

## 1. Docker

W celu uruchomienia kontenerów należy użyć komendy:

- `docker-compose build`

a następnie

- `docker-compose up`

Automatycznie uruchomione zostaną serwisy springboot. Stworzona zostanie baza danych, która wywoła skrypt generujący schematy *familyappdb* oraz *familymemberdb*. Aby móc użyć powyższe komendy bez uprawnień root, należy wykonać poniższe kroki:

- Wywołać komendę `sudo groupadd docker`,
- Wywołać komendę `sudo gpasswd -a $USER docker`,
- Uruchomić ponownie maszynę na której pracujemy

W celu sprawdzenia, czy docker uruchamia się bez uprawnień root'a można uruchomić komendę `docker run hello-world`

## 2. Żądania

### 2.1 Create

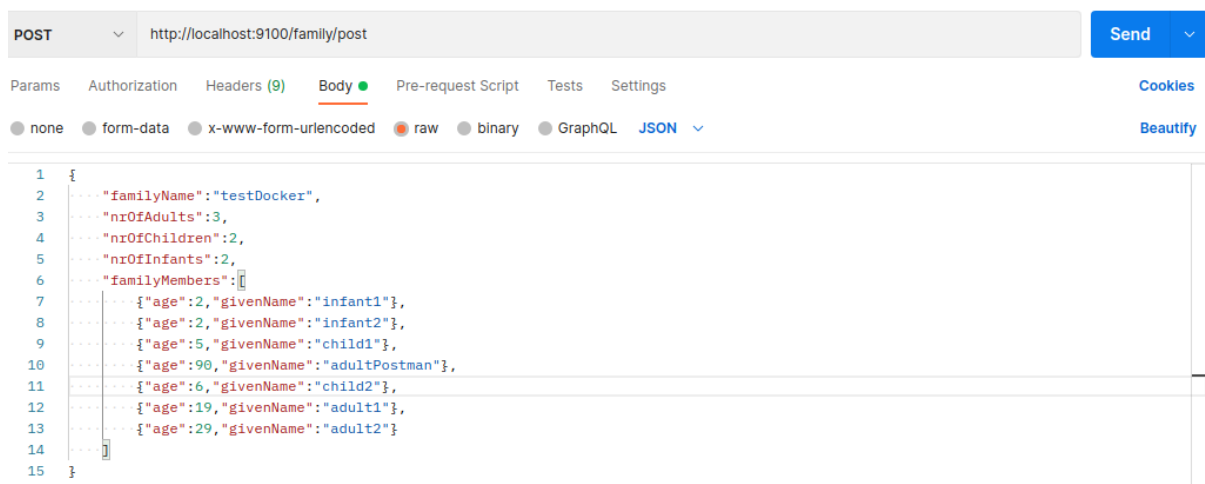
Podajemy url (`http://localhost:9100/family/post`), w nagłówkach dodajemy „Content-type” z value „application/json”.

The screenshot shows a REST client interface with the following details:

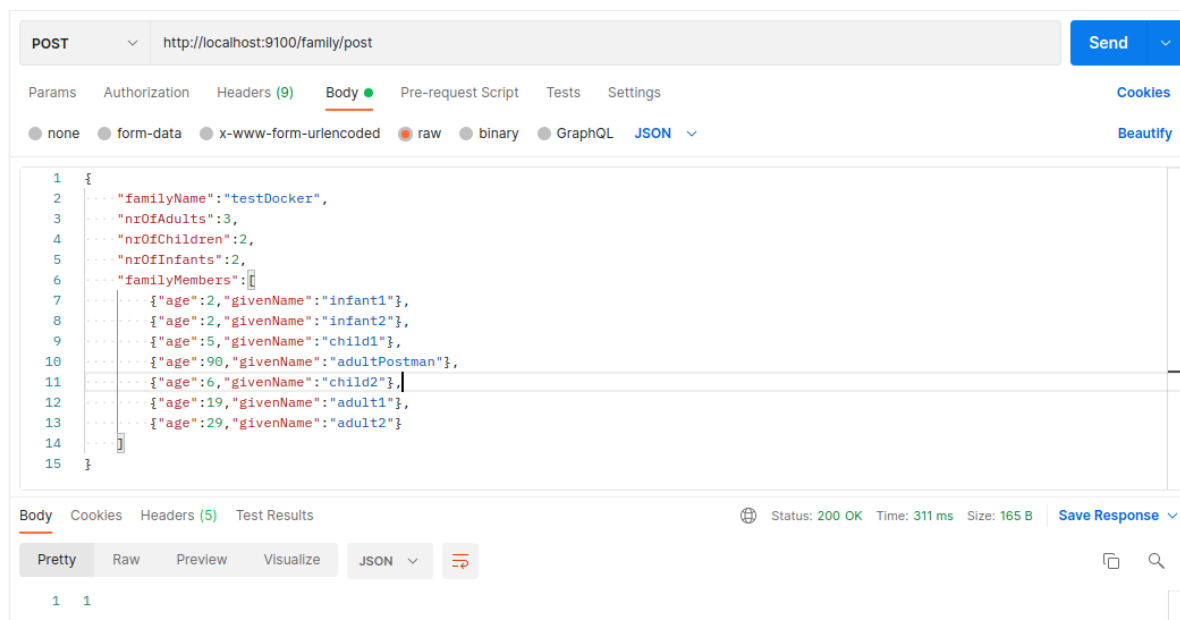
- Method:** POST
- URL:** `http://localhost:9100/family/post`
- Buttons:** Params, Authorization, Headers (9), Body, Pre-request Script, Tests, Settings, Cookies, Send
- Headers Table:**

KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/> Content-Type	application/json				

W body zaznaczamy „raw”, następnie wybieramy typ danych „JSON” i podajemy testowe dane (**dane testowe** zostały umieszczone na końcu pliku)

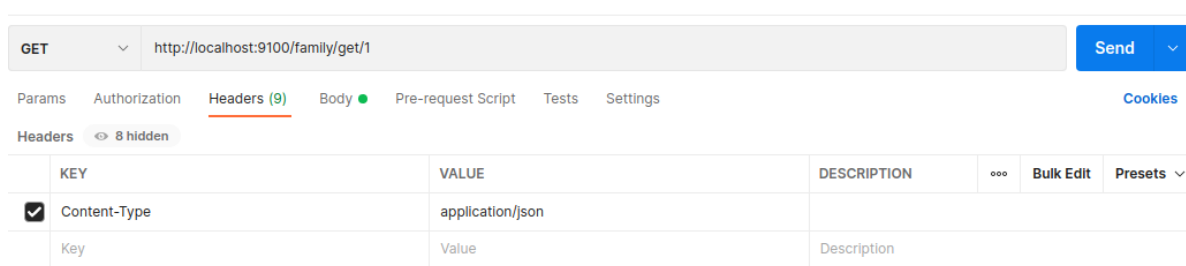


Po kliknięciu przycisku „SEND” aplikacja powinna zwrócić nam ID dodanej rodziny.



## 2.2 Get

Aby otrzymać informacje o danej rodzinie należy zmienić typ żądania na „GET” oraz podać url: <http://localhost:9100/family/get/{id}>.



W odpowiedzi otrzymujemy dane rodziny i listę członków tej rodziny.

GET http://localhost:9100/family/get/1

Status: 200 OK Time: 82 ms Size: 814 B Save Response

```

1 {
2   "id": 1,
3   "familyName": "testDocker",
4   "nrOfAdults": 3,
5   "nrOfChildren": 2,
6   "nrOfInfants": 2,
7   "familyMembers": [
8     {
9       "id": 2,
10      "familyId": 1,
11      "age": 2,
12      "familyName": "testDocker",
13      "givenName": "infant1"
14    },
15    {
16      "id": 3,
17      "familyId": 1,
18      "age": 2,
19      "familyName": "testDocker",
20      "givenName": "infant2"
21    },
22    {
23      "id": 4,
24      "familyId": 1,
25      "age": 5,
26      "familyName": "testDocker",
27      "givenName": "child1"
28    }
29  ],
30  "id": 5.

```

### 3. Flyway

Konfiguracja pliku flyway.conf:

- *flyway.url=jdbc:postgresql://localhost:5432/IntecaDB*
- *flyway.user=postgres*
- *flyway.password=<<password>>*
- *flyway.baselineOnMigrate=true*

Zawartość pliku znajdującego się w folderze *sql* (*V1\_Create\_Schemas.sql*):

```

1 CREATE SCHEMA IF NOT EXISTS FamilyDB;
2 CREATE SCHEMA IF NOT EXISTS FamilyMemberDB;

```

Po wywołaniu komendy *flyway migrate* w bazie danych utworzone zostały powyższe schematy.

**Dane testowe:**

```
{  
  "familyName":"testDocker",  
  "nrOfAdults":3,  
  "nrOfChildren":2,  
  "nrOfInfants":2,  
  "familyMembers":[  
    {"age":2,"givenName":"infant1"},  
    {"age":2,"givenName":"infant2"},  
    {"age":5,"givenName":"child1"},  
    {"age":90,"givenName":"adultPostman"},  
    {"age":6,"givenName":"child2"},  
    {"age":19,"givenName":"adult1"},  
    {"age":29,"givenName":"adult2"}  
  ]  
}
```