6.2. API

6.2.1. Cel i zadania API

Celem API jest zapewnienie komunikacji pomiędzy użytkownikami a pojemnikami transportowymi oraz zapewnienie logiki całego systemu. Obsługuje ona zapytania wysyłane od użytkowników i na ich podstawie zwraca informacje zapisane w bazie danych na temat dostępnych zleceń, wysłanych przez użytkownika paczek i odbieranych przez użytkownika paczek, ich stanu itd.

6.2.2. Narzędzia

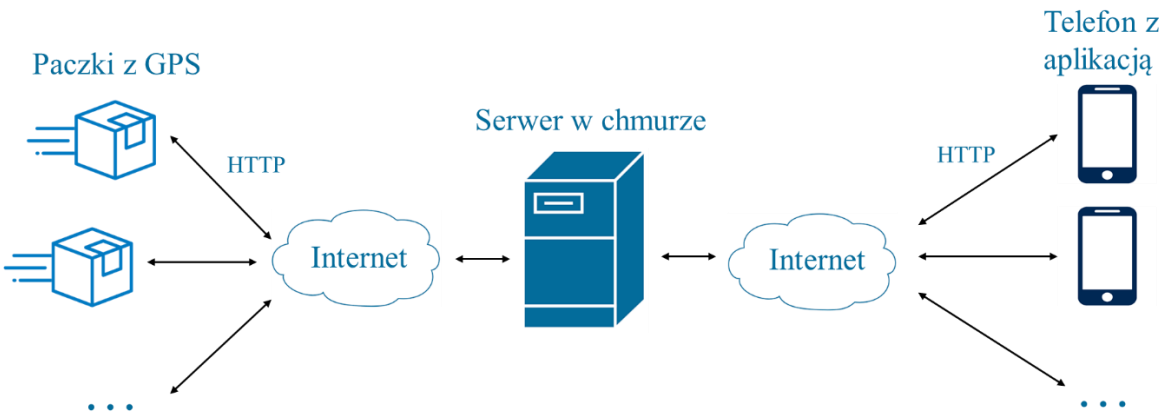
1) Spring Boot REST API
   - Framework do tworzenia aplikacji
   - REST
   - Język programowania: Java
2) Baza danych MongoDB
   - Rozwiązanie chmurowe
3) Postman
   - Narzędzie do testowania endpointów

6.2.3. Diagram klas

Diagram klas został załączony jako osobny plik oraz na końcu tego sprawozdania z realizacji projektu (Dodatek 1).

6.2.4. Komunikacja

API obsługuje całość logiki systemu, aplikacja jedynie wysyła zapytania, pojemnik transportowy wykonuje polecenia wydawane przez API. Całość komunikacji między tymi trzema komponentami odbywa się poprzez sieć Internet poprzez wysyłanie zapytań HTTP do API. Wszystkie ciała wysyłanych zapytań pisane były w formacie JSON.



Rysunek 3. Ogólny schemat komunikacji między komponentami systemu

6.2.4.1. Komunikacja API ↔ Aplikacja

Komunikacja pomiędzy aplikacją a API stanowi standardowe rozwiązanie standardu REST, tj. klient wysyła zapytanie na określony endpoint, serwer następnie wysyła klientowi dane.

Poniżej znajduje się lista endpointów zwracające informacje na temat przesyłek:

1) ***/users/{login}/senders/parcels*** – zwraca informacje o wszystkich przesyłkach użytkownika *login*, które są przez niego nadawane (rola nadawca),

2) ***/users/{login}/receivers/parcels*** – zwraca informacje o wszystkich przesyłkach użytkownika *login*, które są przez niego odbierane (rola odbiorca),
3) ***/users/{login}/senders/parcels*** – zwraca informacje o wszystkich przesyłkach użytkownika *login*, które są przez niego przewożone (rola kurier),
4) ***/parcels*** – zwraca informacje o wszystkich przesyłkach zarejestrowanych w systemie, przewidziane dla administratorów systemu,
5) ***/parcels/{id}*** – zwraca szczegółowe informacje na temat paczki o identyfikatorze *id*,

Przykładowa odpowiedź zawierająca szczegółowe informacje o paczce *6484a00a0dbd87579db3ca74*:


```
{
    "parcel": {
        "id": "6484a00a0dbd87579db3ca74",
        "size": "A",
        "deliverFrom": {
            "region": {
                "country": "Poland",
                "province": "LowerSilesia",
                "city": "Wroclaw",
                "postalCode": "50-337"
            },
            "street": "Uliczna",
            "nrOfHouse": "22B"
        },
        "deliverTo": {
            "region": {
                "country": "Poland",
                "province": "LowerSilesia",
                "city": "Olawa",
                "postalCode": "55-200"
            },
            "street": "Gornickiego",
            "nrOfHouse": "22"
        },
        "insurance": 1000.0,
        "sender": {
            "login": "admin2",
            "email": "admin@gmail.com",
            "name": "Wojciech",
            "surname": "Niedomagala",
```

```
            "password": "admin1",
            "phNumber": "123456789"
        },
        "receiver": {
            "login": "admin2",
            "email": "admin@gmail.com",
            "name": "Wojciech",
            "surname": "Niedomagala",
            "password": "admin1",
            "phNumber": "123456789"
        },
        "courier": null,
        "box": null
    },
    "status": [
        {
            "time": "2023-06-10T16:08:42.907",
            "status": {
                "status": "Czeka na przyjęcie zgłoszenia",
                "code": 1
            }
        }
    ],
    "a2a_code3_idle_open_courier_agree": false,
    "a2a_code3_protect_close_sender_agree": false,
    "a2a_code5_idle_open_receiver_agree": false,
    "a2a_code5_idle_open_courier_agree": false,
    "a2a_code5_end_open_receiver_agree": false
}
```


6) ***/regions/{countryName}/{provinceName}/{cityName}/parcels*** – zwraca paczki z regionu o kodzie pocztowym *provinceName*,

API do obsługi logiki tworzenia paczek oraz od operacji na paczkach korzysta z następnych grup endpointów. Endpointy oznaczone jako ***/parcels/a2a*** oznaczają przesyłki wysyłane bezpośrednio z adresu nadawcy na adres odbiorcy, z powodu braku czasu nie udało się zaprojektować logiki innych rodzajów przesyłek.

Akcje wykonywane na przesyłkach określane są przez ostatni człon adresu (np. ***box/assign***, ***/deliver*** itd.). To w jakim stanie musi znajdować się przesyłka określa człon wcześniej (np. ***/1***). To, kto może wykonać daną akację, określa człon (***/courier***, ***/receiver***, ***/sender***).

Statusy przesyłek typu *a2a* są z góry określone (słownikowanie w bazie danych), są to:

1) 1 – „Czeka na przyjęcie zgłoszenia”,
2) 3 – „Czeka na kuriera”,
3) 5 – „W drodze do odbiorcy”,
4) 7 – „Odebrana”.

Poniżej znajduje się lista endpointów związanych z przesyłkami typu adres do adresu, zapytania powinny być wykonywane przez użytkowników takiej samej kolejności jak poniżej. API w razie złamania tej reguły wyświetla stosowne komunikaty, np.: „***"message": "This method is not allowed while parcel has status: W drodzę do odbiorcy"***”. Wszystkie endpointy zdefiniowane poniżej są metodami *PUT*, za wyjątkiem ***/parcels/a2a***, które jest metodą *POST*.

Lista endpointów ***/parcels/a2a***:

1) ***/parcels/a2a*** – tworzenie nowego zlecenia, nadawca musi podać podstawowe dane na temat nowej przesyłki, następnie przesyłka czeka na przyjęcie zlecenia.

Przykładowe dane wysyłane do API:

```
{
    "size": "A",
    "deliverFrom":  {
        "region":{
            "postalCode":"55-200"
        },
        "street":"Uliczna",
        "nrOfHouse":"22B"
    },
    "deliverTo": {
        "region":{
            "postalCode":"50-337"
        },
        "street":"Gornickiego",
        "nrOfHouse":"22"
    },
    "insurance": 1000,
    "sender": {"login": "admin2"},
    "receiver": {"login": "admin2"}
}
```

Przykładowe dane zwracane przez API:

```
{
    "parcel": {
        "id": "6484a00a0dbd87579db3ca74",
```

```
        "size": "A",
        "deliverFrom": {
            "region": {
                "country": "Poland",
                "province": "LowerSilesia",
                "city": "Wroclaw",
                "postalCode": "50-337"
            },
            "street": "Uliczna",
            "nrOfHouse": "22B"
        },
        "deliverTo": {
            "region": {
                "country": "Poland",
                "province": "LowerSilesia",
                "city": "Olawa",
                "postalCode": "55-200"
            },
            "street": "Gornickiego",
            "nrOfHouse": "22"
        },
        "insurance": 1000.0,
        "sender": {
            "login": "admin2",
            "email": "admin@gmail.com",
            "name": "Wojciech",
            "surname": "Niedomagala",
            "password": "admin1",
            "phNumber": "123456789"
        },
        "receiver": {
            "login": "admin2",
            "email": "admin@gmail.com",
            "name": "Wojciech",
            "surname": "Niedomagala",
            "password": "admin1",
            "phNumber": "123456789"
```

```
        },
        "courier": null,
        "box": null
    },
    "status": [
        {
            "time": "2023-06-10T16:08:42.907",
            "status": {
                "id": {
                    "timestamp": 1685125170,
                    "date": "2023-05-26T18:19:30.000+00:00"
                },
                "status": "Czeka na przyjęcie zgłoszenia",
                "code": 1
            }
        }
    ],
    "a2a_code3_idle_open_courier_agree": false,
    "a2a_code3_protect_close_sender_agree": false,
    "a2a_code5_idle_open_receiver_agree": false,
    "a2a_code5_idle_open_courier_agree": false,
    "a2a_code5_end_open_receiver_agree": false
}
```


2) ***/parcels/a2a/{id}/courier/1/accept*** – zaakceptowanie zlecenia przez kuriera, kurier musi wysłać swój login, następnie serwer odsyła zaktualizowane, pełne dane paczki *id*, zawierające dane kuriera oraz nowy kod statusu. Status przesyłki zmienia się.

Przykładowe dane wysyłane na serwer:


```
{
    "login": "admin2"
}
```


Zaktualizowane informacje o paczce:


```
{
    "parcel": {
        "id": "6484a00a0dbd87579db3ca74",
        "size": "A",
        "deliverFrom": {
            "region": {
```

```
            "country": "Poland",
            "province": "LowerSilesia",
            "city": "Wroclaw",
            "postalCode": "50-337"
        },
        "street": "Uliczna",
        "nrOfHouse": "22B"
    },
    "deliverTo": {
        "region": {
            "country": "Poland",
            "province": "LowerSilesia",
            "city": "Olawa",
            "postalCode": "55-200"
        },
        "street": "Gornickiego",
        "nrOfHouse": "22"
    },
    "insurance": 1000.0,
    "sender": {
        "login": "admin2",
        "email": "admin@gmail.com",
        "name": "Wojciech",
        "surname": "Niedomagala",
        "password": "admin1",
        "phNumber": "123456789"
    },
    "receiver": {
        "login": "admin2",
        "email": "admin@gmail.com",
        "name": "Wojciech",
        "surname": "Niedomagala",
        "password": "admin1",
        "phNumber": "123456789"
    },
    "courier": {
        "login": "admin2",
```

```
            "email": "admin@gmail.com",
            "name": "Wojciech",
            "surname": "Niedomagala",
            "password": "admin1",
            "phNumber": "123456789"
        },
        "box": null
    },
    "status": [
        {
            "time": "2023-06-10T16:08:42.907",
            "status": {
                "status": "Czeka na przyjęcie zgłoszenia",
                "code": 1
            }
        },
        {
            "time": "2023-06-10T17:24:29.693",
            "status": {
                "status": "Czeka na kuriera",
                "code": 3
            }
        }
    ],
    "a2a_code3_idle_open_courier_agree": false,
    "a2a_code3_protect_close_sender_agree": false,
    "a2a_code5_idle_open_receiver_agree": false,
    "a2a_code5_idle_open_courier_agree": false,
    "a2a_code5_end_open_receiver_agree": false
}
```


3) ***/parcels/a2a/{id}/courier/3/box/assign*** – kurier przypisuje pojemnik transportowy, określony w ciele zapytania, do przesyłki *id*.

Przykładowe dane wysyłane na serwer:


```
{
    "mac": "176:178:28:11:21:204"
```

```
}
```

Zaktualizowane informacje o paczce:

```
{
    "id": "6484a00a0dbd87579db3ca74",
    "size": "A",
    "deliverFrom": {
        "region": {
            "country": "Poland",
            "province": "LowerSilesia",
            "city": "Wroclaw",
            "postalCode": "50-337"
        },
        "street": "Uliczna",
        "nrOfHouse": "22B"
    },
    "deliverTo": {
        "region": {
            "country": "Poland",
            "province": "LowerSilesia",
            "city": "Olawa",
            "postalCode": "55-200"
        },
        "street": "Gornickiego",
        "nrOfHouse": "22"
    },
    "insurance": 1000.0,
    "sender": {
        "login": "admin2",
        "email": "admin@gmail.com",
        "name": "Wojciech",
        "surname": "Niedomagala",
        "password": "admin1",
        "phNumber": "123456789"
    },
    "receiver": {
        "login": "admin2",
        "email": "admin@gmail.com",
```

```
        "name": "Wojciech",
        "surname": "Niedomagala",
        "password": "admin1",
        "phNumber": "123456789"
    },
    "courier": {
        "login": "admin2",
        "email": "admin@gmail.com",
        "name": "Wojciech",
        "surname": "Niedomagala",
        "password": "admin1",
        "phNumber": "123456789"
    },
    "box": {
        "mac": "176:178:28:11:21:204"
    }
}
```

4) ***/parcels/a2a/{id}/courier/3/box/open*** – kurier wydaje pojemnikowi transportowemu polecenie otworzenia pokrywy w celu udostępnia pojemnika .

Odpowiedź od serwera:

```
{
   "message": "Message to the box is being sent"
}
```

5) ***/parcels/a2a/{id}/sender/3/box/protect*** – nadawca wydaje pojemnikowi transportowemu polecenie zabezpieczenia pokrywy. Pojemnik po wykonaniu polecenia jest gotowy do transportu.

Odpowiedź od serwera:

```
{
   "message": "Message to the box is being sent"
}
```

6) ***/parcels/a2a/{id}/courier/5/box/open*** – kurier wydaje pojemnikowi transportowemu polecenie otworzenia pokrywy po dotarciu do odbiorcy. Polecenie może zostać wydane po ***/parcels/a2a/{id}/receiver/5/box/open***.

Odpowiedź od serwera:

```
{
   "message": "Waiting for receiver to agree"
```

```
}
```

7) ***/parcels/a2a/{id}/receiver/5/box/open*** – odbiorca wydaje pojemnikowi transportowemu polecenie otworzenia pokrywy po dotarciu kuriera. Polecenie może zostać wydane przed ***/parcels/a2a/{id}/courier/5/box/open***.

Odpowiedź od serwera:

```
{
  "message": "Message to the box is being sent"
}
```

8) ***/parcels/a2a/{id}/receiver/5/end*** – odbiorca wydaje pojemnikowi transportowemu polecenie zabezpieczenia pokrywy po wyjęciu transportowanych dóbr.

Odpowiedź od serwera:

```
{
  "message": "Message to the box is being sent"
}
```

6.2.4.2. Komunikacja API ↔ Pojemnik transportowy

Komunikacja z pojemnikiem transportowym odbywa się za pomocą kilku endpointów. To, jaki endpoint jest używany zależy od stanu w jakim znajduje się pojemnik. Są to:

1) ***Open*** – jeżeli wartość jest równa „*true*”, wówczas pojemnik powinien mieć pokrywę odblokowaną i nie powinien wysyłać alarmów; jeżeli jest równa „*false*”, wówczas pojemnik powinien być zamknięty i powinien wysyłać alarmy.
2) ***Protect*** – jeżeli wartość jest równa „*true*”, wówczas pojemnik powinien wysyłać dane telemetryczne; w przeciwnym razie powinien tego nie robić.

W zależności od ***Protect*** pojemnik wysyła zapytania na:

1) ***/boxes/{mac}/telemetries*** – gdy wartość ***Protect*** jest równa „*true*”, jest to metoda *POST*, pojemnik wysyła dane telemetryczne. Wartość „***mac***” oznacza zapisany fabrycznie adres MAC w pamięci pojemnika, który jest zarazem jego numerem identyfikacyjnym. Parametr „*maxAcceleration*” aktualizowany jest tylko po przekroczeniu zadanej wartości przyspieszenia.

Przykładowe dane wysyłane na serwer:

```
{
    "latitude":    50.9858741760254,
    "longitude":   17.234375,
    "maxAcceleration":     1.27978515625,
    "temperature": 25.891889572143555,
    "humidity":    21.011672973632812,
    "time": "15:44:40",
    "batteryStatus":       100
```

```
}
```

2) ***/boxes/{mac}/idle*** – gdy wartość ***Protect*** jest równa „*false*”, metoda *GET*, ma zadanie tylko określić pojemnikowi właściwy stan, w jakim powinien się znajdować.

Tak jak wspomniano wcześniej, pojemnik musi pobrać informację z serwera na temat stanu, w którym powinien się znajdować. Aby nie tworzyć do tego dedykowanego endpointu, każdy endpoint, na który zapytanie wyśle paczka, zwraca właściwy stan. Na przykład pojemnik wysyła dane telemetryczne na endpoint ***/boxes/{mac}/telemetries***, wówczas serwer, jeżeli wysłane dane są poprawne, odeśle właściwy stan w formacie JSON:

```
{
   "protect": false,
   "open": false,
   "ack": false
}
```

Takie podejście ma za zadanie zapewnienie energooszczędności – pojemnik może rzadziej korzystać z modułów komunikacyjnych (SIM800l), które pobierają znaczną część energii.

Kiedy pojemnik odebrał wiadomość i jest ona taka sama jak ta w jego pamięci, wówczas żadne inne akcje nie muszą być podejmowane. W momencie, gdy pojemnik pobrał nową wiadomość i różni się ona od tej, którą ma zapisaną w pamięci, wówczas jest zobligowany do odesłania odpowiedzi potwierdzającej zamianę stanu na endpoint ***/boxes/{mac}/answer***, dla wiadomości powyżej będzie to:

```
{
   "protect": false,
   "open": false,
   "ack": true
}
```

Pole „*ack*” zostało zmienione na wartość „*true*” – tylko taka sama wiadomość jak ta którą nadesłał serwer, tylko ze zmienioną wartością „*ack*” na „*true*”, oznacza potwierdzenie zmiany stanu. Serwer odpowie wówczas taką samą wiadomością.

Pojemnik nie może wysłać wiadomości potwierdzającej zmianę stanu, jeżeli ona faktycznie nie nastąpiła – jeśli pokrywa pojemnika nie została jeszcze zamknięta przez użytkownika, pomimo że wymaga tego serwer, wówczas odpowiedź nie zostanie wysłana. Pojemnik odpowie dopiero po wykonaniu zadania.

Stan pojemnika pełni ważną funkcję w trakcie procesu przekazywania przesyłki – np. jeżeli kurier zlecił otwarcie paczki, ale nie zdążyła ona jeszcze wykonać zadania, wówczas nadawca nie będzie w stanie jej zabezpieczyć – jest to jasny sygnał dla systemu, że przesyłany przedmiot nie został włożony do pojemnika. Ma to za zadanie chronić użytkowników przed przypadkowym wykonaniem akcji, które nie powinny się zdarzyć. Również zakończenie procesu przekazywania paczki jest „zatwierdzane” przez przesyłkę poprzez wysłanie odpowiedniego stanu ("*protect*": *false*, "*open*": *false*).

Ostatni endpoint dostępny dla przesyłki jest to ***/boxes/{mac}/alarms***, który służy do nadsyłania alarmów. Alarmy są z góry zdefiniowane:

1) 0 – zamknięcie pojemnika po tym jak został on otwarty w niewłaściwym momencie,

2) 1 – otwarcie pojemnika w niewłaściwym momencie (zmienna „***Open***” w stanie „*false*”),
3) 2 – przekroczenie zadanego przyspieszenia – nieostrożny transport.

6.2.4.3. Deployment na chmurze AWS

Testy API wykonywane były na chmurze AWS, aby umożliwić w prosty sposób dostęp do serwera pojemnikowi. Serwer otrzymał stały adres IP ***3.215.18.200*** (usługa *Ellastic IPs* w *AWS EC2*). Ponieważ używane było do tego konto studenckie, serwer wyłączał się po czterech godzinach Aby zapewnić szybszy rozruch serwera zdefiniowano domyślny szablon instancji, parametry:

1) System: Amazon Linux 2
2) Typ instancji: t3.small (zapewnia 2 GB RAM, które stanowiły minimum do szybkiego działania API)
3) Kod startowy:

```
#!/bin/bash

yum update -y


wget --no-check-certificate -c --header "Cookie: oraclelicense=accept-securebackup-cookie" https://download.oracle.com/java/17/latest/jdk-17_linux-x64_bin.rpm

rpm -Uvh jdk-17_linux-x64_bin.rpm


yum install unzip


wget https://github.com/dominicus28/BTWL/archive/refs/heads/backend.zip

unzip backend.zip


#startup config

cat >>  /etc/rc.local << EOF


#port redirect

sudo iptables -I INPUT 1 -p tcp --dport 8080 -j ACCEPT

sudo iptables -I INPUT 1 -p tcp --dport 80 -j ACCEPT

sudo iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 8080


#start btwl service

sudo chmod 777 /BTWL-backend/gradlew

sudo chmod 777 /BTWL-backend/src/main/resources/application.properties

cd /BTWL-backend
```

```
sudo ./gradlew bootRun


EOF


#port redirect
iptables -I INPUT 1 -p tcp --dport 8080 -j ACCEPT
iptables -I INPUT 1 -p tcp --dport 80 -j ACCEPT
iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 8080


#set auth
sudo chmod 777 /BTWL-backend/src/main/resources/application.properties
cat > /BTWL-backend/src/main/resources/application.properties << EOF
spring.data.mongodb.uri=mongodb+srv://administrator:YVr8KsQ6yeugYMVh@btwl.0fntafi.mongodb.net/?retryWrites=true&w=majority
spring.data.mongodb.database=btwldb


EOF


#start btwl service
chmod 777 /BTWL-backend/gradlew
cd /BTWL-backend
./gradlew bootRun
```

Po włączeniu serwera niezbędne było przypisanie do niego zarezerwowanego IP (***3.215.18.200***).

6.2.4.4. Realizacja

API zostało zaprojektowane i napisane przez Nataliię Belinską oraz Adama Gołeckiego. API miało oprócz funkcji, które zostały zaimplementowane, zapewniać:

1) uwierzytelnianie,
2) autoryzację,
3) więcej typów przesyłek (paczkomat do paczkomatu itd.).

Niestety z powodu małego zespołu i krótkiego czasu trwania projektu, te cele nie zostały zrealizowane.