



# Sprawozdanie

Przedmiot: Sieci Komputerowe 2  
**Projekt Komunikator**





**Autorzy:**

Aleksandra Kozak 151870  
Jakub Lemiesiewicz 151922

## 1. Opis i struktura projektu

Tematem naszego projektu był **Komunikator**. Jego prezentacja odbywa się za pomocą uruchomienia dwóch emulatorów systemu **iOS**. Aplikacja umożliwia komunikowanie się na żywo przez wielu użytkowników.

Projekt składa się łącznie z 3 mikroservisów oraz bazy danych:

Mikroservis	Technologia	
Serwer		Język: <b>C</b>
Klient		Język: <b>JS/TS</b> Framework <b>React Native</b>
Proxy		Język: <b>Python</b> , Framework: <b>Django Rest Framework</b>
Baza danych		Technologia: <b>SQLite</b>



Film z prezentacją projektu: <https://youtu.be/-NxhRRDjhd0>

## 2. Opis klienta

Klient jest napisany w **React Native**. Aplikacja komunikuje się z serwerem poprzez zapytania API. Otrzymane dane są zapisywane w **React Redux**.

Aplikacja obsługuje logowanie i rejestrację użytkownika. Logowanie jest zapamiętywane, więc użytkownik nie musi się logować na nowo po każdym wyłączeniu aplikacji. Po udanej autentykacji użytkownik przechodzi do głównej części aplikacji - komunikatora.

W komunikatorze użytkownik ma dostęp do listy swoich znajomych oraz listy innych użytkowników. Aby móc rozpocząć czat z inną osobą wiadomości należy dodać ją do znajomych. Czat na bieżąco się odświeża, więc możemy na żywo czatować z innymi użytkownikami.

## 3. Opis serwera

Serwer obsługuje zapytania **API** przychodzące od **klienta** przez **proxy**. Po zaakceptowaniu przychodzącego zapytania zostaje ono przeanalizowane i wysłane do funkcji, która odpowiada z routing. Na jego podstawie jest badane czy dane zapytanie jest typu **POST** czy **GET** i dalej na podstawie ścieżki zapytania **PATH** jest uruchamiany odpowiedni endpoint (endpointy opisane są w dokumentacji **API** (2.4). Serwer podczas uruchamiania łączy się z bazą danych i obsługując zapytania komunikuje się z bazą.

## 4. Opis Proxy

Proxy jest to dodatkowy serwer napisany w **Django (Django Rest Framework)**, odpowiada on dodatkowo za obsługę niektórych błędów oraz służy do przekazania Payload'u w przypadku metod **POST**, aby były w formie przyjaznej dla serwera.

## 5. Struktura bazy danych

Baza danych zawiera 3 tabele:

**Users:** id (*int*), email (*string*), name (*string*), surname (*string*), password (*string*)

**Friends:** main\_id (*int*), related\_id (*int*)

**Messages:** message\_id (*int*), author\_id (*int*), target\_id (*int*), message (*text*), created (*datetime*)

## 6. Dokumentacja API

Metoda	Endpoint	Payload
POST	/api/login/	email, password
POST	/api/register/	email, name, surname, password
GET	/api/list-my-fiends/	
GET	/api/list-others/	
POST	/api/add-friend/	user_id
POST	/api/chat/	user_id
POST	/api/message/	user_id, message
POST	/api/user-exists/	email