

Programowanie systemów rozproszonych	Projekt
Distributed Chat System	Imię i nazwisko: Jakub Olejarczyk Numer albumu: s086748 Kielce 2024/2025

# Spis treści

1. Wstęp.....	1
2. Struktura kontenerów.....	2
3. Konfiguracja projektu.....	2
3. Architektura systemu.....	5
3. Komunikacja w czasie rzeczywistym.....	7
4. Technologie.....	8

# 1. Wstęp

Aplikacja czatowa działa w czasie rzeczywistym, umożliwiając użytkownikom komunikację tekstową w ramach dedykowanych pokoi czatowych. System opiera się na architekturze rozproszonej, co pozwala na integrację z wieloma serwerami, zapewniając płynne i spójne doświadczenie użytkownika niezależnie od obciążenia sieci. Dzięki temu rozwiązanie efektywnie obsługuje dużą liczbę użytkowników jednocześnie, oferując niskie opóźnienia, wysoką dostępność oraz skalowalność w miarę rosnących potrzeb.

Aplikacja posiada zaawansowane funkcje, takie jak rejestracja i logowanie użytkowników z wykorzystaniem bezpiecznych tokenów JWT, zarządzanie profilami użytkowników oraz możliwości tworzenia pokoi czatowych. Powiadomienia w czasie rzeczywistym informują o nowych wiadomościach lub działaniach w pokojach, w których użytkownik bierze udział, co zwiększa interaktywność platformy.

Rozwiązanie zostało zbudowane w oparciu o nowoczesne technologie. Komunikację w czasie rzeczywistym obsługuje Socket.io, backend działa na Node.js, a Redis pełni rolę bufora wiadomości i mechanizmu kolejkowania zdarzeń. Architektura kontenerowa oparta na Dockerze zapewnia łatwość wdrażania i niezależność od środowiska.

Interfejs użytkownika został zaprojektowany z wykorzystaniem Angulara oraz biblioteki komponentów PrimeNG, co gwarantuje spójny wygląd i intuicyjną obsługę. Zarządzanie stanem aplikacji odbywa się za pomocą NgRx, co zapewnia wysoką wydajność i niezawodność systemu.

Aplikacja jest stabilna, skalowalna i przyjazna dla użytkownika, doskonale sprawdzając się zarówno w zastosowaniach prywatnych, jak i biznesowych.

## 2. Struktura kontenerów

```
jaqba98@jakub:~/jakub/studio/1/PSR/distributed-chat-system$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
61954f957e6d	nginx:1.27.2-alpine-slim	"docker-entrypoint..."	About a minute ago	Up 54 seconds	0.0.0.0:3003->80/tcp, [::]:3003->80/tcp	chat_load-balancer
9390848c32d7	nginx:stable-alpine-perl	"docker-entrypoint..."	About a minute ago	Up 55 seconds	0.0.0.0:3001->80/tcp, [::]:3001->80/tcp	accounts_load-balancer
883a95c30cf6	nginx:stable-alpine-perl	"docker-entrypoint..."	About a minute ago	Up 55 seconds	0.0.0.0:3002->80/tcp, [::]:3002->80/tcp	api_gateway_load-balancer
2c2cecb9eb1fc	nginx:stable-alpine-perl	"docker-entrypoint..."	About a minute ago	Up 55 seconds	0.0.0.0:3004->80/tcp, [::]:3004->80/tcp	rooms_load-balancer
03e93c4c44d5	distributed-chat-system-chat1	"docker-entrypoint.s..."	About a minute ago	Up 56 seconds	0.0.0.0:30031->3000/tcp, [::]:30031->3000/tcp	chat1
7d7b1eb5011f	distributed-chat-system-chat3	"docker-entrypoint.s..."	About a minute ago	Up 58 seconds	0.0.0.0:30033->3000/tcp, [::]:30033->3000/tcp	chat3
6011f10f0e7a	distributed-chat-system-chat2	"docker-entrypoint.s..."	About a minute ago	Up 57 seconds	0.0.0.0:30032->3000/tcp, [::]:30032->3000/tcp	chat2
505c91cd0e78	distributed-chat-system-chat5	"docker-entrypoint.s..."	About a minute ago	Up 58 seconds	0.0.0.0:30035->3000/tcp, [::]:30035->3000/tcp	chat5
f8d90e0809c8	distributed-chat-system-chat4	"docker-entrypoint.s..."	About a minute ago	Up 56 seconds	0.0.0.0:30034->3000/tcp, [::]:30034->3000/tcp	chat4
3f2d62b09f11	distributed-chat-system-base_chat	"docker-entrypoint.s..."	About a minute ago	Up 58 seconds		base_chat
175359b728e3	distributed-chat-system-accounts3	"docker-entrypoint.s..."	About a minute ago	Up 58 seconds	0.0.0.0:30013->3000/tcp, [::]:30013->3000/tcp	accounts3
654e21b01c0c	distributed-chat-system-accounts2	"docker-entrypoint.s..."	About a minute ago	Up 59 seconds	0.0.0.0:30012->3000/tcp, [::]:30012->3000/tcp	accounts2
16811a4d1878	distributed-chat-system-rooms1	"docker-entrypoint.s..."	About a minute ago	Up 56 seconds	0.0.0.0:30041->3000/tcp, [::]:30041->3000/tcp	rooms1
79672acaf776	distributed-chat-system-accounts1	"docker-entrypoint.s..."	About a minute ago	Up 59 seconds	0.0.0.0:30011->3000/tcp, [::]:30011->3000/tcp	accounts1
95b2330210cc	distributed-chat-system-accounts5	"docker-entrypoint.s..."	About a minute ago	Up 59 seconds	0.0.0.0:30015->3000/tcp, [::]:30015->3000/tcp	accounts5
da754c8b9024	distributed-chat-system-rooms3	"docker-entrypoint.s..."	About a minute ago	Up 55 seconds	0.0.0.0:30043->3000/tcp, [::]:30043->3000/tcp	rooms3
c6a40cd50991	distributed-chat-system-base_rooms	"docker-entrypoint.s..."	About a minute ago	Up 59 seconds		base_rooms
d1a5783a0f85	distributed-chat-system-rooms4	"docker-entrypoint.s..."	About a minute ago	Up 55 seconds	0.0.0.0:30044->3000/tcp, [::]:30044->3000/tcp	rooms4
f8a174d09704	distributed-chat-system-accounts4	"docker-entrypoint.s..."	About a minute ago	Up 58 seconds	0.0.0.0:30014->3000/tcp, [::]:30014->3000/tcp	accounts4
01e4cf2740f9	distributed-chat-system-rooms5	"docker-entrypoint.s..."	About a minute ago	Up 55 seconds	0.0.0.0:30045->3000/tcp, [::]:30045->3000/tcp	rooms5
5e78a220e271	distributed-chat-system-base_accounts	"docker-entrypoint.s..."	About a minute ago	Up 59 seconds		base_accounts
62f3ae2b306	distributed-chat-system-rooms2	"docker-entrypoint.s..."	About a minute ago	Up 55 seconds	0.0.0.0:30042->3000/tcp, [::]:30042->3000/tcp	rooms2
a4d53637841a	distributed-chat-system-base_api_gateway	"docker-entrypoint.s..."	About a minute ago	Up About a minute		api_gateway
1c9141c0e0af	distributed-chat-system-api_gateway1	"docker-entrypoint.s..."	About a minute ago	Up About a minute	0.0.0.0:30021->3000/tcp, [::]:30021->3000/tcp	api_gateway1
3e53180bc3fb	distributed-chat-system-api_gateway3	"docker-entrypoint.s..."	About a minute ago	Up About a minute	0.0.0.0:30023->3000/tcp, [::]:30023->3000/tcp	api_gateway3
deef68da01af	distributed-chat-system-api_gateway2	"docker-entrypoint.s..."	About a minute ago	Up 59 seconds	0.0.0.0:30022->3000/tcp, [::]:30022->3000/tcp	api_gateway2
7c08c17c7b3c	distributed-chat-system-api_gateway4	"docker-entrypoint.s..."	About a minute ago	Up About a minute	0.0.0.0:30024->3000/tcp, [::]:30024->3000/tcp	api_gateway4
52c72ea2a639	distributed-chat-system-api_gateway5	"docker-entrypoint.s..."	About a minute ago	Up 59 seconds	0.0.0.0:30025->3000/tcp, [::]:30025->3000/tcp	api_gateway5
0fe5747f33df	distributed-chat-system-chat-client	"docker-entrypoint.s..."	About a minute ago	Up About a minute	0.0.0.0:3333->80/tcp, [::]:3333->80/tcp	chat-client
6af477d0d0dc	redis:8.0-m02-alpine	"docker-entrypoint.s..."	About a minute ago	Up About a minute	0.0.0.0:6379->6379/tcp, [::]:6379->6379/tcp	distributed-chat-system-redis-1
737b42b50953	mysql:8.4	"docker-entrypoint.s..."	About a minute ago	Up About a minute	3306/tcp, 33060/tcp, 0.0.0.0:30010->30010/tcp, [::]:30010->30010/tcp	accounts_db
36669f2e7821	mysql:8.4	"docker-entrypoint.s..."	About a minute ago	Up About a minute	3306/tcp, 33060/tcp, 0.0.0.0:30030->30030/tcp, [::]:30030->30030/tcp	chat_db
65e578ad12a6	mysql:8.4	"docker-entrypoint.s..."	About a minute ago	Up About a minute	3306/tcp, 33060/tcp, 0.0.0.0:30020->30020/tcp, [::]:30020->30020/tcp	api_gateway_db
1780df49ccac	mysql:8.4	"docker-entrypoint.s..."	About a minute ago	Up About a minute	3306/tcp, 33060/tcp, 0.0.0.0:30040->30040/tcp, [::]:30040->30040/tcp	rooms_db

```
jaqba98@jakub:~/jakub/studio/1/PSR/distributed-chat-system$
```

Aplikacja składa się z wielu kontenerów, z których każdy realizuje pojedynczą funkcjonalność, składającą się na kompletną aplikację czatu.

## 3. Konfiguracja projektu

1. Link do projektu: [www.github.com/jaqba98/distributed-chat-system](https://www.github.com/jaqba98/distributed-chat-system)
2. Wymagane oprogramowanie:
  1. NodeJS [www.nodejs.org](https://www.nodejs.org)
  2. PNPM [www.pnpm.io](https://www.pnpm.io)
  3. Docker [www.docker.com](https://www.docker.com)
  4. Git [www.git-scm.com](https://www.git-scm.com)
  5. NX [www.nx.dev](https://www.nx.dev)
3. Kroki do wykonania w celu uruchomienia projektu:

1. Instalacja oprogramowania
  1. Zainstaluj NodeJS z oficjalnej strony oprogramowania
  2. Zainstaluj PNPM za pomocą odpowiedniej komendy

```
npm install -g pnpm@latest-10
```

3. Zainstaluj NX za pomocą odpowiedniej komendy

```
pnpm install -g nx
```

4. Zainstaluj Docker z oficjalnej strony oprogramowania
5. Zainstaluj Git z oficjalnej strony oprogramowania

## 2. Przygotowanie projektu

### 1. Uzyskaj link do projektu

1. Przejdź na stronę projektu na serwisie GitHub
2. Wciśnij zielony przycisk „Code” i wybierz zakładkę „HTTPS”
3. Skopiuj wyświetlony link (w następnych krokach będzie nam potrzebny)

2. Przejdź do dowolnej lokalizacji na twoim komputerze i wykonaj polecenie klonujące repozytorium z serwisu GitHub na twój komputer lokalny

```
git clone https://github.com/jaqba98/distributed-chat-system.git
```

3. Przejdź do nowo utworzonego folderu i wykonaj polecenie instalacji pnpm.

```
pnpm install -r
```

### 4. Zbuduj projekt

```
nx run-many --target=build
```

## 3. Uruchomienie projektu

1. Za pomocą docker compose uruchom projekt

```
sudo docker compose up -d
```

2. Weryfikacja czy wszystko działa – aby mieć pewność że wszystko działa wykonaj polecenie `sudo docker logs` na dowolnym serwisie

```
sudo docker logs accounts1
```

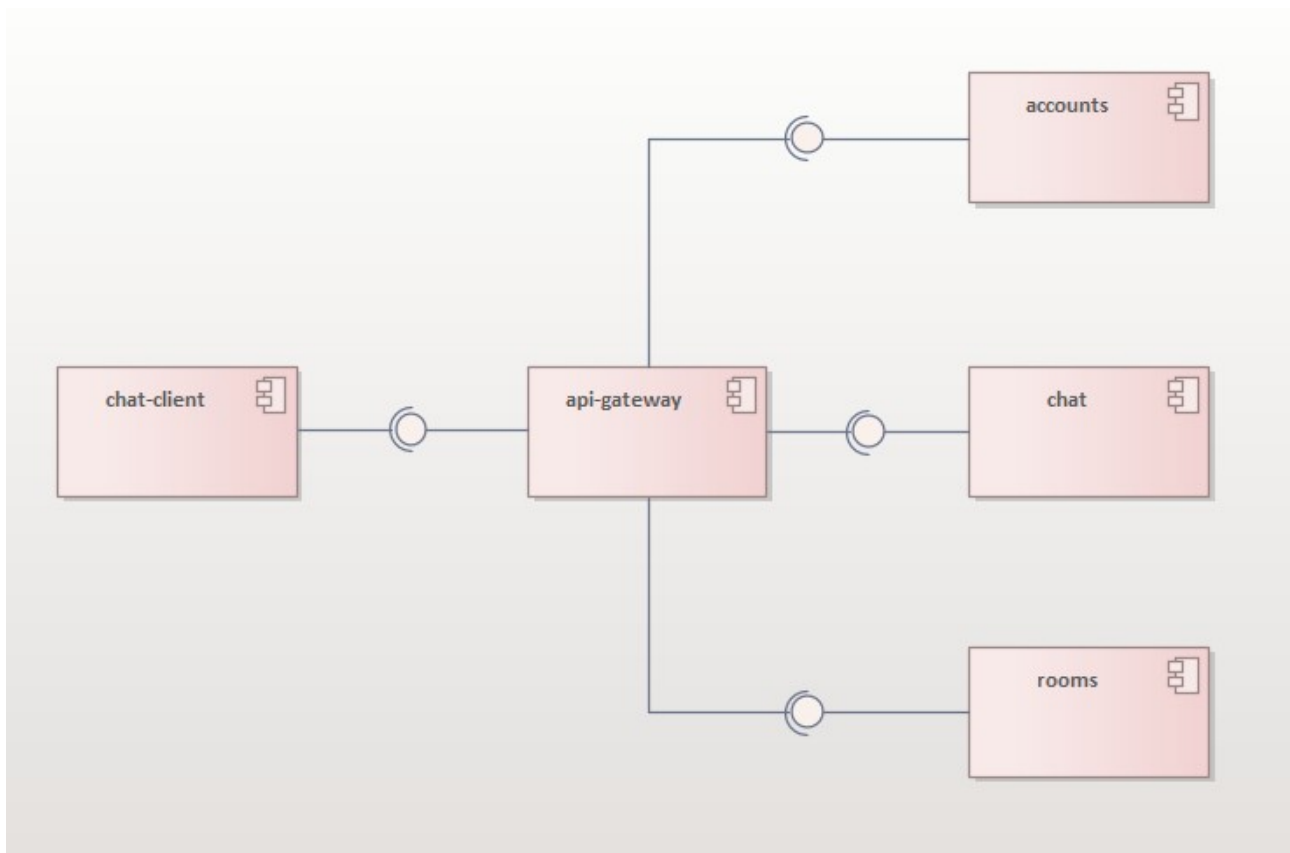
Jeśli w konsoli wyświetlił się napis "" wtedy mamy pewność że wszystko uruchomiło się poprawnie.

### 4. Wyłączenie projektu

```
sudo docker logs accounts1
```

### 3. Architektura systemu

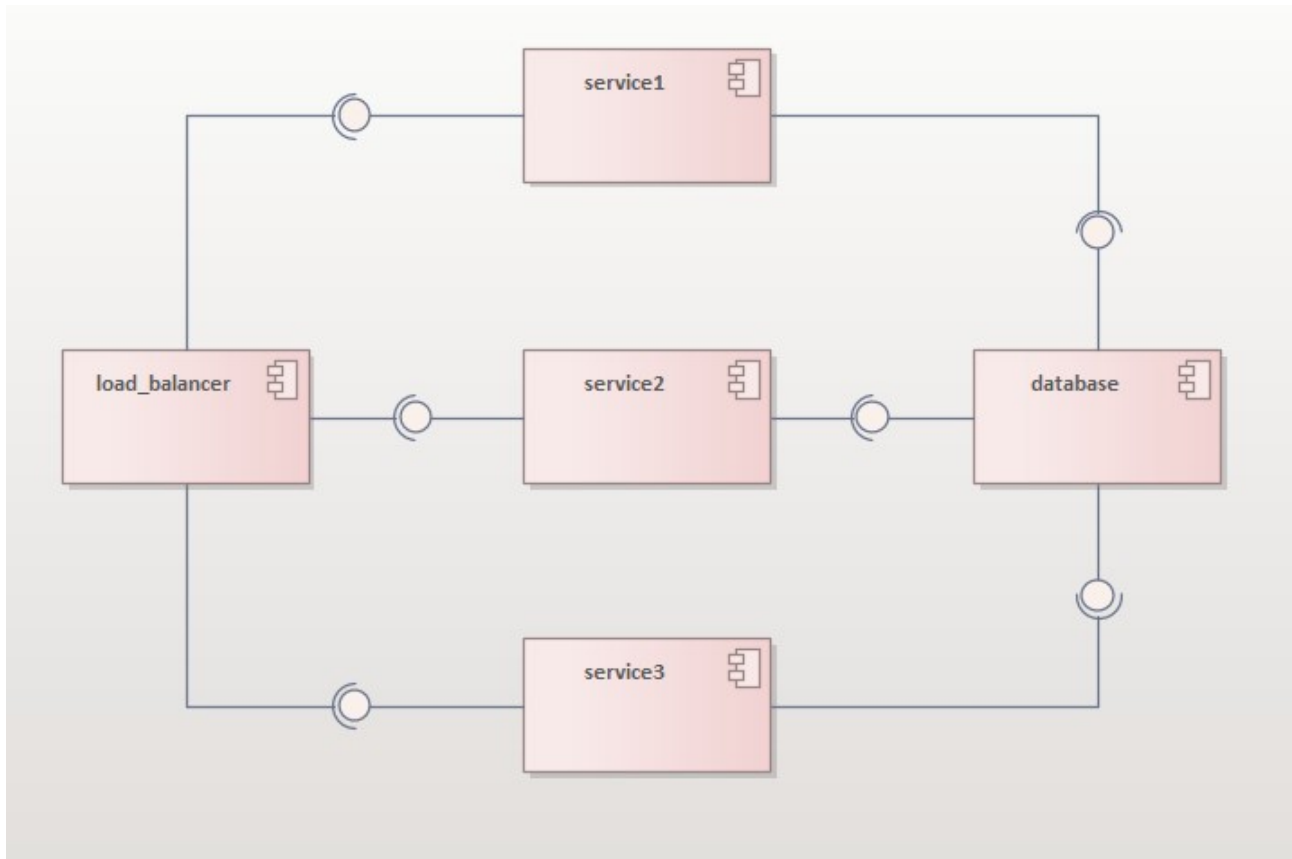
Architektura microservices na najwyższym poziomie



Składa się z:

- chat-client – Aplikacja frontendowa działająca po stronie klienta, zawierająca pełny interfejs użytkownika oraz funkcjonalności umożliwiające interakcję z systemem w czasie rzeczywistym.
- api-gateway – Mikroserwis pełniący funkcję centralnego punktu dostępu, zarządzający ruchem sieciowym pomiędzy klientem a poszczególnymi mikroserwisami.
- accounts – Mikroserwis odpowiedzialny za zarządzanie danymi użytkowników, w tym rejestrację, logowanie, autoryzację oraz przechowywanie.
- chat – Mikroserwis zajmujący się obsługą komunikacji w czasie rzeczywistym, zarządzający wymianą wiadomości pomiędzy użytkownikami, w tym kolejkowaniem i dostarczaniem ich do odpowiednich odbiorców.
- rooms – Mikroserwis dedykowany zarządzaniu pokojami czatowymi, obejmujący tworzenie.

## Architektura pojedynczego microservice



Każdy z mikroserwisów składa się z następujących komponentów:

- **load-balancer** – Odpowiedzialny za równoważenie obciążenia pomiędzy wieloma instancjami serwisu, zapewniając wysoką dostępność oraz optymalne wykorzystanie zasobów.
- **service** – Główna jednostka logiczna mikroserwisu, realizująca jego funkcjonalności; może być powielana wielokrotnie, aby skalować aplikację w zależności od obciążenia i wymagań.
- **database** – Dedykowana baza danych dla danego mikroserwisu, służąca do przechowywania oraz zarządzania danymi związanymi z jego działaniem.

### **3. Komunikacja w czasie rzeczywistym**

Główną funkcją aplikacji jest umożliwienie użytkownikom prowadzenia rozmów w czasie rzeczywistym w ramach wybranego pokoju czatowego. Oznacza to, że jeśli użytkownik „A” wyśle wiadomość, to wszyscy pozostali użytkownicy znajdujący się w tym samym pokoju natychmiastowo ją otrzymują, bez opóźnień. Użytkownicy aplikacji łączą się z jedną z wielu instancji mikroserwisu „chat”, które są rozmieszczone w różnych lokalizacjach. Pomimo tego, że użytkownicy mogą być podłączeni do różnych serwerów, rozproszonych geograficznie, na przykład na dwóch różnych kontynentach, komunikacja między nimi jest synchronizowana dzięki technologii „Redis”. Redis zapewnia efektywne buforowanie i synchronizację danych pomiędzy serwerami, co pozwala na utrzymanie płynnej i natychmiastowej wymiany wiadomości niezależnie od miejsca, w którym znajduje się użytkownik. Dzięki temu użytkownicy mogą bezproblemowo i w czasie rzeczywistym komunikować się, niezależnie od lokalizacji geograficznej i obciążenia poszczególnych serwerów.

## 4. Technologie

### 1. Frontend:

1. HTML: Język znaczników używanym do tworzenia struktury stron internetowych.
2. CSS / SCSS: CSS odpowiada za stylizację stron internetowych, a SCSS to jego rozszerzenie oferujące zaawansowane funkcje, takie jak zmienne i zagnieżdżanie stylów.
3. TypeScript: Jest rozszerzeniem JavaScript z obsługą statycznego typowania, co ułatwia tworzenie i utrzymanie złożonych aplikacji.
4. Angular: Jest frameworkiem frontendowym do budowy dynamicznych i modułowych aplikacji internetowych w oparciu o TypeScript.
5. RxJs: Biblioteka do reaktywnego programowania w JavaScript, umożliwiająca obsługę strumieni danych i zdarzeń asynchronicznych.
6. Nx: Narzędzie do zarządzania projektami monorepo, które usprawnia pracę nad wieloma aplikacjami i bibliotekami w jednym repozytorium.
7. NgRx: Biblioteka do zarządzania stanem w aplikacjach Angular z wykorzystaniem wzorca Redux.
8. PrimeNG: Zbiór komponentów UI dla Angulara, umożliwiający szybkie tworzenie atrakcyjnych interfejsów użytkownika.

### 2. Backend:

1. Socket.io: Biblioteka do obsługi komunikacji w czasie rzeczywistym między klientem a serwerem, często używana w aplikacjach czatowych.
2. Node.js: Środowisko uruchomieniowe dla JavaScript, umożliwiające tworzenie aplikacji serwerowych.
3. JWT: Standard tokenów służący do bezpiecznej autoryzacji użytkowników w aplikacjach.
4. Docker: Platforma do konteneryzacji aplikacji, pozwalająca na łatwe uruchamianie i skalowanie środowisk.
5. MySQL: Relacyjna baza danych, często używana do przechowywania i zarządzania danymi w aplikacjach internetowych.
6. Redis: Szybki magazyn danych w pamięci, używany do buforowania i obsługi kolejek.