Obraz zawierający tekst, Czcionka, zrzut ekranu, logo

Opis wygenerowany automatycznie

**Damian Biskupski**

**236503**

PRACA DYPLOMOWA

inżynierska

na kierunku Informatyk Stosowana

**Aplikacja webowa do zamawiania i automatyzowania procesu wytwarzania własnoręcznie robionych gier planszowych**

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej

**Promotor:** dr inż. Joanna Ochelska-Mierzejewska

Łódź 2023

**Spis treści**

[1. Wstęp 3](#_Toc150446322)

[1.1. Problematyka i zakres pracy 3](#_Toc150446323)

[1.2. Cele pracy 3](#_Toc150446324)

[1.3. Przegląd literatury 4](#_Toc150446325)

[1.4. Układ pracy 4](#_Toc150446326)

[2. Przegląd wybranych aplikacji wspomagających projektowanie gier planszowych 5](#_Toc150446327)

[2.1. Aplikacja webowa Ntask 5](#_Toc150446328)

[2.2. Aplikacja webowa Jira Software 5](#_Toc150446329)

[2.3. Aplikacja webowa Trello 5](#_Toc150446330)

[2.4. Zalety i wady wybranych aplikacji wspomagających projektowanie gier planszowych 5](#_Toc150446331)

[3. Stos technologiczny 5](#_Toc150446332)

[3.1. Uzasadnienie wyboru technologii 5](#_Toc150446333)

[3.2. Język programowania TypeScript 7](#_Toc150446334)

[3.3. Framework NestJS 7](#_Toc150446335)

[3.4. Framework React 8](#_Toc150446336)

[3.5. Baza danych PostgreSQL 8](#_Toc150446337)

[3.6. System kolejkowania BullMQ 8](#_Toc150446338)

[4. Cykl projektowy aplikacji webowej BoardDesigner 9](#_Toc150446339)

[4.1. Sylwetka klienta i jego wymagania 9](#_Toc150446340)

[4.2. Wymagania funkcjonalne 9](#_Toc150446341)

[4.3. Wymagania niefunkcjonalne 9](#_Toc150446342)

[4.4. Architektura aplikacji 9](#_Toc150446343)

[4.5. Implementacja – punkty kluczowe 9](#_Toc150446344)

[4.6. Testy 9](#_Toc150446345)

[4.7. Instalacja i konserwacja 9](#_Toc150446346)

[5. Podręcznik użytkowania aplikacji webowej BoardDesigner 9](#_Toc150446347)

[5.1. Wprowadzenie do panelu klienta 9](#_Toc150446348)

[5.2. Wprowadzenie do panelu pracownika 9](#_Toc150446349)

[5.2.1. Zarządzanie aplikacją 9](#_Toc150446350)

[5.2.2. Zarządzanie projektami 9](#_Toc150446351)

[6. Podsumowanie 9](#_Toc150446352)

[6.1. Wnioski 9](#_Toc150446353)

[6.2. Perspektywy dalszego rozwoju tematyki 9](#_Toc150446354)

[Spis rysunków 9](#_Toc150446355)

[Spis tabel 10](#_Toc150446356)

[Bibliografia 10](#_Toc150446357)

# **Wstęp**

# **Problematyka i zakres pracy**

Prowadzenie działalności gospodarczej od zawsze było wymagającym wyzwaniem. Od czasów powstania pierwszych sklepów przedsiębiorcy starają się dotrzeć do jak najszerszego grona odbiorców. Z biegiem czasu ten cel osiągali coraz to nowszymi środkami masowego przekazu, gazetą, radiem, telewizją i najnowszą powstałą formą – Internetem. Ten ostatni sposób stał się normą, która jest niezbędna do przetrwania, a nawet istnienia współczesnej działalności gospodarczej. Statystyczny konsument stał się wygodniejszy przez ogólną wirtualizację świata, przez co posiadanie internetowej sprzedaży może stać się czynnikiem kluczowym w przypadku wyboru sklepu w którym dokona się zakupu produktu. E-commerce [1] jest obecnie jedną z najbardziej dochodowych gałęzi biznesu, a co za tym idzie, chcąc przetrwać na rynku przedsiębiorcy zmuszeni są do wyboru tej formy handlu.

Dodatkowym czynnikiem, które również jest ważny przy prowadzeniu działalności gospodarczej jest skuteczne planowanie pracy. W dzisiejszych czasach, kiedy świat wymaga coraz bardziej niebanalnych pomysłów, trzymanie planu pracy w głowie przestaje być możliwe przez złożoność wytwarzanych produktów. Kluczowe w tym przypadku staję się miejsce w którym możemy trzymać plan naszej pracy, co już zrobiliśmy, a co należy jeszcze zrobić. Można tego dokonać na różne sposoby za pomocą specjalnej tablicy lub zwykłej kartki papieru, jednak formą, która najlepiej się sprawdzi w większości przypadków to dedykowane miejsce do tego typu aktywności. Takim miejscem są wszelkiego typu programy wspomagające zarządzanie projektami, pozwalają one kategoryzować naszą pracę i skutecznie ją zaplanować, a co ważniejsze są dostępne z każdego miejsca, a jednocześnie są szybsze w użyciu niż inne sposoby na zarządzanie projektem.

Zakresem prac będzie analiza procesów towarzyszących obecnie w procesie wytwarzania i zamawiania produktu oraz pozostałych potrzeb konsumenckich i przełożenie tego na wymagania funkcjonalne i niefunkcjonalne oprogramowania. Dodatkowo poddane analizie i porównaniu zostaną obecnie dostępne rozwiązania na rynku wspierające projektowanie i na bazie ich wad i zalet stworzona zostanie nowa aplikacja.

# **Cele pracy**

Celem niniejszej pracy jest analiza wybranych istniejących aplikacji na rynku wspierających proces projektowania poprzez porównanie ich mocnych i słabych stron. Na podstawie przeprowadzonej analizy i sylwetki klienta zostanie utworzona nowa aplikacja webowa, która będzie automatyzować proces wytwarzania i zamawiania gier planszowych poprzez łączenie najlepszych cech i omijanie popełnionych błędów w porównywanych serwisach. Powstałe rozwiązanie końcowo zostanie porównane z wcześniej analizowanymi dostępnymi serwisami na rynku, w celu podsumowania czy wszystkie założenia zostały spełnione.

# **Przegląd literatury**

Nest.js: A Progressive Node.js Framework [2] – oficjalna dokumentacja techniczna framework’a Nest.js. Zasób ten wybrałem ze względu, że jest to jedna z lepiej napisanych dokumentacji na rynku. Opisuję ona działanie całej platformy programistycznej, jak i tego w jaki sposób można używać danych zależności i biblioteki współpracują z danym szkieletem aplikacyjnym.

Dav Vanderkam, TypeScript: Skuteczne programowanie [3] - książka zawierająca porady dotyczące dobrych praktyk i skutecznego posługiwania się językiem programowania TypeScript [4]. Zasób ten wybrałem ze względu na praktyczne przepisy oraz wskazówki, które mogą przynieść korzyści w celu optymalnego wykorzystania potencjału języka TypeScript [4].

Ian Sommerville, Software Engineering Ninth Edition [5] – opis cyklu projektowego w procesie wytwarzania oprogramowania. Zasób ten wybrałem ze względu na lepsze zrozumienie potencjału inżynierii oprogramowania oraz jak wygląda krok po kroku wytwarzanie profesjonalnego oprogramowania od wymagań klienta, aż po fazę konserwacji i utrzymania docelowego programu.

# **Układ pracy**

Praca zbudowana jest z dwóch części, część teoretyczna – rozdziały 1-3, część praktyczna – rozdziały 4-6. Pierwsza część pracy opisuję podstawy napotkanego problemu oraz obecnie dostępne na rynku rozwiązania oraz technologię, które zostaną przeanalizowane. Drugi rozdział opisuje cykl projektowy aplikacji wraz z wszystkimi jego fazami, według metodologii waterfall [6]. W tej części znajduję się również podręcznik użytkowania aplikacji oraz podsumowanie przeprowadzonego cyklu projektowego oraz perspektywy dalszego rozwoju tematyki.

# **Przegląd wybranych aplikacji wspomagających projektowanie gier planszowych**

# **Aplikacja webowa Ntask**

# **Aplikacja webowa Jira Software**

# **Aplikacja webowa Trello**

# **Zalety i wady wybranych aplikacji wspomagających projektowanie gier planszowych**

# **Stos technologiczny**

# **Uzasadnienie wyboru technologii**

Przez ostatnie paręnaście lat można zauważyć zachodzącą rewolucję w której Internet staje się drugą rzeczywistością handlu zwiększając swoją wartość o miliardy dolarów każdego roku, co przedstawia wykres dostępny rysunku 1. Oczywistą odpowiedzią na zachodzące zmiany, jak i prognozy rynkowe z których jasno wynika, że proces ten będzie postępował coraz szybciej jest przenoszenie się każdego przedsiębiorstwa do Internetu. Generuję to tym samym coraz więcej potrzeb „komputeryzacji” sklepów oraz systemów. Wynikiem tego procesu jest ogromny rozwój przeróżnych frameworków webowych i coraz to nowszych języków programowania.

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

Rysunek 1: Wykres wartości rynkowej i prognozowanej w handlu internetowym na świecie według portalu ecommerce guide [7].

Jednym z takich języków programowania, który co roku notuję coraz większy udział na rynku jest język programowania TypeScript [4], którego to wzrost popularności i ilości pobrań przedstawiony jest na wykresach na rysunkach 2 oraz 3. Wynika z tego jasno, że technologia ta zyskuję coraz to większą renomę notując nawet 3 krotny wzrost w ciągu ostatnich czterech lat. Dzieję się tak ze względu na szereg korzyści płynący z używania tej semantyki, między innymi:

* Wprowadzanie typowania przez nadawanie zmiennym określonego typu danych. Pozwala to uniknąć wielu błędów z odwoływania się do nieistniejących instancji obiektów klas.
* Dostęp do dekoratorów [10], które pozwalają dopisać dodatkową logikę do klas metod parametrów, zwiększając tym samym czytelność kodu.
* Wprowadzenie jawnej i prostej enkapsulacji [11] zmiennych i metod klas, co pozwala zachować kontrolę nad udostępnianiem wewnętrznej logiki na zewnątrz obiektu.

Obraz zawierający zrzut ekranu, linia, Wielobarwność, diagram

Opis wygenerowany automatycznie

Rysunek 2: Wykres reprezentujący rozkład użycia języków programowania w ciągu ostatnich 12 miesięcy [8].

Obraz zawierający linia, Wykres, diagram, stok

Opis wygenerowany automatycznie

Rysunek 3: Wykres reprezentujący ilość pobrań języka programowania TypeScript [9].

# **Język programowania TypeScript**

TypeScript [4] jako obudowa języka programowania JavaScript [12] jest wysokopoziomowym językiem zorientowanym obiektowo umożliwiającym również statyczne typowanie. Został on zaprojekowany przez korporację Microsoft [13] w 2012 roku głównie w celu umożliwienia definiowania przez programistę typów zmiennych. W praktyce oznacza to, że TypeScript [4] rozszerza JavaScript [12] o dodatkowe elementy ułatwiając tym samym tworzenie dużych projektów. Wprowadza on również szereg usprawnień z których część została wymieniona w rozdziale 3.1 oraz takich jak:

* Interfejsy – umożliwiające tworzenie kontraktów między warstwami w aplikacji.
* Klasy wraz z dziedziczeniem – umożliwiające tym samym tworzenie hierarchii obiektów oraz ułatwiając dzielenie się funkcjami i właściwościami.
* Moduły – umożliwiające podział całej aplikacji na niezależne części, ułatwiając tym samym lepsze zarządzanie udostępnianą logiką z danego modułu.
* Typy generyczne – umożliwiające tworzenie uniwersalnych fragmentów kodu, które możliwe są do użycia w różnych miejscach programu.

# **Framework NestJS**

NestJS [4] jest platformą programistyczną do budowy aplikacji serwerowych w środowisku uruchomieniowym Node.js [14]. Umożliwia programistą programowanie w czystym języku programowania JavaScript, ale przede wszystkim przy użyciu języka programowania TypeScript. Łączy w sobie zasady programowanie obiektowego poprzez wspieranie klas oraz ich instancji, ale również programowania funkcyjnego poprzez możliwość definiowania metod bez konieczności tworzenia obiektów, jak i również umożliwia programowanie reaktywne poprzez wspieranie operacji asynchronicznych opartych o promise [15] oraz funkcje zwrotne [16]. Zbudowana została przy użyciu TypeScript [4] na bazie wcześniej już wspomnianego Node.js [14] oraz serwera HTTP Express [17]. Platforma zapewnia specyficzną architekturę projektu, wprowadzając skalowalne, luźno połączone, ale przede wszystkim wysoce testowalne moduły aplikacyjne poprzez zastosowanie mechanizmy wstrzykiwania zależności [18]. Zapewnia ona możliwość budowy różnych aplikacji, takich jak monolity, mikroserwisy jak i również aplikację CLI [19] przy użycia API opartego na różnych protokołach, jak i również wsparcie GraphQL [20]. Dzięki takiej architekturze pomimo zmieniającego się sposobu komunikacji między modułami zapewniony jest pewnego rodzaju kontrakt między modułami ułatwiający zrozumienie, jak i wejście w projekt nowemu programiście. Warto również zauważyć, że budowa tej platformy jest mocno inspirowana platformą programistyczną Angular [21] poprzez zastosowanie modułów i wcześniej wspomnianego wstrzykiwania zależności [18].

# **Framework React**

React [22] jest platformą programistyczną, a nawet bardziej biblioteką języka programowania JavaScript [12] służącą do tworzenia interfejsów graficznych w środowisku uruchomieniowym Node.js [14]. Dysproporcja ta wynika z faktu, iż twór ten nie rozwiązuję problemów strukturalnych i architektonicznych, oferuje za to nowe podejście do tworzenia interfejsów oparte na budowie komponentowej. Nie narzuca on konkretnego stylu programowania dając pełną swobodę, jeżeli chodzi o sposób implementacji projektu. Fenomen tej platformy polega na tworzeniu wielu izolowanych komponentów, które same zarządzają własnym stanem, tworząc wspólnie jednolity i spójny interfejs graficzny. Pojedynczy komponent zbudowany jest z następujących elementów:

* Funkcji komponentu pełniącej reprezentacje komponentu.
* Funkcji stanów, które pozwalają przetrzymać pewien zdefiniowany, zmieniany stan w cyklu życia React
* Funkcji ubocznej w która pozwala na wykonanie pewnych zdarzeń asynchronicznie poza komponentem na przykład podczas renderowania danego komponentu
* Zwracanego elementu react, czyli fragmentu, który ma zostać wyrenderowany.

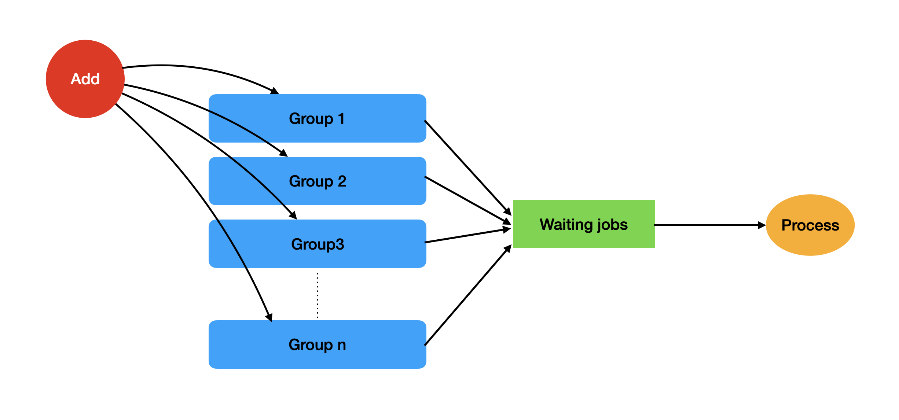
Taka budowa umożliwia pewien sposób izolowania logiki na mniejsze fragmenty pozwalając programiście na bycie zgodnym z jedną z głównych zasad SOLID – „single responsibility” [23]. Warto również zauważyć, że React udostępnia możliwość programowana opartego o klasy, jednak nowoczesne podejście opiera się głównie na funkcyjnym podejściu do komponentów.

# **Baza danych PostgreSQL**

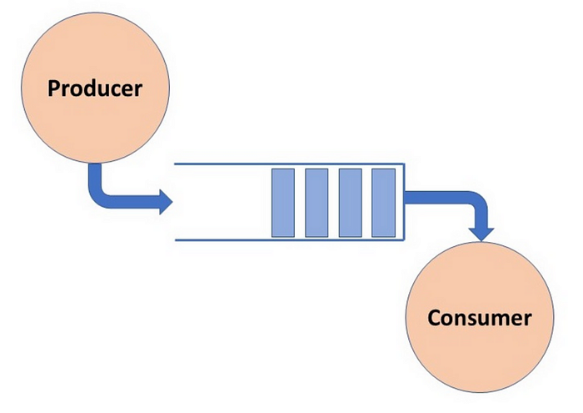
PostgreSQL [24] jest system zarządzania bazą danych oferującą obiektowo-relacyjne podejście. Oznacza to nie wiele więcej niż, połączenie cech relacyjnych baz danych z elementami programowania obiektowego. Takie podejście umożliwia elastyczne modelowanie skomplikowanych struktur danych poprzez możliwość korzystania z zapytań obiektowych ułatwiając tym samym operacje na samych obiektach, jak i relacjach między nimi. Sam system jest oprogramowaniem typu open source [25] oferując swoim użytkownikom wieloplatformowość poprzez dostępność na wszystkich dystrybucjach systemów typu UNIX oraz Windows oraz skalowalność poprzez tabele o rozmiarach nawet do 32 TB. Warto również zaznaczyć, że system ten udostępnia możliwość tworzenia zaawansowanych zapytań poprzez wsparcie dla proceduralnego SQL PL/pgSQL [26].

# **System kolejkowania BullMQ**

BullMQ [27] jest biblioteką do obsługi kolejkowania zadań zbudowaną na bazie Redis’a [28] w celu wykorzystania jego przepustowości. Pozwala ona rozwiązać wiele problemów komunikacyjnych między poszczególnymi modułami lub mikroserwisami, odciążając tym samym sam serwer. W praktyce oznacza to, że biblioteka pozwala zakolejkować, czyli odłożyć na stos zadań do zrobienia pewną pracę bez konieczności czekania innego procesu na jej zakończenie. Idealnie sprawdza się w przypadku operacji zakolejkowania wysyłki maila, operacji na plikach lub dostępu do usług w której w jednym czasie może znajdować się tylko jedno zadanie przez ogromną konsumpcję zasobów przez ten proces takich jak generowanie obrazów przez sztuczną inteligencję. Budowa tego systemu przedstawiona na rysunku 4 jest dość prosta, a co za tym idzie przyjazna jeżeli chodzi o obsługę lub ewentualne poszukiwanie błędów. Proces czyli Producer [29] dodaje zadanie do utworzonej kolejki lub grupy zadań, następnie zadanie czeka na wejście do Consumera [29] rysunek 5, gdzie następuję wykonanie i zakończenie danego zadania. Sama kolejka może być skonfigurowana na wiele sposobów LIFO [31], FIFO [30], ale przede wszystkim poprzez przyznanie priorytetu danym zadaniom.



Rysunek 4 Schemat działania systemu kolejkowania BullMQ [32].



Rysunek 5: Schemat wzorca Producer Consumer [33].

# **Zewnętrzne pakiety**

Pozostałe zewnętrzne pakiety również zostały wykorzystane w projekcie wraz z opisem:

Moduł Mailer

Moduł Jwt

Moduł Schedule

# **Cykl projektowy aplikacji webowej BoardDesigner**

# **Sylwetka klienta i jego wymagania**

# **Wymagania funkcjonalne**

# **Wymagania niefunkcjonalne**

# **Architektura aplikacji**

# **Implementacja – punkty kluczowe**

# **Testy**

# **Instalacja i konserwacja**

# **Podręcznik użytkowania aplikacji webowej BoardDesigner**

# **Wprowadzenie do panelu klienta**

# **Wprowadzenie do panelu pracownika**

# **Zarządzanie aplikacją**

# **Zarządzanie projektami**

# **Podsumowanie**

# **Wnioski**

# **Perspektywy dalszego rozwoju tematyki**

# **Spis rysunków**

[Rysunek 1: Wykres wartości rynkowej i prognozowanej w handlu internetowym na świecie według portalu ecommerce guide [7]. 6](#_Toc150512784)

[Rysunek 2: Wykres reprezentujący rozkład użycia języków programowania w ciągu ostatnich 12 miesięcy [8]. 6](#_Toc150512785)

[Rysunek 3: Wykres reprezentujący ilość pobrań języka programowania TypeScript [9]. 7](#_Toc150512786)

[Rysunek 4 Schemat działania systemu kolejkowania BullMQ [32]. 9](#_Toc150512787)

[Rysunek 5: Schemat wzorca Producer Consumer [33]. 10](#_Toc150512788)

# **Spis tabel**

# **Bibliografia**

[1] E-Commerce, [online], [dostęp 05.11.2023] https://www.techtarget.com/searchcio/definition/e-commerce

[2] Nest.js: A Progressive Node.js Framework, [online], [dostęp 06.11.2023] https://docs.nestjs.com/

[3] Dav Vanderkam, TypeScript: Skuteczne programowanie, APN Promise, 2020, ISBN 978-83-754-1420-2

[4] TypeScript, [online], [dostęp 06.11.2023] https://www.typescriptlang.org/

[5] Ian Sommerville, Software Engineering Ninth Edition, Pearson, 2011, ISBN 978-0-13-703515-

[6] Watterfall, [online], [dostęp 05.11.2023] https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model

[7] Wykres wartości rynkowej i prognozowanej w handlu internetowym na świecie, [online], [dostęp 07.11.2023] https://ecommerceguide.com/ecommerce-statistics/

[8] Wykres reprezentujący rozkład użycia języków programowania w ciągu ostatnich 12 miesięcy, [online], [dostęp 07.11.2023] https://www.jetbrains.com/lp/devecosystem-2022/

[9] Wykres reprezentujący ilość pobrań języka programowania TypeScript, [online], [dostęp 07.11.2023] https://npmtrends.com/typescript

[10] TypeScript decorators, [online], [dostęp 07.11.2023] https://www.typescriptlang.org/docs/handbook/decorators.html

[11] Encapsulation, [online], [dostęp 07.11.2023] https://www.sumologic.com/glossary/encapsulation/

[12] JavaScript, [online], [dostęp 08.11.2023] https://developer.mozilla.org/en-US/docs/Web/JavaScript

[13] Microsoft, [online], [dostęp 08.11.2023] https://www.microsoft.com/pl-pl

[14] Node.js, [online], [dostęp 09.11.2023] https://nodejs.org/en

[15] Promise, [online], [dostęp 09.11.2023] https://docs.nestjs.com/fundamentals/async-providers

[16] Callback, [online], [dostęp 09.11.2023] https://docs.nestjs.com/techniques/events

[17] Express, [online], [dostęp 09.11.2023] https://expressjs.com/

[18] Dependency Injection, [online], [dostęp 09.11.2023] https://docs.nestjs.com/fundamentals/custom-providers

[19] CLI, [online], [dostęp 09.11.2023] https://www.w3schools.com/whatis/whatis\_cli.asp

[20] GraphQL, [online], [dostęp 09.11.2023] https://graphql.org/

[21] Angular, [online], [dostęp 09.11.2023] https://angular.io/

[22] React, [online], [dostęp 10.11.2023] https://react.dev/

[23] SOLID, [online], [dostęp 10.11.2023] https://www.bmc.com/blogs/solid-design-principles/

[24] PostgreSQL, [online], [dostęp 10.11.2023] https://www.postgresql.org/

[25] Open source, [online], [dostęp 10.11.2023] https://opensource.org/licenses/

[26] PL/pgSQL, [online], [dostęp 10.11.2023] https://www.postgresql.org/docs/current/plpgsql.html

[27] BullMQ, [online], [dostęp 10.11.2023] https://docs.bullmq.io/

[28] Redis, [online], [dostęp 10.11.2023] https://redis.io/

[29] Kolejki, [online], [dostęp 10.11.2023] https://docs.nestjs.com/techniques/queues

[30] FIFO, [online], [dostęp 10.11.2023] https://www.investopedia.com/terms/f/fifo.asp

[31] LIFO, [online], [dostęp 10.11.2023] https://www.investopedia.com/terms/l/lifo.asp

[32] Schemat działania systemu kolejkowania BullMQ, [online], [dostęp 10.11.2023] https://docs.bullmq.io/bullmq-pro/groups

[33] Producer, Consumer wzorzec, [online], [dostęp 10.11.2023] https://medium.com/@karthik.jeyapal/system-design-patterns-producer-consumer-pattern-45edcb16d544