



**POLITECHNIKA
RZESZOWSKA**
im. IGNACEGO ŁUKASIEWICZA

CYFRYWIZACJA SYSTEMÓW PRODUKCYJNYCH

Dokumentacja

Automatyzacja systemu powiadomień z wykorzystaniem narzędzie low-code n8n

Data wykonania: 26.01.2024

Autorzy:

Mykola Nykolaichuk 150174

Jan Drabek 175304

Spis treści

1.	Architektura	3
2.	Dostosowanie aplikacji backendowej	5
3.	Apache Kafka	11
4.	Narzędzie low-code n8n	15
5.	Wdrożenie.....	19
6.	Wnioski	22

1. Architektura

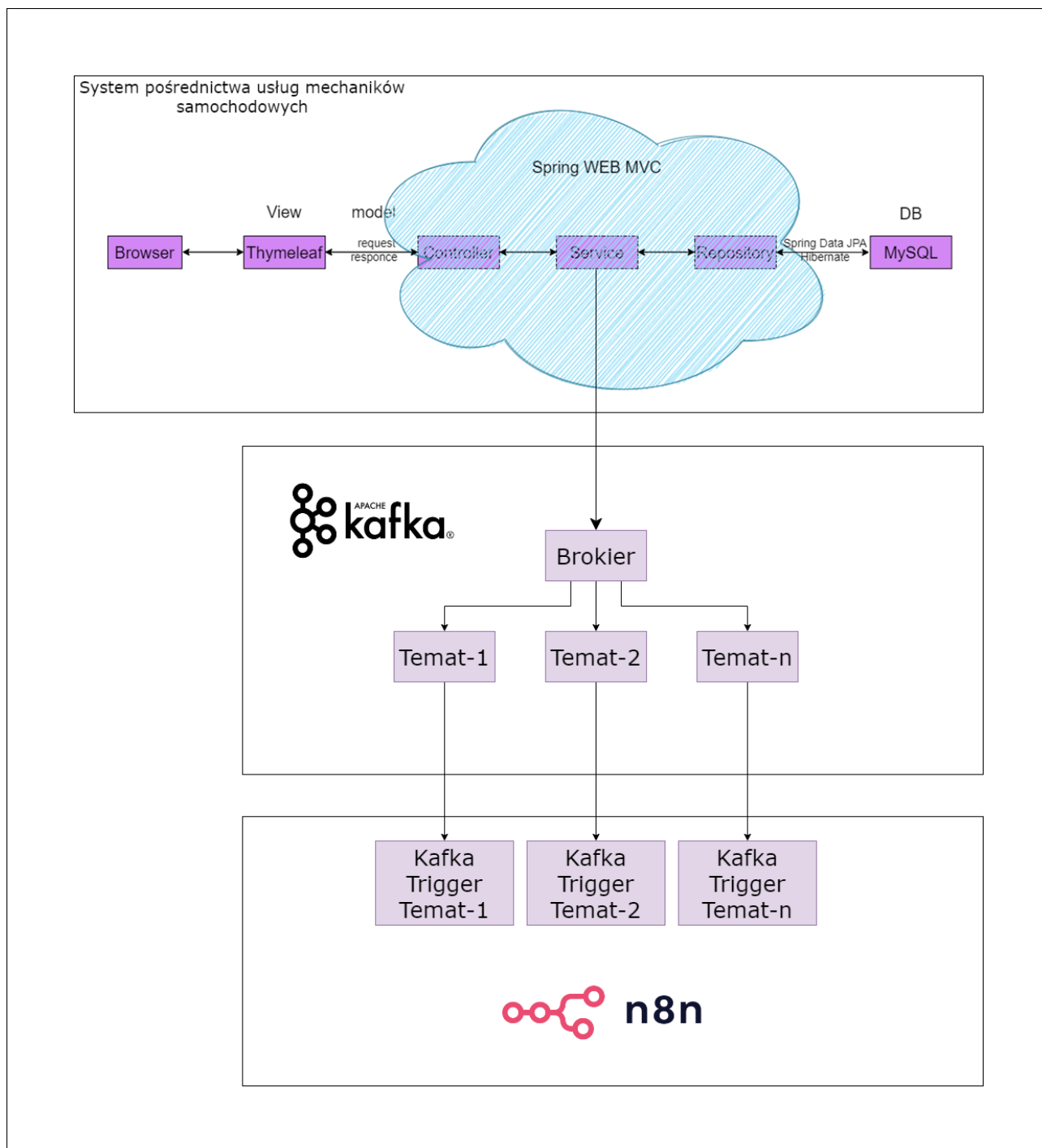
Projekt zakłada implementację automatyzacji systemu powiadomień przy użyciu narzędzi low-code n8n. Pierwszym krokiem jest rozbudowa istniejącej aplikacji backendowej opartej na technologii Java/Spring o funkcjonalność umożliwiającą integrację z narzędziem n8n. Koncepcja architektury zakłada dostosowanie backendu poprzez dodanie generowania zdarzeń, które następnie będą przekazywane do narzędzia n8n w celu obsługi. Na podstawie nasłuchanych zdarzeń blokowy interfejs narzędzia n8n zostanie wykorzystany do automatyzacji wysyłania powiadomień z minimalizacją pisania kodu.

Aby zrealizować architekturę, należy rozbudować aplikację backendową o funkcjonalność generowania zdarzeń w odpowiednich miejscach biznes logiki aplikacji. Do tego celu wykorzystany zostanie broker zdarzeń Apache Kafka.

Kolejnym krokiem jest obsługa wygenerowanych zdarzeń po stronie serwera Kafka oraz umożliwienie dostępu do nich z narzędzia n8n. W tym celu wybrany został projekt coductor.io, który zapewnia interfejs do konfiguracji Kafki.

Finalnym elementem architektury jest narzędzie n8n z modułami nasłuchującymi odpowiednie tematy na brokerze Kafka. Na podstawie odebranych zdarzeń zapewniona zostanie automatyzacja systemu powiadomień.

Rysunek 1.1 przedstawia wizualizację opisaną architekturę.



Rys. 1.1. Architektura projektu

2. Dostosowanie aplikacji backendowej

Projekt wykorzystuje aplikację do pośrednictwa usług mechaników samochodowych w celu przedstawienia architektury oraz umożliwienia automatyzacji systemu powiadomień. Część backendowa tej aplikacji została napisana w technologii Java/Spring, a dostosowanie do potrzeb projektu wymaga jedynie dodania funkcjonalności generowania zdarzeń.

Integracja brokera Kafka z projektem, który ma backend napisany w frameworku Spring, będzie zrealizowana przy użyciu modelu Spring-Kafka. Ten proces polega na dodaniu odpowiedniego kodu do pliku pom.xml. Na Rysunku 2.1 przedstawiono fragment pliku pom.xml z zdefiniowanymi zależnościami Spring-Kafka.

```
20
21     <dependencies>
22     @t     <dependency>
23           <groupId>org.springframework.boot</groupId>
24           <artifactId>spring-boot-starter-data-jpa</artifactId>
25     </dependency>
26     @t     <dependency>
27           <groupId>org.springframework.boot</groupId>
28           <artifactId>spring-boot-starter-thymeleaf</artifactId>
29     </dependency>
30     @t     <dependency>
31           <groupId>org.springframework.boot</groupId>
32           <artifactId>spring-boot-starter-web</artifactId>
33     </dependency>
34     @t     <dependency>
35           <groupId>org.springframework.boot</groupId>
36           <artifactId>spring-boot-starter-security</artifactId>
37     </dependency>
38     @t     <dependency>
39           <groupId>org.springframework.boot</groupId>
40           <artifactId>spring-boot-starter-jdbc</artifactId>
41     </dependency>
42     @t     <dependency>
43           <groupId>mysql</groupId>
44           <artifactId>mysql-connector-java</artifactId>
45           <scope>runtime</scope>
46     </dependency>
47     @t     <dependency>
48           <groupId>org.springframework.kafka</groupId>
49           <artifactId>spring-kafka</artifactId>
50     </dependency>
51     @t     <dependency>
52           <groupId>org.springframework.boot</groupId>
53           <artifactId>spring-boot-starter-mail</artifactId>
54     </dependency>
55 </dependencies>
56
57 <build>
58   <plugins>
59     <plugin>
60       <groupId>org.springframework.boot</groupId>
61       <artifactId>spring-boot-maven-plugin</artifactId>
62       <version>2.3.0.RELEASE</version>
63     </plugin>
64   </plugins>
65 </build>
66
```

Rys. 2.1. Plik konfiguracyjny pom.xml

Po zdefiniowaniu modułu, pierwszym krokiem jest dodanie klasy konfiguracyjnej `KafkaProducerConfig` w celu zdefiniowania producenta Kafka.

```

15  /**
16   * Zwraca obiekt typu ProducerFactory<String, Object>.
17   * Ten obiekt jest używany do konfiguracji i utworzenia instancji producenta Kafka.
18   * Tworząc kolejne obiekty producentów korzystamy z ProducerFactory
19   * Klucz('String') - określa do której partycji w danym temacie zostanie przekazana wiadomość
20   * Wartość('Object') - zawartość wiadomości w postaci słownika (obecnie Object w praktyce Map)
21
22   * Obiekt KafkaTemplate jest gotową klasą Springa ułatwiającą wysyłanie wiadomości do brokera Kafka
23   * przy użyciu producenta Kafka skonfigurowanego wcześniej przez producerFactory()
24   */
25
26  @Configuration
27  public class KafkaProducerConfig {
28      @Bean
29      public ProducerFactory<String, Object> producerFactory() {
30          Map<String, Object> configProps = new HashMap<>();
31          // configProps.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:19092");
32          configProps.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "redpanda-0:9092");
33          configProps.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
34          configProps.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, JsonSerializer.class);
35          return new DefaultKafkaProducerFactory<>(configProps);
36      }
37
38      @Bean
39      > public KafkaTemplate<String, Object> kafkaTemplate() { return new KafkaTemplate<>(producerFactory()); }
42  }
43  |

```

Rys. 2.2. Klasa konfiguracyjna `KafkaProducerConfig`

Klasa `KafkaProducerConfig` odpowiada za konfigurację i tworzenie instancji producenta Kafka w kontekście aplikacji Spring. Udostępnia obiekty typu `ProducerFactory<String, Object>` oraz `KafkaTemplate<String, Object>`, ułatwiające wysyłanie wiadomości do brokera Kafka.

Metoda `producerFactory()` zwraca obiekt typu `ProducerFactory<String, Object>`, który służy do konfiguracji i tworzenia instancji producenta Kafka. Klasa ta przyjmuje parametry konfiguracyjne, takie jak serwery startowe, serializatory klucza i wartości, a następnie inicjalizuje fabrykę producentów Kafka.

Metoda `kafkaTemplate()` zwraca gotowy do użycia obiekt `KafkaTemplate<String, Object>`, wykorzystujący skonfigurowaną wcześniej fabrykę producentów. Umożliwia to łatwe wysyłanie wiadomości do brokera Kafka z poziomu aplikacji.

Konfiguracja obejmuje ustawienia, takie jak serwery startowe, serializatory klucza i wartości. Cały projekt zostanie wdrożony przy użyciu kontenerów Docker, dla tego jako serwer podano nazwę kontenera z brokerem Kafka.

Teraz, korzystając z zdefiniowanego szablonu, generowanie zdarzeń zostanie dodane w odpowiednie miejsca w biznesowej logice aplikacji.

System pośrednictwa usług mechaników samochodowych zapewnia takie dwie główne role:

- Klient: Osoba poszukująca wykonawcę usługi związanej z mechaniką samochodową.
- Warsztat Samochodowy: Oferuje usługi w zakresie mechaniki samochodowej.

Funkcjonalność systemu pośrednictwa usług mechaników samochodowych zostanie wykorzystana do generowania odpowiednich zdarzeń. Poniżej opisane są endpointy, w których zdarzenia będą generowane.

Tworzenie zlecenia przez klienta

Klient tworzy nowe zlecenie w celu poszukiwania wykonawcy usługi związanej z mechaniką samochodową. Wytworzone zlecenia zawiera:

- Dane kontaktowe klienta;
- Dane dotyczące samochodu;
- Opis żądanej usługi.

Stworzone zlecenia trafiają na panel użytkownika warsztatów samochodowych.

Podczas tworzenia zlecenia należy zapewnić generowanie wiadomości oraz wysłanie jej jako zdarzenia na odpowiedni temat w Kafka. Wysyłana wiadomość powinna zawierać wszystkie dane potrzebne narzędziu n8n do obsługi i wysyłania powiadomień do klientów. W tym punkcie wszystkie warsztaty samochodowe powinny zostać poinformowane o powstaniu nowego zlecenia.

Temat, na który zostanie wysłane zdarzenie, nosi nazwę „createOrder”. Rysunek 2.3 przedstawia fragment kodu odpowiedzialny za utworzenie słownika zawierającego wszystkie niezbędne dane oraz wysłanie go jako zdarzenia na właściwy temat.

```
96
97      // Generate event new order
98      Map<String, String> messageData = new HashMap<>();
99      messageData.put("CarMake", order.getCar().getCarModel().getCarMake().getMake());
100     messageData.put("CarModel", order.getCar().getCarModel().getModel());
101     messageData.put("Year", order.getCar().getYear().toString());
102     messageData.put("Description", order.getDescription());
103
104     messageData.put("email", order.getCustomerDetail().getEmail());
105     messageData.put("PhoneNumber", order.getCustomerDetail().getPhoneNumber());
106
107     messageData.put("WorkshopEmailList",
108         order.getCity().getWorkshops().stream() Stream<Workshop>
109             .map(Workshop::getEmail) Stream<String>
110             .collect(Collectors.toList()).toString());
111     messageData.put("WorkshopPhoneNumberList",
112         order.getCity().getWorkshops().stream() Stream<Workshop>
113             .map(Workshop::getPhoneNumber) Stream<String>
114             .collect(Collectors.toList()).toString());
115
116     kafkaTemplate.send(createOrderTopic, messageData);
117 }
```

Rys. 2.3. Fragment klasy OrderServiceImpl

Rysunek 2.4 przedstawia tworzenie zlecenia od strony użytkownika serwisu.

Nowe zlecenie

BMW

M3

2023

Diesel

AT0777BP

ZFA26300006L92380

Opis naprawy

Wymiana oleju.

Rzeszów

▼

Rys. 2.4. Forma tworzenia zlecenia

Zaproponowanie oferty przez warsztat samochodowy

Stworzone zlecenia trafiają do warsztatów samochodowych, gdzie mogą one przedstawić swoje warunki dotyczące realizacji zlecenia. Ten proces jest nazywany tworzeniem ofert. Oferta w odpowiedzi na zlecenie zawiera planowany termin wykonania zlecenia oraz szacowaną cenę, jaką proponuje warsztat samochodowy. Rysunek 2.5 przedstawia, jak wygląda panel użytkownika warsztatu samochodowego z widokiem umożliwiającym zaoferowanie warunków realizacji dostępnych zleceń.

IREMONT		Nowe zlecenia Bieżące zlecenia Archiwum zleceń							Wyloguj się		
#	Data stworzenia	Marka	Model	Rok	Imię i Nazwisko	Numer telefonu	Data Realizacji	Cena			
1	2024-01-24T13:53	BMW	M3	2023	Mykola Mykola	+48791863388	дд.мм.пппп		Zaoferować	Opcje	Usuń
2	2024-01-24T14:06	Mercedes-Benz	C63 AMG	2024	Vasyl Vynnychuk	+48791863380	дд.мм.пппп		Zaoferować	Opcje	Usuń

Rys. 2.5. Panel użytkownika warsztatu samochodowego

To jest odpowiednie miejsce do generowania zdarzenia, gdy warsztat samochodowy tworzy ofertę. Klient powinien być informowany o nowej ofercie w odpowiedzi na jego zlecenie. Do klienta powinna być dostarczona informacja dotycząca warunków realizacji zlecenia oraz dane kontaktowe warsztatu samochodowego. Rysunek 2.6 przedstawia fragment kodu z klasy logiki biznesowej do tworzenia oferty, obejmujący zarówno utworzenie, jak i wysłanie wiadomości o zdarzeniu na odpowiednią temat w Kafka. W tym przypadku temat nazywa się „sendOffer”.

```

118 // Sending email with offer
119 Map<String, String> messageData = new HashMap<>();
120
121 messageData.put("WorkshopEmail", orderAnswer.getWorkshop().getEmail());
122 messageData.put("WorkshopPhoneNumber", orderAnswer.getWorkshop().getPhoneNumber());
123 messageData.put("WorkshopAddress", orderAnswer.getWorkshop().getAddress());
124
125 messageData.put("CarRegistry", orderAnswer.getOrder().getCar().getRegistrationNumber());
126 messageData.put("ImplementationDate", orderAnswer.getImplementationDate().toString());
127 messageData.put("Price", String.valueOf(orderAnswer.getPrice()));
128
129 messageData.put("CustomerEmail", orderAnswer.getOrder().getCustomerDetail().getEmail());
130 messageData.put("CustomerPhoneNumber", orderAnswer.getOrder().getCustomerDetail().getPhoneNumber());
131
132 kafkaTemplate.send(sendOfferTopic, messageData);
133 sendStanChangeEmail(orderAnswer);
134 }

```

Rys. 2.6. Fragment klasy OrderAnswerServiceImpl, tworzenie oferty

Wybór oferty przez klienta

Klient otrzymuje oferty realizacji zlecenia od różnych warsztatów samochodowych i może je porównać w odpowiedniej zakładce na panelu użytkownika. Rysunek 2.7 przedstawia widok panelu użytkownika, gdzie klient może dokonać wyboru najbardziej odpowiadającej mu oferty.

IREMONT							Wstecz	Wyloguj się
#	Nazwa autoserwisu	Miasto	adres	Numer telefonu	Data realizacji	Cena		
1	Bosch Service	Rzeszów	ul. Bławatkowa 18	+48791863384	2024-01-25	400.0	Wybrać	
2	M-ka	Rzeszów	ul. Bławatkowa 18	+48791863383	2024-01-24	350.0	Wybrać	

Rys. 2.7. Panel użytkownika klienta

Kiedy klient dokona wyboru oferty, warsztat samochodowy zostanie o tym poinformowany. W tym miejscu, w odpowiedniej klasie logiki biznesowej, generowana jest wiadomość zawierająca dane kontaktowe klienta oraz informacje o samochodzie. Wiadomość zostaje wysłana jako zdarzenie na temat Kafka o nazwie „choseOffer”. Rysunek 2.8 przedstawia fragment kodu służący do tego celu.

```

146 // Customer chose offer
147 Map<String, String> messageData = new HashMap<>();
148
149 messageData.put("CustomerEmail", orderAnswer.getOrder().getCustomerDetail().getEmail());
150 messageData.put("CustomerPhoneNumber", orderAnswer.getOrder().getCustomerDetail().getPhoneNumber());
151
152 messageData.put("CarRegistration", orderAnswer.getOrder().getCar().getRegistrationNumber());
153 messageData.put("CarMake", orderAnswer.getOrder().getCar().getCarModel().getCarMake().getMake());
154 messageData.put("CarModel", orderAnswer.getOrder().getCar().getCarModel().getModel());
155 messageData.put("CarYear", orderAnswer.getOrder().getCar().getYear().toString());
156
157 messageData.put("WorkshopEmail", orderAnswer.getWorkshop().getEmail());
158 messageData.put("WorkshopPhoneNumber", orderAnswer.getWorkshop().getPhoneNumber());
159
160 kafkaTemplate.send(choseOfferTopic, messageData);
161 sendStanChangeEmail(orderAnswer);
162 }

```

Rys. 2.8. Fragment klasy OrderAnswerServiceImpl, wybór oferty

W przedstawionych wyżej przypadkach generowane są zdarzenia, które są przesyłane do brokera Kafka. Podsumowując, zlecenia zawierają treść w postaci mapy klucz-wartość i są przesyłane jako format JSON.

Generowane są zdarzenia na trzy tematy:

createOrder: wytworzono nowe zlecenie,

sendOffer: zaproponowano ofertę realizacji zlecenia,

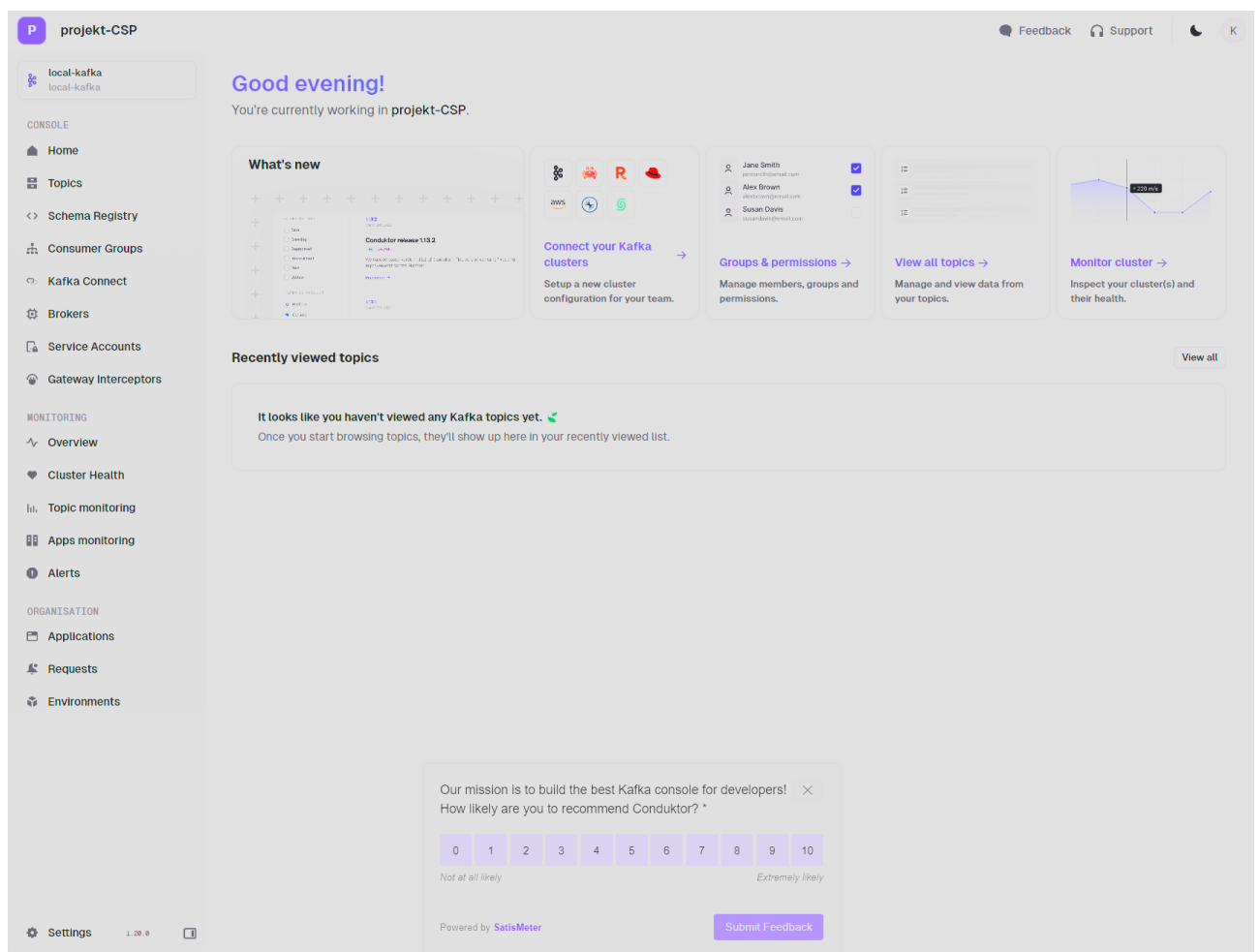
choseOffer: wybrano warsztat do realizacji zlecenia.

3. Apache Kafka

Wygenerowane zdarzenia muszą być obsługiwane, a do tego celu wymagany jest serwer Apache Kafka. Niemniej jednak celem całego projektu jest wykorzystanie narzędzi low-code. Aby umożliwić zdefiniowanie potrzebnych tematów po stronie Kafki bez konieczności pisania kodu został użyty projekt Conduktor.io, który pozwala również na wyświetlenie obsługiwanych zdarzeń.

Conduktor.io pełni kluczową rolę w ułatwianiu współpracy z Apache Kafka, oferując wygodne narzędzie oparte na interfejsie graficznym. To narzędzie istotnie ułatwia pracę z Apache Kafka, abstrahując od konieczności implementacji w konkretnym języku programowania. Dzięki temu użytkownicy mogą korzystać z niego niezależnie od swoich preferencji programistycznych, co czyni cały proces bardziej dostępnym i efektywnym. Conduktor.io umożliwia również łatwe monitorowanie i wyświetlanie obsługiwanych zdarzeń, co przyczynia się do efektywnej administracji systemem komunikacyjnym Apache Kafka.

Rysunek 3.1 przedstawia interfejs użytkownika conduktor.io.



Rys. 3.1. conduktor.io interfejs użytkownika

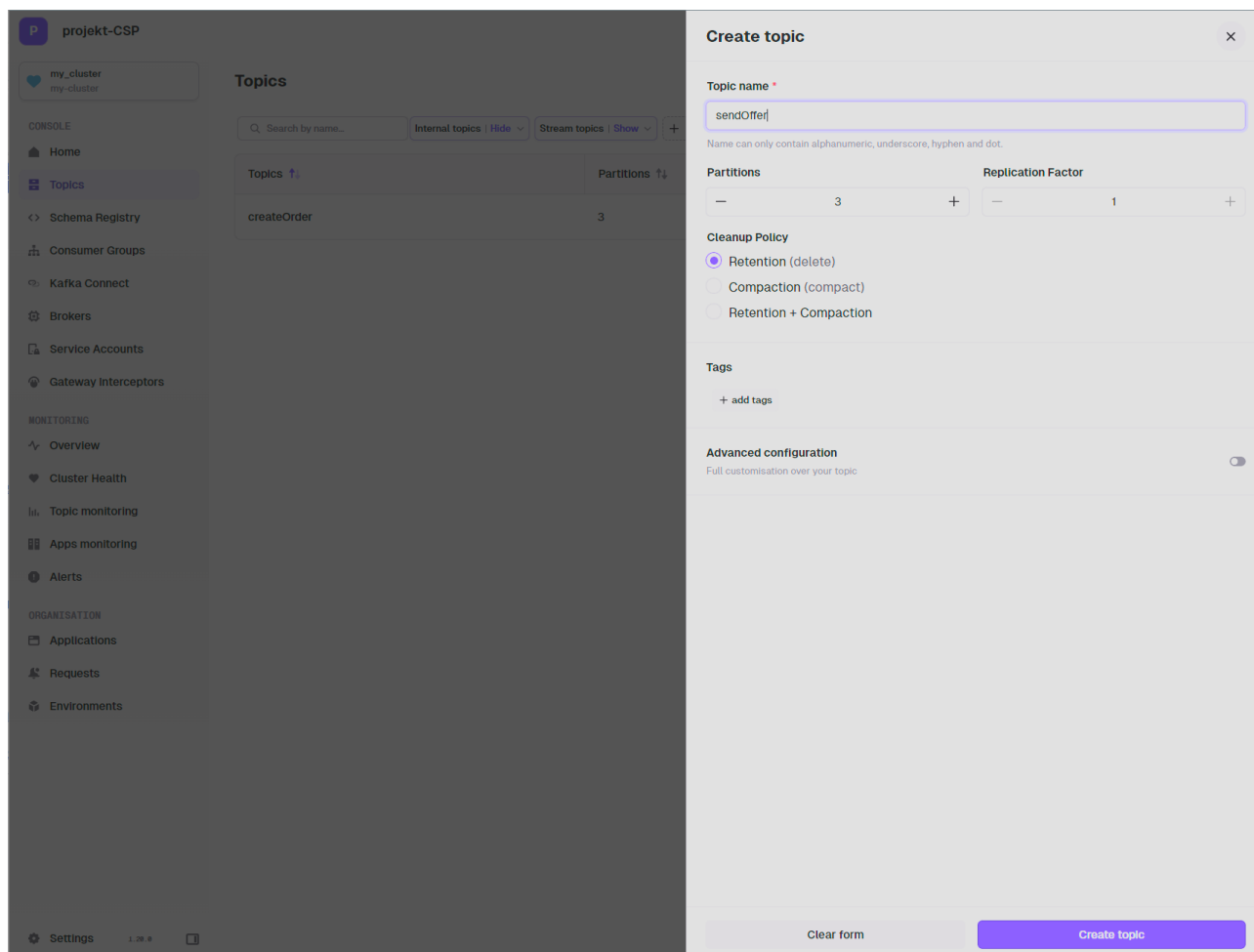
Pierwszym krokiem było zdefiniowanie własnego klastra. Pojęcie 'cluster' odnosi się do grupy serwerów Apache Kafka, które są skonfigurowane i zarządzane jako jednostka. Klastry w Conduktor.io reprezentują całe zespoły powiązanych ze sobą brokerów Apache Kafka, współpracujących w celu udostępniania i przetwarzania wiadomości. Projekt Conduktor.io oferuje możliwość utworzenia trzech klastrów w ramach bezpłatnego planu, natomiast dostęp do kolejnych jest dostępny wyłącznie po dokonaniu uaktualnienia planu.

Rysunek 3.2 przedstawia widok okna dodania nowego klastra.

The screenshot displays the 'New cluster configuration' interface in the Conductor.io console. The left sidebar contains navigation links for 'Back', 'Clusters', 'Certificates', 'Data Policies', 'YOUR ORGANISATION', 'User & Groups', 'Audit Log', 'Integrations', 'Api Keys', 'Plan', 'ACCOUNT', and 'Profile'. The main content area is titled 'New cluster configuration' and includes a help link. Below this, there are tabs for different cluster types: Aiven, Confluent, MSK, Redpanda, and Apache Kafka. The configuration form includes fields for 'Cluster icon & name' (with a dropdown and a text input containing 'Imie nie wpływa na łączenie'), 'Technical ID' (with a text input containing 'imie-nie-wp-ywa-na-aczenie' and a 'Random ID' button), and 'Bootstrap servers' (with a text input containing 'redpanda-0-9092'). The 'Authentication method' section shows four radio buttons: 'None' (selected), 'SASL', 'SSL', and 'AWS IAM'. Below this is an 'Advanced properties' section with a dropdown arrow. At the bottom of the form, there is a 'Test connection' button, a 'Connected' status indicator, a 'Cancel' button, and a 'Create Configuration' button. A feedback survey is visible at the bottom of the page, asking 'How likely are you to recommend Conductor?' with a scale from 0 to 10.

Rys. 3.2. conductor.io dodanie klastra

W ramach klastra definiuje się tematy, które będą nasłuchiwane. W rozdziale 2 dokumentacji wymieniono tematy, na które generowane są zdarzenia w aplikacji backendowej. Rysunek 3.3 przedstawia widok okna dodawania nowego tematu.



Rys. 3.3. conductor.io dodanie tematu

Po zdefiniowaniu tematów, conductor.io umożliwia także deserializację oraz obserwację wiadomości zdarzeń jak również całej historii zdarzeń, co przedstawiono na rysunku 3.4.

projekt-CSP

my_cluster
my-cluster

CONSOLE

Home

Topics

Schema Registry

Consumer Groups

Kafka Connect

Brokers

Service Accounts

Gateway Interceptors

MONITORING

Overview

Cluster Health

Topic monitoring

Apps monitoring

Alerts

ORGANISATION

Applications

Requests

Environments

Settings

Topics > createOrder

RECORDS 20 TOPIC SIZE 13 KiB REPLICATION FACTOR 1 < MIN ISR N/A IS COMPACTED

Consume Produce Configuration Consumer Groups Schema ACL

Show From Most recent Limit Number of records (500) Partitions all

Timestamp (Local)	Key
2024-01-27 02:17:36	null
2024-01-23 19:34:26	null
2024-01-23 19:21:30	null
2024-01-23 16:56:05	null
2024-01-23 16:54:52	null
2024-01-23 16:54:48	null
2024-01-23 16:54:46	null
2024-01-23 16:54:40	null
2024-01-23 16:54:28	null
2024-01-23 16:54:26	null
2024-01-23 16:54:11	null
2024-01-23 16:54:03	null
2024-01-23 16:53:40	null
2024-01-23 16:50:50	null
2024-01-23 16:50:16	null
2024-01-23 16:47:16	null
2024-01-23 16:46:16	null
2024-01-23 16:41:21	null
2024-01-23 16:39:32	null
2024-01-23 16:38:38	null

Data Headers Metadata

KEY

1 null

VALUE JSON

Field	Value
CarMake	BMW
CarModel	M3
Description	Wymiana oleju.
Year	2024
WorkshopEmailList	[mnykolaichuk96@gmail.com, 150174@stud.piz.edu.pl]
PhoneNumber	+48791863388
WorkshopPhoneNumberList	[+48791863383, +48791863384]
email	kolya5179596@gmail.com

Copy all Reprocess message

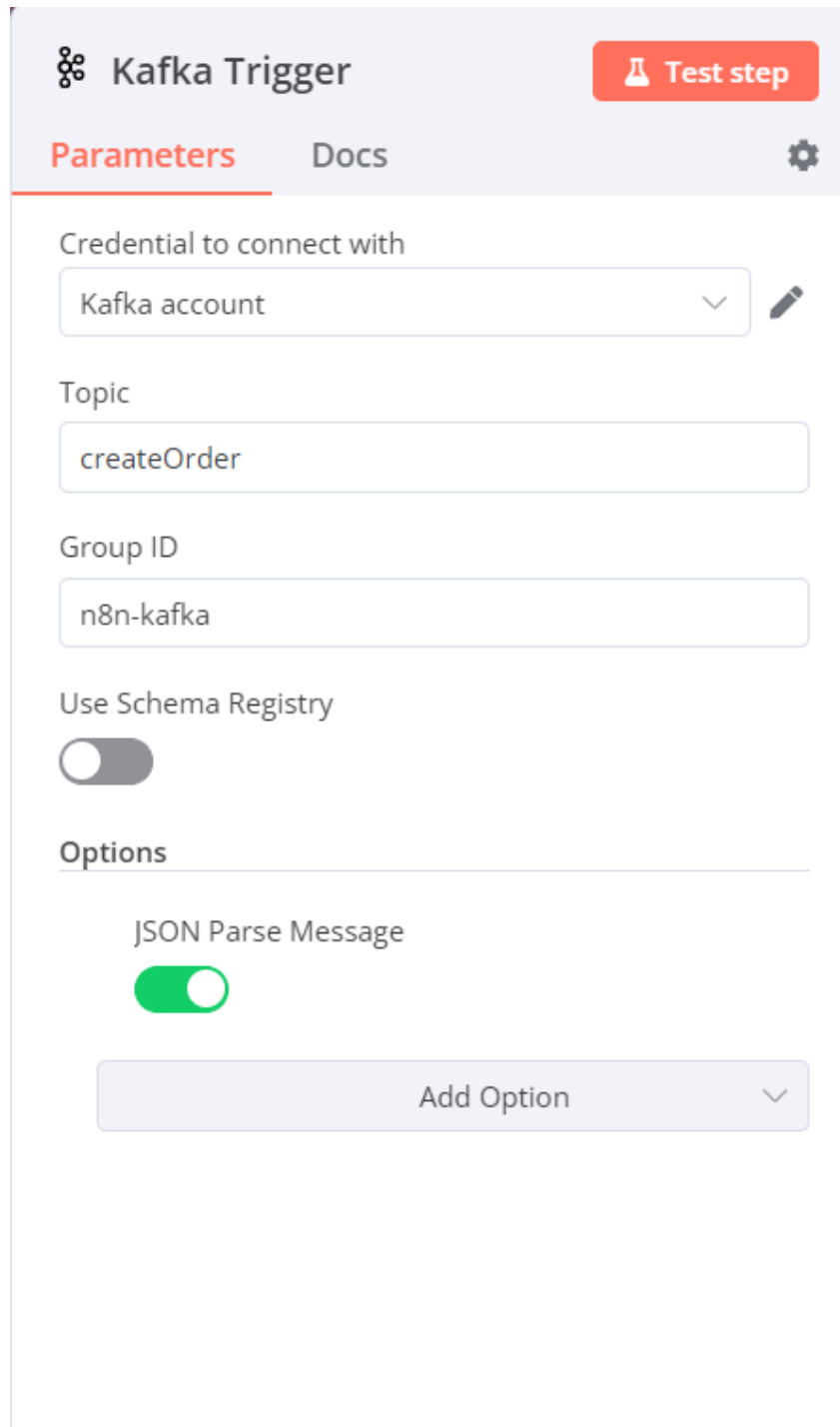
Rys. 3.4. konduktor.io podgląd tematu

Podsumowując, przy użyciu Apache Kafka, zdarzenia są najpierw generowane po stronie serwera aplikacji backendowej, następnie przesyłane i przechowywane w odpowiednich tematach na serwerze Kafka. Są one dostępne dla konsumentów do nasłuchiwania. W ramach rozpatrywanego w projekcie przypadku, konsumentami będą odpowiednie moduły n8n, co zostanie przedstawione w dalszych rozdziałach.

4. Narzędzie low-code n8n

Narzędzie n8n umożliwia automatyzację powiadomień w ramach projektu. n8n nasłuchuje odpowiednich tematów na brokerze Kafka, a po otrzymaniu wiadomości zdarzenia wysyła odpowiednie komunikaty do klientów.

Do nasłuchiwania tematów wykorzystano dedykowany moduł o nazwie Kafka Trigger. Rysunek 4.1 przedstawia konfigurację tego modułu.



The screenshot displays the configuration interface for the 'Kafka Trigger' module in n8n. At the top, the module name 'Kafka Trigger' is shown next to a red 'Test step' button. Below this, there are two tabs: 'Parameters' (active) and 'Docs'. A settings gear icon is located to the right of the tabs. The 'Parameters' section contains the following fields and controls:

- Credential to connect with:** A dropdown menu showing 'Kafka account' with a pencil icon for editing.
- Topic:** A text input field containing 'createOrder'.
- Group ID:** A text input field containing 'n8n-kafka'.
- Use Schema Registry:** A toggle switch currently turned off.
- Options:** A section header with a horizontal line below it.
- JSON Parse Message:** A toggle switch currently turned on (green).
- Add Option:** A button with a dropdown arrow.

Rys. 4.1. n8n moduł Kafka Trigger

W celu umożliwienia połączenia między serwerem Kafka a serwerem n8n, konieczne jest skonfigurowanie odpowiednich danych uwierzytelniających. Rysunek 4.2 przedstawia konfigurację okna dodawania credentials.



Connection

✔ Connection tested successfully

Retry

Sharing

Details

Need help filling out these fields? [Open docs](#)

Client ID

my-app

Will not affect the connection, but will be used to identify the client in the Kafka server logs. Read more [here](#)

Brokers

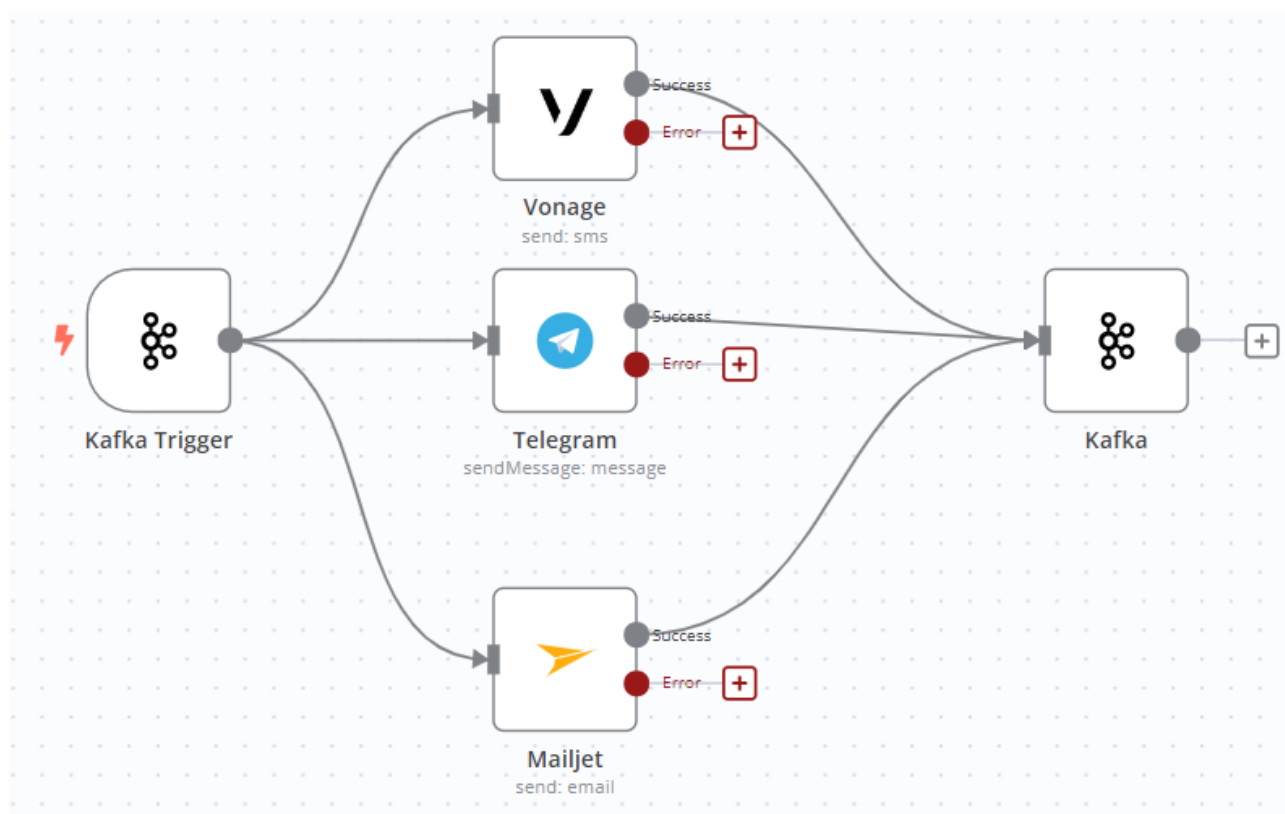
redpanda-0:9092

SSL



Rys. 4.2. n8n credential

Odebrane zdarzenie musi być następnie obsłużone w celu zautomatyzowania odpowiednich wiadomości. Rysunek 4.3 przedstawia widok schematu workflow, który zapewnia wysłanie wiadomości przez trzy kanały komunikacyjne: SMS, Telegram, E-Mail. Potwierdzenia wysłania z poszczególnych modułów komunikacyjnych trafiają do modułu Kafka, który generuje potwierdzenie odebrania oraz wysyła o tym wiadomość na odpowiedni temat.



Rys. 4.3. n8n workflow

Treść wiadomości składa się z tekstu wpisanego przez autorów projektu oraz sparsowanych z otrzymanej wiadomości danych. Obsługa wysyłania wiadomości na poziomie narzędzia n8n umożliwia generowanie spersonalizowanych komunikatów bez dodatkowego pisania kodu. Rysunek 4.4 przedstawia widok okna modułu komunikacji Telegram.

The screenshot shows the configuration interface for the Telegram module in n8n. The interface is divided into two tabs: 'Parameters' (active) and 'Docs'. A red 'Test step' button is located in the top right corner. The configuration fields include:

- Credential to connect with:** A dropdown menu set to 'Telegram account'.
- Resource:** A dropdown menu set to 'Message'.
- Operation:** A dropdown menu set to 'Send Message'.
- Chat ID:** A text input field containing '5854450473'. To its right are two buttons: 'Fixed' (selected) and 'Expression'.
- Text:** A large text area containing a JSON template: `{{ $json.message.CarMake }}` `{{ $json.message.CarModel }}` z `{{ $json.message.Year }}`. Poniżej zamieszczam opis problemu: `{{ $json.message.Description }}`. Below the text area, an error message is displayed: `[ERROR: No data found for item-index: "0"]`.
- Reply Markup:** A dropdown menu set to 'None'.
- Additional Fields:** A section for additional configuration options, currently empty.

Rys. 4.4. n8n moduł Telegram

Moduły komunikacyjne są połączone z modułem Kafka w celu potwierdzenia wysłania wiadomości. Rysunek 4.5 przedstawia widok okna modułu Kafka z wysłaniem potwierdzenia na odpowiedni temat.

The screenshot shows the configuration for the 'Kafka' node in n8n. At the top, there's a 'Test step' button. Below it are tabs for 'Parameters' (selected) and 'Docs'. The 'Credential to connect with' dropdown is set to 'Kafka account'. The 'Topic' field is set to an expression: `{{ $('Kafka Trigger').item.json.topic + 'SendConfirmation' }}`. Below this, an error message states: '[ERROR: no data, execute "Kafka Trigger" node first]'. There are four toggle switches: 'Send Input Data' (turned on), 'JSON Parameters' (turned off), 'Use Schema Registry' (turned off), and 'Use Key' (turned off). At the bottom, there's a section for 'Headers' with the message 'Currently no items exist' and an 'Add Header' button.

Rys. 4.5. n8n moduł Kafka

Ponieważ nasłuchujemy trzech tematów, potrzebujemy również trzech Workflow zgodnych z tym, co zostało przedstawione. Rysunek 4.6 przedstawia widok n8n z zakładki Workflows, gdzie zdefiniowano i aktywowano wszystkie potrzebne Workflow.

The screenshot shows the 'Workflows' tab in n8n. There's a search bar and a sort dropdown set to 'Sort by last updated'. An 'Add Workflow' button is on the left. The table lists three workflows, all of which are active.

Workflow Name	Last updated	Created	Status
Kafka createOrder	12 minutes ago	23 January	Active
Kafka sendOffer	53 minutes ago	24 January	Active
Kafka choseOffer	53 minutes ago	24 January	Active

Rys. 4.6. n8n Workflows

5. Wdrożenie

Wdrożenie projektu zostało zrealizowane przy użyciu kontenerów Docker i pliku docker-compose. Umożliwia to uruchomienie całego projektu za pomocą jednego polecenia docker-compose up. Wewnątrz pliku docker-compose znajdują się instrukcje uruchomienia dla każdego z kontenerów.

Autorzy podzielili projekt na trzy główne części:

aplikacja backendowa

Backend został skonteneryzowany przy użyciu narzędzia Maven. Po zbudowaniu pliku JAR, komenda mvn create image tworzy obraz kontenera z aplikacją wewnątrz. Kontener bazy danych został utworzony na podstawie kontenera MySQL, a następnie uruchomiono odpowiedni skrypt SQL z bazą danych wewnątrz kontenera. Dodatkowo, w celu wyświetlania oraz zarządzania bazą danych, uruchomiono kontener z phpMyAdmin.

conduktor.io

Projekt Conduktor.io udostępnia w dokumentacji do projektu plik docker-compose, zawierający instrukcję dla wszystkich potrzebnych kontenerów. Jediną zmianą w porównaniu do pliku zaproponowanego w instrukcji było usunięcie kontenera generate_data, ponieważ służy on tylko do generowania testowych danych, co jest niezwiązane z celem projektu.

n8n

Narzędzie n8n również jest dostępne w wersji do wdrożenia przy użyciu Dockera. W dokumentacji n8n zamieszczona jest komenda do uruchomienia kontenera z n8n. W celu uruchomienia całego projektu za pomocą jednego pliku docker-compose, na podstawie obrazu z n8n oraz komendy do uruchomienia, został dodany odpowiedni serwis w pliku docker-compose.

Na rysunku 5.1 oraz 5.2 przedstawiono plik docker-compose zawierający wszystkie opisane wyżej instrukcje.

```
1  version: '3.8'
2
3  services:
4    db:
5      image: mnykolaichuk/praca-dyplomowa-mysql-db:2023
6      container_name: db
7      environment:
8        - MYSQL_ROOT_PASSWORD=0985179596
9        - MYSQL_DATABASE=iremونت.pl
10       DATABASE_HOST=host.docker.internal
11      ports:
12        - "3306:3306"
13      restart: always
14
15   phpmyadmin:
16     image: phpmyadmin/phpmyadmin:latest
17     container_name: my-php-myadmin
18     ports:
19       - "8082:80"
20     restart: always
21     depends_on:
22       - db
23     environment:
24       SPRING_DATASOURCE_USERNAME: root
25       SPRING_DATASOURCE_PASSWORD: 0985179596
26
27   iremont:
28     image: mnykolaichuk/praca-dyplomowa-app:2024-CSP-V5
29     container_name: iremont-app
30     ports:
31       - "8083:8083"
32     restart: always
33     depends_on:
34       - db
35     environment:
36       - DATABASE_HOST=db
37       - DATABASE_USER=root
38       - DATABASE_PASSWORD=0985179596
39       - DATABASE_NAME=iremونت.pl
40       - DATABASE_PORT=3306
41
```

Rys. 5.1. Fragment pliku docker-compose.yml

























































```

42 postgresql:
43   image: postgres:14
44   hostname: postgresql
45   volumes:
46     - pg_data:/var/lib/postgresql/data
47   environment:
48     PGDATA: "/var/lib/postgresql/data"
49     POSTGRES_DB: "konduktor-platform"
50     POSTGRES_USER: "konduktor"
51     POSTGRES_PASSWORD: "change_me"
52     POSTGRES_HOST_AUTH_METHOD: "scram-sha-256"
53
54 konduktor-platform:
55   image: konduktor/konduktor-platform:1.20.0
56   depends_on:
57     - postgresql
58     - redpanda-0
59   ports:
60     - "8080:8080"
61   volumes:
62     - konduktor_data:/var/konduktor
63   environment:
64     CDK_DATABASE_URL: "postgresql://konduktor:change_me@postgresql:5432/konduktor-platform"
65     CDK_CLUSTERS_0_ID: "local-kafka"
66     CDK_CLUSTERS_0_NAME: "local-kafka"
67     CDK_CLUSTERS_0_BOOTSTRAPSERVERS: "redpanda-0:9092"
68     CDK_CLUSTERS_0_SCHEMAREGISTRY_URL: "http://redpanda-0:18081"
69     CDK_CLUSTERS_0_COLOR: "#6A57C8"
70     CDK_CLUSTERS_0_ICON: "kafka"
71     CDK_CLUSTERS_1_ID: "cdk-gateway"
72     CDK_CLUSTERS_1_NAME: "cdk-gateway"
73     CDK_CLUSTERS_1_BOOTSTRAPSERVERS: "konduktor-gateway:6969"
74     CDK_CLUSTERS_1_SCHEMAREGISTRY_URL: "http://redpanda-0:18081"
75     CDK_CLUSTERS_1_KAFKAFLAVOR_URL: "http://konduktor-gateway:8888"
76     CDK_CLUSTERS_1_KAFKAFLAVOR_USER: "admin"
77     CDK_CLUSTERS_1_KAFKAFLAVOR_PASSWORD: "konduktor"
78     CDK_CLUSTERS_1_KAFKAFLAVOR_VIRTUALCLUSTER: "passthrough"
79     CDK_CLUSTERS_1_KAFKAFLAVOR_TYPE: "Gateway"
80     CDK_CLUSTERS_1_COLOR: "#6A57C8"
81     CDK_CLUSTERS_1_ICON: "dog"
82     CDK_MONITORING_CORTEX_URL: http://konduktor-monitoring:9009/
83     CDK_MONITORING_ALERT_MANAGER_URL: http://konduktor-monitoring:9010/
84     CDK_MONITORING_CALLBACK_URL: http://konduktor-platform:8080/monitoring/api/
85     CDK_MONITORING_NOTIFICATIONS_CALLBACK_URL: http://localhost:8080
86
87 konduktor-monitoring:
88   image: konduktor/konduktor-platform-cortex:1.20.0
89   environment:
90     CDK_CONSOLE_URL: "http://konduktor-platform:8080"
91
92 redpanda-0:
93   command:
94     - redpanda
95     - start
96     - --kafka-addr internal://0.0.0.0:9092,external://0.0.0.0:19092
97     - --advertise-kafka-addr internal://redpanda-0:9092,external://localhost:19092
98     - --pandaproxy-addr internal://0.0.0.0:8082,external://0.0.0.0:18082
99     - --advertise-pandaproxy-addr internal://redpanda-0:8082,external://localhost:18082
100    - --schema-registry-addr internal://0.0.0.0:8081,external://0.0.0.0:18081
101    - --rpc-addr redpanda-0:33145
102    - --advertise-rpc-addr redpanda-0:33145
103    - --smp 1
104    - --memory 16
105    - --mode dev-container
106    - --default-log-level=info
107   image: docker.redpanda.com/redpandadata/redpanda:v23.1.11
108   container_name: redpanda-0
109   volumes:
110     - redpanda-0:/var/lib/redpanda/data
111   ports:
112     - 18081:18081
113     - 18082:18082
114     - 19092:19092
115     - 19644:19644
116   healthcheck:
117     test: ["CMD-SHELL", "rpk cluster health | grep -E 'Healthy:.*true' || exit 1"]
118     interval: 15s
119     timeout: 3s
120     retries: 5
121     start_period: 5s
122
123 konduktor-gateway:
124   image: konduktor/konduktor-gateway:2.3.0
125   hostname: konduktor-gateway
126   container_name: konduktor-gateway
127   environment:
128     KAFKA_BOOTSTRAP_SERVERS: redpanda-0:9092
129   ports:
130     - "8888:8888"
131   healthcheck:
132     test: curl localhost:8888/health
133     interval: 5s
134     retries: 25
135   depends_on:
136     redpanda-0:
137       condition: service_healthy
138
139 n8n:
140   image: docker.n8n.io/n8nio/n8n
141   container_name: n8n
142   ports:
143     - "5678:5678"
144   volumes:
145     - n8n_data:/home/node/.n8n
146
147 volumes:
148   pg_data: {}
149   konduktor_data: {}

```

Rys. 5.2. Fragment pliku docker-compose.yml

Rysunek 5.3 przedstawia okno desktopowej wersji Docker, aby ukazać cały projekt wdrożony w kontenerach na lokalnym hoście.

<input type="checkbox"/>	<div> kafka</div>		Running (9/9)	6.86%	1 hour ago	<div></div> <div></div> <div></div>	
<input type="checkbox"/>	<div> conduktor-gateway</div> <div>d5c59ce8b549 </div>	conduktor/conduktor-gateway:2.3.0	Running	0.46%	8888.8888 	1 hour ago	<div></div> <div></div> <div></div>
<input type="checkbox"/>	<div> lremont-app</div> <div>5fb265e7f4ae </div>	mnykolaichuk/oraca-dyplomowa-app:2024-CSP-V5	Running	0.08%	8083.8083 	1 hour ago	<div></div> <div></div> <div></div>
<input type="checkbox"/>	<div> my-php-mysql</div> <div>f4a898831456 </div>	phomvadmin/phomvadmin:latest	Running	0%	8082.80 	1 hour ago	<div></div> <div></div> <div></div>
<input type="checkbox"/>	<div> conduktor-platform-1</div> <div>6c5596d40d10 </div>	conduktor/conduktor-platform:1.20.0	Running	2.43%	8080.8080 	1 hour ago	<div></div> <div></div> <div></div>
<input type="checkbox"/>	<div> redpanda-0</div> <div>b2bb11090cd8 </div>	docker.redpanda.com/redpandadata/redpanda:v23.1.11	Running	2.4%	18081.18081  Show all ports (4)	1 hour ago	<div></div> <div></div> <div></div>
<input type="checkbox"/>	<div> postgresql-1</div> <div>84eb93ffe361 </div>	postgres:14	Running	0.01%		1 hour ago	<div></div> <div></div> <div></div>
<input type="checkbox"/>	<div> n8n</div> <div>974deca2ec4c </div>	docker.n8n.io/n8nio/n8n	Running	0.02%	5678.5678 	1 hour ago	<div></div> <div></div> <div></div>
<input type="checkbox"/>	<div> db</div> <div>4594b624c219 </div>	mnykolaichuk/oraca-dyplomowa-mysql-db:2023	Running	0.39%	3306.3306 	1 hour ago	<div></div> <div></div> <div></div>
<input type="checkbox"/>	<div> conduktor-monitoring-1</div> <div>b6778e6d322a </div>	conduktor/conduktor-platform-cortex:1.20.0	Running	1.07%		1 hour ago	<div></div> <div></div> <div></div>

Rys. 5.3. Docker Desktop

6. Wnioski

W ramach projektu udało się uzyskać skalowalną i odporną na awarie architekturę, dzięki zastosowaniu konteneryzacji oraz brokera zdarzeń Apache Kafka. Ten ostatni zapewnia natychmiastową wbudowaną skalowalność w przypadku wzrostu ilości zdarzeń. Narzędzie low-code n8n umożliwia dość intuicyjną obsługę automatyzacji wysyłania wiadomości, jednak nie jest ono pozbawione pewnych wad. Przy pracach nad projektem autorzy napotkali problemy z wersjami dokumentacji n8n, pomimo powszechności danych, wyszukanie aktualnych informacji stanowiło wyzwanie. Niemniej jednak postawione cele zostały osiągnięte, a wynikiem jest elastyczna architektura, idealnie przygotowana do przyszłego rozwoju.