z racji tego, iż język polski jest słabo przystosowany do pisania kodu źródłowego. Polskie słowa często bywają długie, co skutkuje większą objętością kodu. Wiele form oraz odmian wyrazów prowadzi do wątpliwości oraz niejednoznaczności. Ponadto w naszym języku występują liczne znaki diakrytyczne. Słowa kluczowe w językach programowania oraz bibliotekach występują w języku angielskim, co w przypadku połączenia z polskim nazewnictwem dałoby rezultat bardzo słabo czytelnego kodu. Kolejnym argumentem jest to, że kod źródłowy napisany w języku angielskim może zostać rozwijany przez zespoły programistów z obcych krajów. Zatem przyjęto następujące standardy nazewnicze odnośnie kodu źródłowego aplikacji oraz *chaincode* napisanego w języku programowania Java:

1. Nazwy jakie nadano zmiennym, klasom muszą być opisujące i łatwe do zidentyfikowania. Przykład: ***amount*** zamiast *a1*.
2. Należy trzymać się wcześnie przyjętej terminologii. Przykładowo, jeśli użytkownik systemu został zdefiniowany jako ***Student***,należy użyć tej nazwy, a nie np. *Client*.
3. Należy używać kombinacji małych oraz wielkich liter - wielka litera rozpoczyna kolejny wyraz w nazwie (patrz przykład 1.)
4. Nazwy klas oraz interfejsów zaczynają się wielką literą
5. Rozważnie używać skrótów, trzymając się jednej konwencji. Przykład: dla zmiennej *amountWithPercent* używać jednego skrótu ***aWithPer***, a nie np. *amWithPe*, *aWithP*.
6. Unikać długich nazw (przekraczających 15 znaków)
7. Unikać nazw podobnych do siebie

Tak zwane *Gettery* są to metody, które umożliwiają pobranie wartości prywatnego pola z zewnątrz, natomiast *Settery* umożliwiają na zmianę tego pola. Konstruktor jest to specjalna metoda tworząca obiekty danej klasy. Dla wspomnianych metod przyjęto następujące standardy nazewnicze:

1. *Gettery* powinny zawsze posiadać prefix „*get*”, a następnie po nim nazwę zmiennej. W przypadku zmiennych typu *boolean* prefixem będzie „*is*”. Przykład: ***getName()***, **isVerified()**
2. *Settery* powinny zawsze posiadać prefix „set”, a następnie po nim nazwę zmiennej. Przykład***: setName(String name)***, ***setVerified(boolean isVerified)***
3. Konstruktory powinny zawsze posiadać taką samą nazwę jak klasa oraz dodane po niej nawiasy. Przykład: ***Student()***

W Javie występują również stałe które, określone są jako *static final*. W ich przypadku nazwy składają się jedynie z wielkich liter, a wyrazy rozdzielone są znakiem podkreślenia. Przykład: **MAX\_LOAN\_AMOUNT**

## Zastosowane wzorce projektowe

Wybór wzorców projektowych:

1. Design Patterns: Elements of Reusable Object-Oriented Software, Erich Gamma , Richard Helm , et al.
2. Patterns of Enterprise Application Architecture, Martin Fowler
3. Clean Architecture: A Craftsman's Guide to Software Structure and Design, Robert C. Martin

<https://mohan-venkataraman.medium.com/using-patterns-in-coding-smart-contracts-412547051cb9>

<https://objectcomputing.com/resources/publications/sett/april-2020-hyperledger-fabric-myths-and-reality>

### Wzorce logiki dziedziny

### Wzorce obsługi źródeł danych

### Wzorce zachowań dla mapowania obiektowo-relacyjnego

### Wzorce struktury dla mapowania obiektowo-relacyjnego

### Wzorce odwzorowań obiektów i relacyjnych metadanych

### Wzorce warstwy prezentacji

### Wzorce dystrybucji

### Wzorce współbieżności autonomicznej

### Wzorce stanu sesji

### Wzorce podstawowe

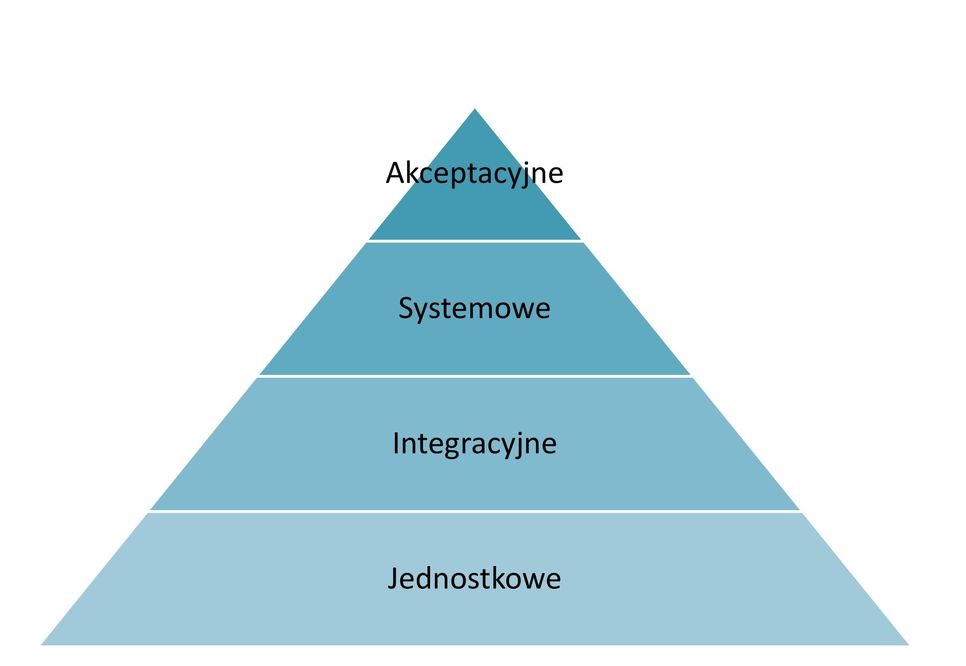
### Wzorce obsługi błędów

## Testy systemu informatycznego

Testowanie systemu informatycznego stanowi proces, który ma na celu zminimalizowanie ryzyka wystąpienia błędu w środowisku produkcyjnym, a ponadto uzyskanie maksymalnej pewności, iż oprogramowanie funkcjonuje zgodnie z założeniami. Jest to jeden ze środków, pozwalających na zapewnienie wysokiej jakości oprogramowania. Podczas jego trwania przeprowadzana jest weryfikacja oraz walidacja oprogramowania (czy oprogramowanie jest zgodne ze specyfikacją i oczekiwaniami użytkownika). Dzięki przeprowadzonym testom jesteśmy w stanie wykryć defekty, które mogą wynikać np. z błędów w kodowaniu czy definicji wymagań. Należy mieć na uwadze, iż podczas testowania nie jest zazwyczaj możliwe wykrycie wszystkich defektów oprogramowania. Wyróżniamy następujące poziomy testów:

1. Testy jednostkowe
2. Testy integracyjne
3. Testy systemowe
4. Testy akceptacyjne

Rysunek 68. Poziomy testowania systemów informatycznych



Źródło: Wstęp do testowania : Szymon Ramczykowski

### Testy jednostkowe

<https://medium.com/coinmonks/how-to-create-a-java-chaincode-and-deploy-in-a-hyperledger-fabric-2-network-65199e5f645d>

<https://javaguirre.me/2019/01/21/testing-chaincode-on-hyperledger-fabric>

<https://blog.qalabs.pl/junit/junit5-pierwsze-kroki/>

<https://blog.codeleak.pl/2017/11/junit-5-meets-assertj.html>

Apache JMeter, Junit, mockito, assertj

Testy jednostkowe są testami na najniższym poziomie. Odnoszą się one do niewielkich i ściśle wyizolowanych fragmentów kodu, a każdy z elementów systemu musi być potraktowany w sposób odosobniony. Owe testy nie powinny wnikać w szczegóły procesu biznesowego. Wykonywane są zwykle w środowisku programistycznym z dostępem do kodu źródłowego. Polegają one na wykonywaniu wybranego fragmentu instrukcji w celu zweryfikowania, czy jej działanie wykonywane jest prawidłowo według przyjętych założeń.

Testy jednostkowe zostały wykonane za pomocą biblioteki *JUnit 5*, która umożliwia ich automatyczne tworzenie w technologii Java. Narzędzie spełnia założenia owych testów i umożliwia:

* oddzielenie testów od kodu,
* parametryzowane przypadki testowe,
* wiele mechanizmów uruchamiania,
* wstrzykiwanie zależności
* integracje z różnymi środowiskami programistycznymi.

Do pracy z projektem opartym o *JUnit 5* należy posiadać środowisko z zainstalowanym *Java Development Kit* w wersji co najmniej 8 oraz zintegrowane środowisko programistyczne (w przypadku niniejszych testów *Intellij IDEA Community*).

Do zaimplementowania testów integracyjnych użyto ponadto biblioteki *AssertJ*, która służy do pisania asercji (instrukcji, których celem jest potwierdzenie, że predykat jest prawdziwy w danym miejscu w kodzie).

Dzięki wykorzystaniu metod pochodzących z *JUnit* oraz *AssertJ* możemy połączyć najlepsze co oferują, dzięki czemu jesteśmy w stanie pisać czytelne testy jednostkowe.

W celu instalacji powyższych bibliotek należy dodać zależności w pliku *build.gradle*:

testImplementation 'org.junit.jupiter:junit-jupiter:5.8.2'  
testImplementation 'org.assertj:assertj-core:3.21.0'

Do przeprowadzania testów *chaincode* została utworzona do tego przeznaczona klasa *LoanTransferTest*.Testy objęły metody dotyczące operacji odczytu, dodania, zmiany oraz usunięcia poszczególnych obiektów. Mają one na celu sprawdzenie poprawności wykonania się metod przy zadanych danych wejściowych. Wywoływane są one poprzez narzędzie kompilacji *gradle*.

Zaimplementowane zostały następujące testy jednostkowe:

@Test  
public void whenLoanExists()

@Test  
public void whenLoanDoesNotExist()

@Test  
public void invokeInitLedgerTransaction()

@Test  
public void invokeGetAllLoanTransaction()

Przetestowano przypadek, w którym następuje odwołanie do nieznanej transakcji:

@Test  
public void invokeUnknownTransaction()

Rysunek . Wynik przeprowadzonych testów jednostkowych

Obraz zawierający tekst

Opis wygenerowany automatycznie

Źródło: opracowanie własne

Rysunek 70. Przykładowy wynik testów integracyjnych wygenerowany poprzez narzędzie *Gradle*

Obraz zawierający tekst

Opis wygenerowany automatycznie

Źródło: Opracowanie własne

### Testy integracyjne

System informatyczny może składać się z wielu podsystemów, aplikacji czy modułów. W porównaniu do testów jednostkowych, które obejmowały pojedyncze części kodu, testy integracyjne mają na celu weryfikacje działania komponentów współpracujących ze sobą. Ponadto mają wykazać jak zachowują się poszczególne elementy w sytuacji awarii lub dysfunkcji jednego z nich. Przekonanie o spójności systemu otwiera drogę do przeprowadzenia dalszych testów.

W celu przeprowadzenia testów integracyjnych pomiędzy siecią *blockchain* a *chaincode* skorzystano z *Command Line Interface* - *GitBash*. Aby uzyskać dostęp do *peer0* należało ustawić zmienne środowiskowe:

export CORE\_PEER\_TLS\_ENABLED=true

export CORE\_PEER\_LOCALMSPID="Org1MSP"

export CORE\_PEER\_TLS\_ROOTCERT\_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt

export CORE\_PEER\_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp

export CORE\_PEER\_ADDRESS=localhost:7051

W tym momencie jesteśmy w stanie sprawdzać połączenie pomiędzy utworzoną siecią a kodem źródłowym zawartym w *chaincode*. Za pomocą CLI możemy zainicjować nasz *Ledger.* W *endorsement policy* zostało określone, że transakcje wymagają zatwierdzenia poprzez obie organizacje, zatem wywołanie operacji zawartej w *chaincode* (metoda *initLedger*)należy kierować do obu węzłów:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Został zwrócony komunikat o poprawnym wykonaniu metody. W celu potwierdzenia poprawności jej wykonania zostały odczytane pożyczki z *Ledger*, który jest komponentem znajdującym się w sieci *blockchain*

Obraz zawierający tekst

Opis wygenerowany automatycznie

Poprawne wykonanie tej metody poprzez CLI w *GitBash* świadczy o tym, że utworzona sieć łańcucha bloków jest zintegrowana z *chaincode.*

<https://github.com/hyperledger/fabric-sdk-java>

### Testy systemowe

Testy systemowe obejmują cały system. Powinny one ujmować wymagania funkcjonalne oraz nie funkcjonalne. Ocenie podlega gotowy produkt bez nadmiernego zgłębiania wewnętrznej architektury oraz kodu źródłowego. Wyróżniamy następujące typy:

* instalacyjne
* funkcjonalne
* interfejsu (inaczej zwane testami użyteczności)
* wydajnościowe
* regresywne
* bezpieczeństwa

<https://wiki.hyperledger.org/display/fabric/Quality+Assurance%3A+Tests%2C+Strategy%2C+Reports>

### Testy akceptacyjne

Testy akceptacyjne realizowane są przez zewnętrzny zespół testerów lub już bezpośrednio przez klienta w oparciu o własne zasoby. Nie mają one za zadanie wykrywania błędów w aplikacji, lecz mają na celu potwierdzenie, że dostarczony produkt jest zgodny z dokumentacją czy zapisami w umowie. Argumentem dlaczego testy akceptacyjne powinny być wykonywane przez klienta jest to, że testerowi może brakować wiedzy branżowej, którą zwyczajnie posiada pracownik. Kolejnym aspektem jaki trzeba wziąć pod uwagę podczas końcowego poziomu testowania są kwestie prawne. Wymuszają one działanie systemu w konkretny sposób, przez co musi być zgodny z wszelkimi ustawami oraz rozporządzeniami prawnymi. Dlatego też testy akceptacyjne systemu powinny być zlecane do zewnętrznego audytu.

## Wnioski

Podczas etapu implementacji systemu informatycznego dobrano odpowiednio środowisko wytwórcze oraz uruchomieniowe tak, aby zostały spełnione przyjęte założenia. Właściwie dobrane technologie mają za zadanie ułatwienie oraz przyśpieszenie tego procesu. W tym celu należało dokonać oceny technologii ze względu na określone wymogi. Pod uwagę trzeba było wziąć kryteria takie jak:

* Dostosowanie do potrzeb wytwarzania oprogramowania
* Znajomość technologii
* Użyteczność
* Wsparcie technologii
* Struktura
* Skalowalność
* Czas i koszt wdrożenia

Reasumując, wybór technologii zależy od budowanego produktu. Każda technologia ma pewne ograniczenia, dlatego nie warto z niej korzystać w dowolnym projekcie. Narzędzia techniczne należy stosować do celów, do których zostały stworzone.

# EWALUACJA systemu informatycznego

## Wprowadzenie

W niniejszym rozdziale zostanie przeprowadzona ewaluacja systemu informatycznego, która ma na celu ocenę realizacji postawionego zadania. Zostaną porównane założenia i cele, jakie zostały przyjęte na początku pracy dyplomowej z uzyskanymi wynikami. Na sam koniec zostaną przedstawione rekomendację dotyczące zarówno rozwoju systemu, jak i doskonalenia w zakresie metod inżynierii systemów.

## Metoda oceny uzyskanych wyników

## Ocena uzyskanych wyników

### Ocena stopnia implementacji systemu

Ocena jaka część z wybranego wariantu (maksymalnego, minimalnego lub realnego) została zaimplementowana

JAK JA NIC NIE ZROBIE

### Ocena stopnia pokrycia wymagań

Ocena odnosząca się do stopnia pokrycia wymagań w ramach wybranego wariantu (maksymalnego, minimalnego lub realnego).

### Ocena funkcjonalności systemu

Ocena funkcjonalności odnosi się do stopnia zgodności wytworzonego systemu z postawionymi wymaganiami.

### Ocena jakości systemu

### Ocena użyteczności systemu

Ocena użyteczności odnosi się do stopnia przydatności systemu wytworzonego w wybranym wariancie do wsparcia działalności organizacji która będzie go eksploatować.

## Kierunki rozwoju i doskonalenia

### Rekomendacje kierunku rozwoju systemu

System mógłby zostać rozszerzony o następujące możliwości przebiegu obsługi pożyczki:

* Pożyczkobiorca oraz pożyczkodawca ustalają warunki(a nie pożyczkobiorca jak do tej pory było realizowane). Kiedy te pokrywają się, pożyczki realizują się automatycznie poprzez wykonanie *chaincode*.
* Studenci wpłacają kapitał na wspólny portfel uczelni – następuje głosowanie (co jest jednym z popularniejszych zastosowań technologii *blockchain*) wśród społeczności komu zostaje przydzielona pożyczka

### Rekomendacje w zakresie doskonalenia metod inżynierii systemów

W celu doskonalenia w zakresie inżynierii systemów zostaną przedstawione metody odnoszące się do poszczególnych etapów cyklu życia oprogramowania.

Kluczem do sukcesu jest dobrze przeprowadzona analiza biznesowa. Rekomendowany jest rozwój w tym kierunku poprzez **wykorzystanie sprawdzonych źródeł wiedzy** - BABOK (*Business Analysis Body of Knowlegde*). Ważne jest **doskonalenie** **umiejętności modelowania** w języku UML oraz BPMN. Narzędziemużytym podczas analizy biznesowej oraz analizy systemu informatycznego jest *Enterprise Architect*, którego nauka jest godna polecenia.

Dokonując analizy, gdzie potrzebujemy uzyskać informację i dokonać specyfikacji systemu, **pomocnymi technikami** mogą być: przyjęcie roli uczestnika zewnętrznego np. klienta, prowadzenie dyskusji z ekspertami, ankiety i przeprowadzenie wywiadów czy analiza istniejących rozwiązań.

Podczas projektu systemu informatycznego należy zdefiniować reprezentację architektoniczną. Etap ten polega na **gromadzeniu zdobytych doświadczeń** w podobnych systemach lub dziedzinach problemowych, tak aby wysiłek nie został zmarnowany na ponowne odkrywanie architektury. Dlatego też zalecane jest **wykorzystywanie wszelkich wzorców architektonicznych**, które stanowią gotowe elementy rozwiązujące powtarzające się problemy architektoniczne. Projektując system informatyczny należy **zdefiniować modele danych** znajdujących się w nim. Należy wskazać informacje, które powinny być przechowywane w bazie danych, oraz określić ich strukturę. Szczególną uwagę trzeba zwrócić na takie cechy jak poprawność (zgodność z rzeczywistością), pełność (czy żaden z elementów nie został pominięty) oraz istotność każdego z elementów dla funkcjonowania organizacji.

Jeśli chodzi o etap implementacji w pierwszej kolejności trzeba **dobrać odpowiednie** języki programowania, biblioteki, bazy danych oraz wszelkie narzędzia służące do tworzenia oprogramowania **zgodnego z przyjętymi wymaganiami oraz założeniami**. Przede wszystkim rekomendowane jest **doskonalenie umiejętności programowania** oraz **nauka technologii**. Bez znajomości obsługi danych narzędzi będą one bezużyteczne. Głównym i priorytetowym źródłem wiedzy w tym przypadku jest **dokumentacja**. W przypadku kiedy nie możemy rozwiązać naszego problemu w oparciu o dokumentacje, dobrym sposobem będzie próba odnalezienia rozwiązania na **forum internetowym** związanym z daną tematyką (np. *stackoverflow.com*)

## Wnioski

# Zakończenie

# BIBLIOGRAFIA

W tym miejscu należy osadzić generowany automatycznie spis pozycji literaturowych do których odwołujemy się w pracy. Należy wykorzystać narzędzie MENDELEY - jest ono darmowe i pozwala zapanować nad literaturą i przypisami. Przypisy literaturowe proszę tworzyć w zgodzie ze standardem cytowań IEEE. Wykaz literatury musi być numerowany tj. nr, bez kropki w nawiasach kwadratowych i posortowany alfabetycznie (trzeba to skonfigurować we wtyczce mendeley dla MS Office)

* Sinclair Davidson , Primavera De Filippi, Jason Potts, **„*Economics of Blockchain*”**
* Nikhil Yadav, Sarasvathi V ***„Venturing Crowdfunding using Smart Contracts in Blockchain”***
* Satoshi Nakamoto,***“Bitcoin: A Peer-to-Peer Electronic Cash System”***
* Huasheng Zhu, Zach Zhizhong Zhou, ***„Analysis and outlook of applications of blockchain technology to equity crowdfunding in China”***
* Nofer, M., Gomber, P., Hinz, O., & Schiereck, D. (2017). Blockchain. *Business & Information Systems Engineering*, *59*(3), 183-187.
* Dariusz Dziuba, **„*Technologia blockchain crowdfunding: zastosowania, korzyści i oczekiwania*”**
* Dariusz Dziuba, **„*Rozwój systemów crowdfundingu – modele, oczekiwania i uwarunkowania*”**
* K.Król, „***Finansowanie społecznościowe jako źródło finansowania przedsięwzięć w Polsce”***
* Dariusz Dziuba, ***„Finansowanie społecznościowe: prognozy długookresowe vs. uwarunkowania rozwoju”***
* Piotr Międlar, **„*Blockchain w systemie finansowym*”**
* Raful P. Naik, „***Optimising the SHA256 Hashing Algorithm for Faster and More Efficient Bitcoin Mining”***
* Agnieszka Szewczyk, „***Wykorzystanie aplikacji mobilnych mediów społecznościowych”***
* Primavera De Filippi, ***„Blockchain-based Crowdfunding: what impact on artistic production and art consumption?”***
* Katarzyna Ciupa, ***„Warianty zastosowania koncepcji blockchain a modele ich doboru”***
* Krzysztof Piech, „***Leksykon pojęć na temat technologii blockchain oraz kryptowalut”***
* Disruption of ﬁnancial intermediation by FinTech: a
* review on crowdfunding and blockchain
* Disruption of ﬁnancial intermediation by FinTech: a
* review on crowdfunding and blockchain
* Disruption of ﬁnancial intermediation by FinTech: a
* review on crowdfunding and blockchain
* Disruption of ﬁnancial intermediation by FinTech: a
* review on crowdfunding and blockchain
* Disruption of ﬁnancial intermediation by FinTech: a
* review on crowdfunding and blockchain
* Disruption of ﬁnancial intermediation by FinTech: a
* review on crowdfunding and blockchain
* ial intermediation by FinTech: a
* review on c
* Cynthia Weiyi Cai, “***Disruption of ﬁnancial intermediation by FinTech: a review on crowdfunding and blockchain”***
* Martin Valenta, Philipp Sandner, ***„Comparison of Ethereum, Hyperledger Fabric and Corda”****,* Czerwiec 2017, FSBC Working Paper
* Monika Sitarska-Buba, ***„ARCHITEKTURA SYSTEMÓW BLOCKCHAIN 2.0”*** , Uniwersytet Ekonomiczny we Wrocławiu
* Maciej Gil, ***“Inżynieria oprogramowania”***
* Krzysztof Wąsek, ***„Rola mediów społecznościowych oraz ich wpływ na procesy rekrutacji pracowników”***
* Michał Chmielecki, ***„Rekrutacja z wykorzystaniem mediów społecznościowych – wyniki badań”***
* Kamila Tomczyk, Ilona Pawełoszek, ***„Zarządzanie relacjami z klientami banków spółdzielczych z wykorzystaniem mediów społecznościowych”***
* Wyrozębski P., Metodyka PRINCE2 [w] „***Metodyki zarządzania projektami”,*** wyd. Bizarre, Warszawa 2011
* International Institute of Business Analysis “***BABOK® v3 A GUIDE TO THE BUSINESS ANALYSIS BODY OF KNOWLEDGE***”
* R.W.Griffin, ***„Podstawy zarządzania organizacjami”***,*1997*
* Krzysztof Wołowicz***, „Wpływ sytuacji gospodarczej na korzystanie z pożyczek i kredytów”***
* [*https://hyperledger-fabric.readthedocs.io/en/release-2.2/*](https://hyperledger-fabric.readthedocs.io/en/release-2.2/)
* Ence Zhou, Haoli Sun, Bingfeng Pi, Jun Sun, Kazuhiro Yamashita, Yoshihide Nomura ***„Ledgerdata Refiner: A Powerful Ledger Data Query Platform for Hyperledger Fabric”***
* Scott W. Ambler, „***Software Process MentorWriting Robust Java Code*”**, The AmbySoft Inc. Coding Standards for Java v17.01d, 15.01.2000
* Brendan M. Sullivan, Gopikrishna Karthikeyan, Zuli Liu, Wouter Lode, Paul Massa, Mahima Gupta, ***”Socioeconomic group classification based on user features”***
* Christopher Lunt, ***„Authorization and authentication based on an individual's social network”***
* Pawlak Rafał***”Testowanie oprogramowania. Podręcznik dla początkujących”*** Helion, 2014

[1] *A Framework for Assessing and Improving Enterprise Architecture Management*, Version 2.0, GAO-10-846G, U.S. Government Accountability Office, Washington 2010.

[2] Ackerman L., „The Challenge of Change”, w *Managing change : cases and concepts*, T. Jick, Red. Irwin, Boston, 1993.

[3] Alberts D., J. Garstka, F. Stein, *Network Centric Warfare: Developing and Leaveraging Information Superiority*, CCRP 1999.

[4] Apanowicz J., *Metodologia nauk*, Dom Organizatora, Toruń 2003.

[5] *ArchiMate® 3.0 Specification*, The Open Group 2016.

[6] „Architektura Korporacyjna Państwa”. [Online]. Dostępne na: https://mac.gov.pl/projekty/architektura-korporacyjna-panstwa. [Udostępniono: 01-sty-2015].

[7] Armstrong M., *Zarządzanie zasobami ludzkimi*, Oficyna Ekonomiczna, Kraków 2004.

[8] Ayyagari M., T. Beck, A. Demirguc-Kunt, „Small and medium enterprises across the globe”, *Small Business Economics*, t. 29, nr 4, s. 415–434, 2007.

[9] Babbie E., *Badania społeczne w praktyce*, Wydawnictwo Naukowe PWN, Warszawa 2005.

[10] Balcerek A., „Od zarządzania do przywództwa”, *Zarządzanie Zasobami Ludzkimi*, nr 3–4, s. 51–61, 2008.

[11] Barker V., M. Duhaime, „Strategic change in the turnaround process: Theory and empirical evidence”, *Strategic Management Jurnal*, nr 18, s. 13–38, 1997.

[12] Becla A., S. Czaja, A. Zielińska, *Analiza kosztów-korzyści w wycenie środowiska naturalnego*, Difin, Warszawa 2012.

# Wykaz Rysnunków

[Rysunek 1. Różnica pomiędzy siecią a) zcentralizowaną b) decentralizowaną 2](file:///C:\Users\jkupi\OneDrive\Dokumenty\pracainz\PRACE_WERSJE\praca_inzynierska_v5.4.docx#_Toc91667860)

[Rysunek 2. Architektura łańcucha bloków danych 2](file:///C:\Users\jkupi\OneDrive\Dokumenty\pracainz\PRACE_WERSJE\praca_inzynierska_v5.4.docx#_Toc91667861)

[Rysunek 3. Przebieg transakcji zawieranych w blockchain 2](file:///C:\Users\jkupi\OneDrive\Dokumenty\pracainz\PRACE_WERSJE\praca_inzynierska_v5.4.docx#_Toc91667862)

[Rysunek 4. Wykres zainteresowania w ujęciu czasowym (czerwony – Ethereum, niebieski – Hyperledger) 2](#_Toc91667863)

[Rysunek 5. Różnice pomiędzy a) transakcja tradycyjną b) *smart contract* 2](file:///C:\Users\jkupi\OneDrive\Dokumenty\pracainz\PRACE_WERSJE\praca_inzynierska_v5.4.docx#_Toc91667864)

[Rysunek 6. Wykres przedstawiający media społecznościowe z największą ilością aktywnych użytkowników (w miliardach) 2](#_Toc91667865)

[Rysunek 7. Cykle życie oprogramowania w modelu spiralnym 2](#_Toc91667866)

[Rysunek 8. Projekt według PRINCE2 2](#_Toc91667867)

[Rysunek 9. Struktura metodyki PRINCE2 2](#_Toc91667868)

[Rysunek 10. Fazy *Agile Unified Process* 2](#_Toc91667869)

[Rysunek 11. Podstawowy model konceptualny analizy biznesowej (The Business Analysis Core Concept Model™) 2](#_Toc91667870)

[Rysunek 12. Analiza biznesowa (*Business Analysis Beyond Projects)* według BABOK® 2](#_Toc91667871)

[Rysunek 13. Elementy otoczenia organizacji 2](#_Toc91667872)

[Rysunek 14. Poziom inflacji r/r w Polsce w latach 2014-2021 2](#_Toc91667873)

[Rysunek 15. Zmiana dynamiki kredytów konsumpcyjnych a inflacja (%, r/r) **Błąd! Nie zdefiniowano zakładki.**](#_Toc91667874)

[Rysunek 16. Kredyty konsumpcyjne w sektorze bankowym i stopa referencyjna NBP **Błąd! Nie zdefiniowano zakładki.**](#_Toc91667875)

[Rysunek 17. Zmiana dynamiki kredytów konsumpcyjnych a stopa bezrobocia (%, r/r) **Błąd! Nie zdefiniowano zakładki.**](#_Toc91667876)

[Rysunek 18. Główne źródła dochodu studentów 2](#_Toc91667877)

[Rysunek 19. Najważniejsze wydatki studentów 2](#_Toc91667878)

[Rysunek 20. Najczęściej wskazywane przez studentów dodatkowe wydatki 2](#_Toc91667879)

[Rysunek 21. Diagram interesariuszy 2](#_Toc91667880)

[Rysunek 22. Aktorzy oraz role biznesowe 2](#_Toc91667881)

[Rysunek 23. Diagram klas – poziom konceptualny 2](#_Toc91667882)

[Rysunek 24. Maszyna stanowa obiektu *Wniosek* 2](#_Toc91667883)

[Rysunek 25. Maszyna stanowa obiektu *Pożyczka* 2](#_Toc91667884)

[Rysunek 26. Procesy główne 2](#_Toc91667885)

[Rysunek 27. Diagram procesu biznesowego - *obsługa pożyczki* 2](file:///C:\Users\jkupi\OneDrive\Dokumenty\pracainz\PRACE_WERSJE\praca_inzynierska_v5.4.docx#_Toc91667886)

[Rysunek 28. Diagram podprocesu biznesowego – *Złożenie wniosku o pożyczkę* 2](#_Toc91667887)

[Rysunek 29. Reguły biznesowe wchodzące w zakres polityki obsługi pożyczki 2](#_Toc91667888)

[Rysunek 30. Cykl specyfikacji potrzeb biznesowych 2](#_Toc91667889)

[Rysunek 31. Diagram wymagań biznesowych 2](#_Toc91667890)

[Rysunek 32. Diagram wymagań interesariuszy 2](#_Toc91667891)

[Rysunek 33. Diagram mapowania wymagań nr 1 2](#_Toc91667892)

[Rysunek 34. Diagram mapowania wymagań nr 2 2](#_Toc91667893)

[Rysunek 35. Diagram aktorów i ról systemowych 2](#_Toc91667894)

[Rysunek 36. Diagram przypadków użycia w obszarze funkcjonalnym obsługi pożyczki 2](#_Toc91667895)

[Rysunek 37. Diagram przypadków użycia w obszarze funkcjonalnym zarządzanie wnioskiem o pożyczkę 2](#_Toc91667896)

[Rysunek 38. Diagram przypadków użycia w obszarze funkcjonalnym zarządzania kontem 2](#_Toc91667897)

[Rysunek 39. Drzewo decyzyjne dotyczące reguły systemowej RS003 2](file:///C:\Users\jkupi\OneDrive\Dokumenty\pracainz\PRACE_WERSJE\praca_inzynierska_v5.4.docx#_Toc91667898)

[Rysunek 40. Diagram klas – poziom logiczny 2](#_Toc91667899)

[Rysunek 41. Diagram maszyny stanowej *wniosek o pożyczkę* 2](#_Toc91667900)

[Rysunek 42. Podział warstwowy oraz układ warstw systemu 2](#_Toc91667901)

[Rysunek 43. Komponenty systemu informatycznego opartego na Hyperledger Fabric 2](#_Toc91667902)

[Rysunek 44. Informacje zawarte w *access token* 2](#_Toc91667903)

[Rysunek 45. Przykładowe żądanie GET w *Graph API* 2](#_Toc91667904)

[Rysunek 46. Struktura bloków Hyperledger Fabric 2](#_Toc91667905)

[Rysunek 47. Funkcja skrótu SHA256 2](#_Toc91667906)

[Rysunek 48. Diagram przedstawiający przykładowy model biznesowy zawarty w *Chaincode* 2](#_Toc91667907)

[Rysunek 49. Model struktury danych Hyperledger Fabric 2](file:///C:\Users\jkupi\OneDrive\Dokumenty\pracainz\PRACE_WERSJE\praca_inzynierska_v5.4.docx#_Toc91667908)

[Rysunek 50. Przykładowe wartości stanu pożyczki P1 w strukturze danych *World State* 2](#_Toc91667909)

[Rysunek 51. Model struktury danych zawartych w *Blockchain* 2](#_Toc91667910)

[Rysunek 52. Przykładowy przepływ transakcji 2](#_Toc91667911)

[Rysunek 53. Reprezentacja grafowa przykładowej sieci społecznościowej 2](#_Toc91667912)

[Rysunek 54. Algorytm weryfikacji przyznania pożyczki 2](#_Toc91667913)

[Rysunek 55. Integracja z profilem społecznościowym *Facebook* 2](#_Toc91667914)

[Rysunek 56. Architektura platformy Docker 2](file:///C:\Users\jkupi\OneDrive\Dokumenty\pracainz\PRACE_WERSJE\praca_inzynierska_v5.4.docx#_Toc91667915)

[Rysunek 57. Uruchomienie programów w języku Java 2](#_Toc91667916)

[Rysunek 58. Szkielet budowy aplikacji 2](#_Toc91667917)

[Rysunek 59. Topologia komunikacji opartej na protokole plotkowania 2](#_Toc91667918)

[Rysunek 60. Operacje pomiędzy *chaincode* a bazą danych 2](#_Toc91667919)

[Rysunek 61. Charakterystyka środowiska uruchomieniowego 2](#_Toc91667920)

[Rysunek 62. Uruchomione kontenery stanowiące węzły w sieci 2](#_Toc91667921)

[Rysunek 63. Struktura repozytorium na platformie GitHub 2](#_Toc91667922)

# Wykaz Tabel

W tym miejscu należy osadzić generowany automatycznie spis tabel

[Tab. 1. Wynik analizy wpływu środowiska na wybór metod wsparcia zarządzania zmianą. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833883)

[Tab. 2. Warianty reengineeringu. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833884)

[Tab. 3. Komponenty techniczne i biznesowe w poszczególnych modelach operacyjnych. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833885)

[Tab. 4. Oczekiwane korzyści w warstwie organizacji, informacji i technologii. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833886)

[Tab. 5. Pytania adresowane w poszczególnych aspektach modelowania architektury SBN. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833887)

[Tab. 6. Pytania adresowane w poszczególnych fazach modelu cyklu życia. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833888)

[Tab. 7. Wykaz procesów realizowanych w poszczególnych fazach modelu cyklu życia. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833889)

[Tab. 8. Dokumenty związane z wdrażaniem praktyki architektonicznej SBN. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833890)

[Tab. 9. Poziomy dojrzałości praktyki architektonicznej SBN wg modelu ACMM. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833891)

[Tab. 10. Charakterystyka elementów metamodelu opisu architektury SBN. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833892)

[Tab. 11. Poziomy kaskady architektonicznej SBN. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833893)

[Tab. 12. Produkty prac architektonicznych wykorzystywane w procesie transformacji SBN. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833894)

[Tab. 13. Katalog wybranych standardów wymagających uwzględnienia w procesach inżynierii architektury SBN. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833895)

[Tab. 14. Przykładowe wartości współczynnika β w zależności od sytuacji decyzyjnej i motywacji eksperta. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833896)

[Tab. 15. Klasy i szczegółowe kryteria oceny wg modelu EAFSS. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833897)

[Tab. 16. Macierz mapowania celów stosowania modelu EAFSS na kryteria oceny. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833898)

[Tab. 17. Rangi porównań w metodzie AHP wg skali Saaty'ego. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833899)

[Tab. 18. Kryteria oceny funkcjonalności modeli na potrzeby zastosowania metody AHP. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833900)

[Tab. 19. Wartości współczynnika zgodności RI wg liczby kryteriów. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833901)

[Tab. 20. Macierz ocen użyteczności wybranych modeli wyznaczonych wg EAFSS. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833902)

[Tab. 21. Ocena rang dla poszczególnych macierzy porównań parami badanych modeli. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833903)

[Tab. 22. Macierz oceny poziomu ryzyka. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833904)

[Tab. 23. Ocena ryzyka wdrożenia opracowanego modelu. **Błąd! Nie zdefiniowano zakładki.**](#_Toc499833905)

1. Przypisy dolne stosować **WYŁĄĆZNIE** do wyjaśnienia lub uzupełnienia toku myślowego! Nie wpisywać w przypisach dolnych odwołań do literatury - do tego służą cytowania zgodne ze std. IEEE tj. [1] w tekście głównym. [↑](#footnote-ref-1)
2. J. Żurek (red.) Przedsiębiorstwo, zasady działania, funkcjonowanie, rozwój, Fundacja rozwoju Uniwersytetu Gdańskiego, Gdańsk 2007 [↑](#footnote-ref-2)
3. International Institute of Business Analysis, “***BABOK® v3 A GUIDE TO THE BUSINESS ANALYSIS BODY OF KNOWLEDGE***”, str. 26 [↑](#footnote-ref-3)