Programowanie Zespołowe 2016

Opis aplikacji

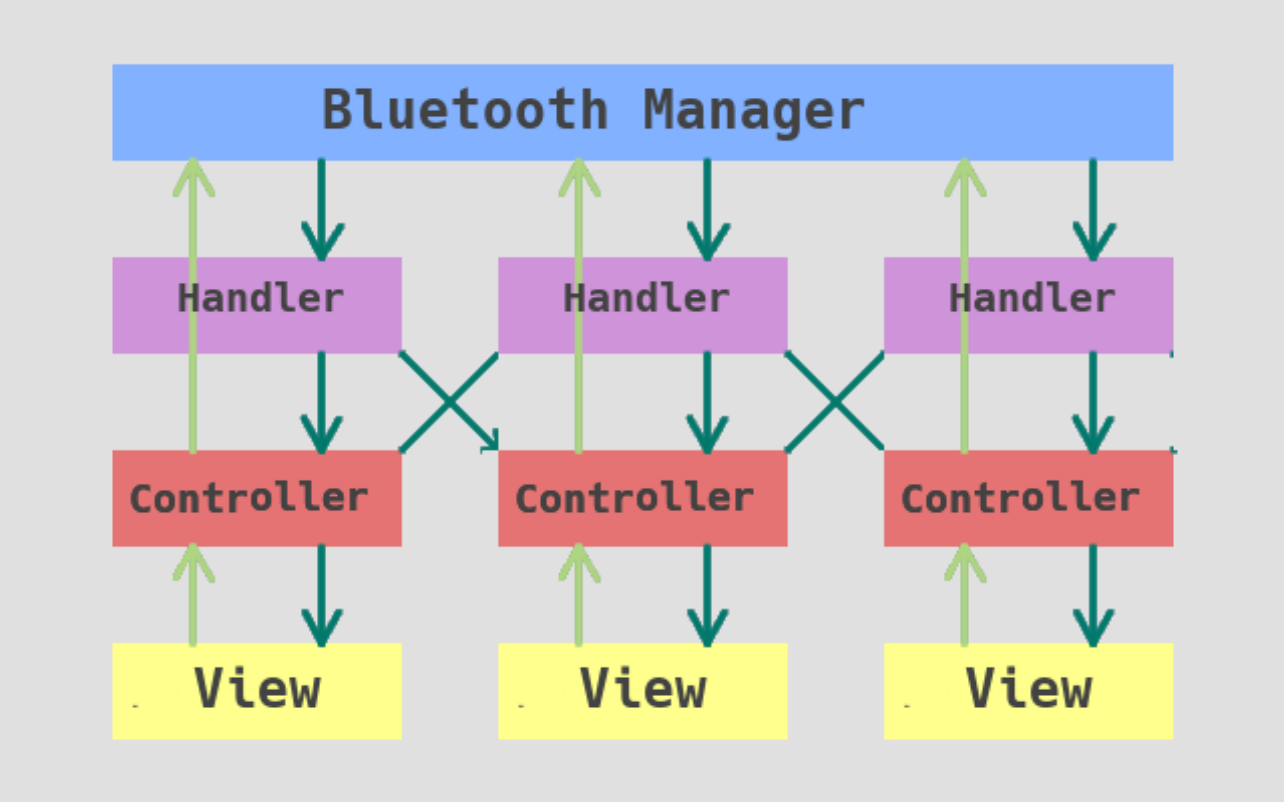
Aplikacja ma za zadanie połączenie dwóch lub więcej urządzeń, poprzez protokół komunikacyjny Bluetooth, celem zapewnienia podstawowej komunikacji w postaci wiadomości tekstowych.

Użyte technologie

Całość zostanie wykonana w technologii Xamarin w języku C#. Ma to znaczenie, gdyż większość zespołu już jest dobrze zaznajomiona z tym językiem. Xamarin pozwala nam pisać jeden kod i budować aplikacje zarówno dla iOS, Androida czy np. Windows Phone. Tym samym klient dostaje więcej za mniej.

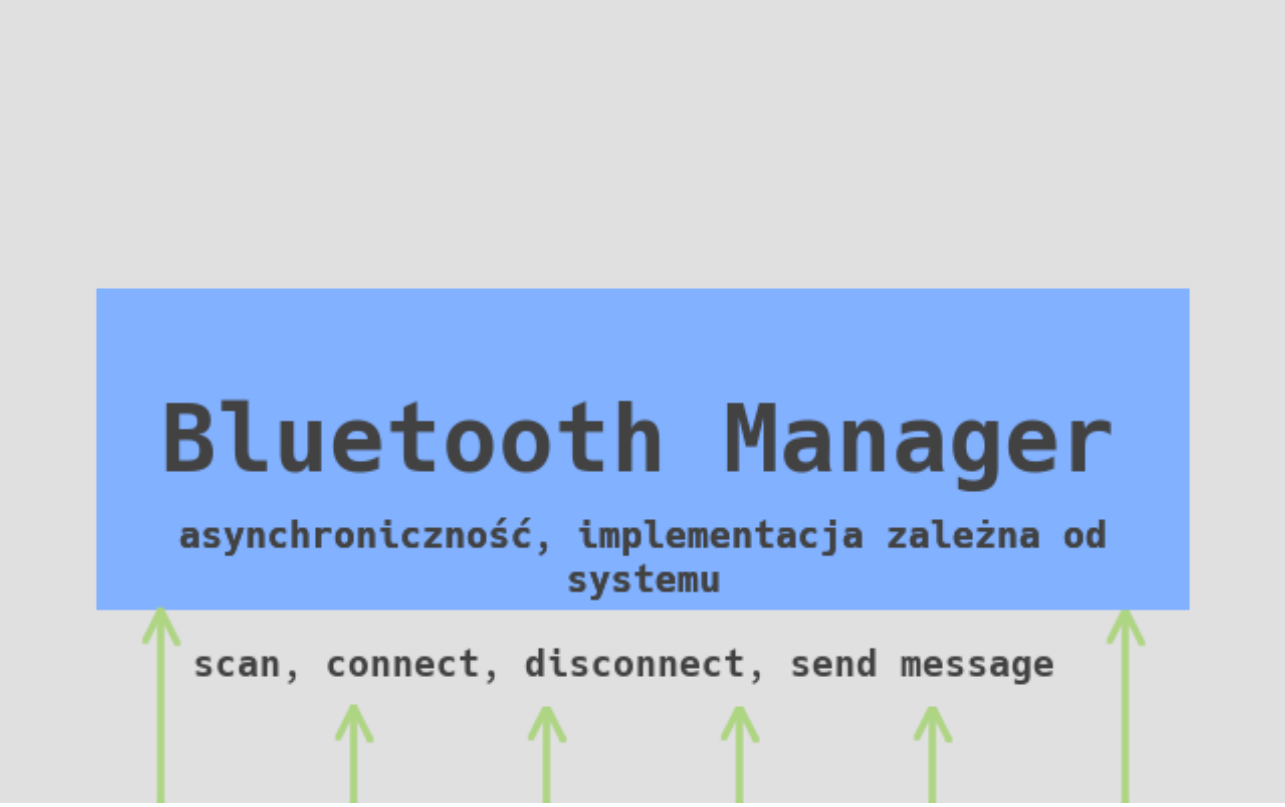
Sama firma Xamarin jest założona przez człowieka który zainicjował projekt Mono - warstwę aplikacji i bibliotek pozwalającą skompilować i uruchomić znaczną część frameworka .NET (od Microsoftu) na innych systemach takich jak macOS czy GNU/Linux.

Zarys architektury aplikacji



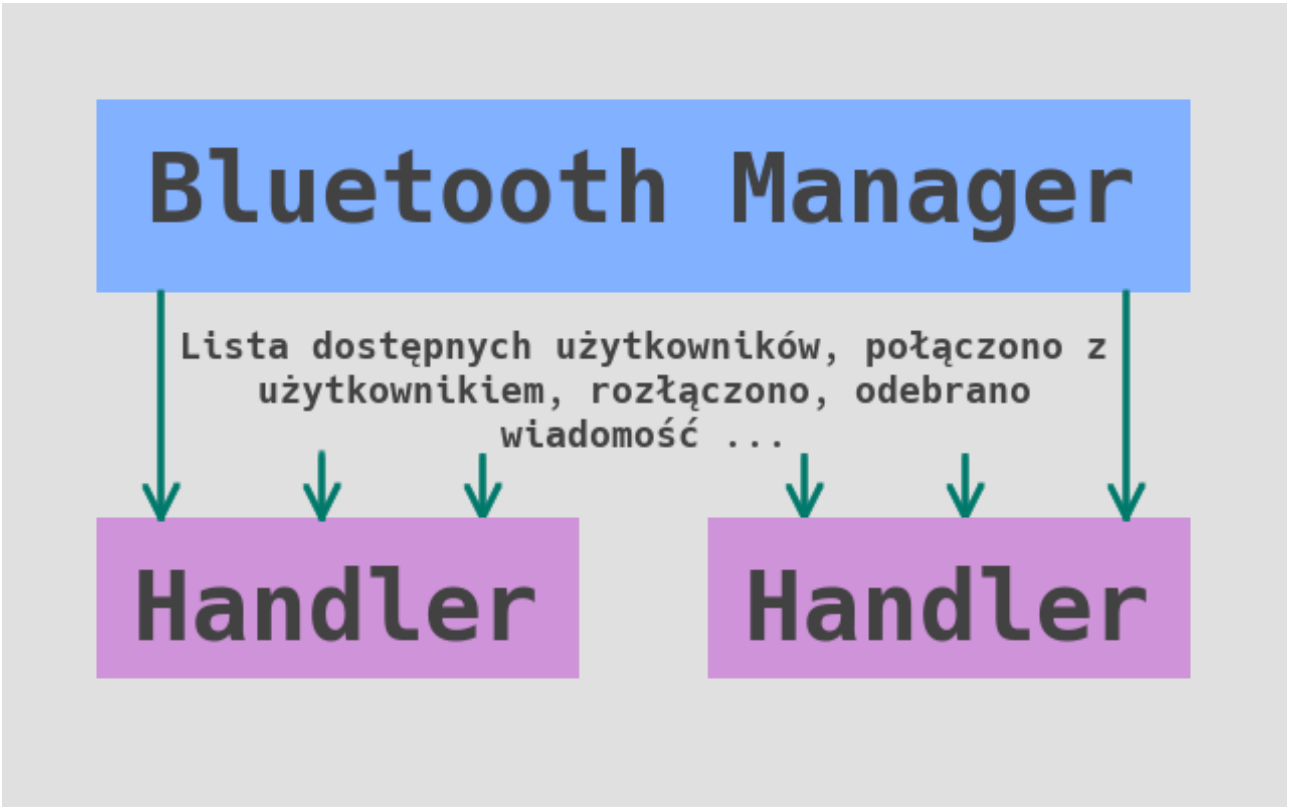
Aplikacja jest budowana w wzorcu projektowym Model - View - Controller.

W momencie gdy użytkownik naciśnie dowolny przycisk w oknie aplikacji, akcja ta zostanie przesłana do **kontrolera.** Następnie ten przekazuje rozkaz do **BluetoothManagera.**



BluetoothManager odpowiada za najniższą warstwę. Każde z urządzeń - odpowiednio iOS, jak i Android posiadają inne rozwiązania techniczne. Dlatego wymagane jest, aby najniższa warstwa, będąca „najbliżej sprzętu", była rozdzielona pomiędzy ich systemy. Wymusza to na nas zaimplementowanie dwóch modułów- do obsługi komunikacji **Bluetooth** na iOS oraz modułu do komunikacji na Androida.

Ponieważ zależy nam na tym, aby aplikacja w przyszłości była łatwo rozszerzalna, wprowadziliśmy warstwę abstrakcji - **IBluetoothManager,** która służy do uogólniania działań, jakie będą wykonywane w modułach dla danych systemów.



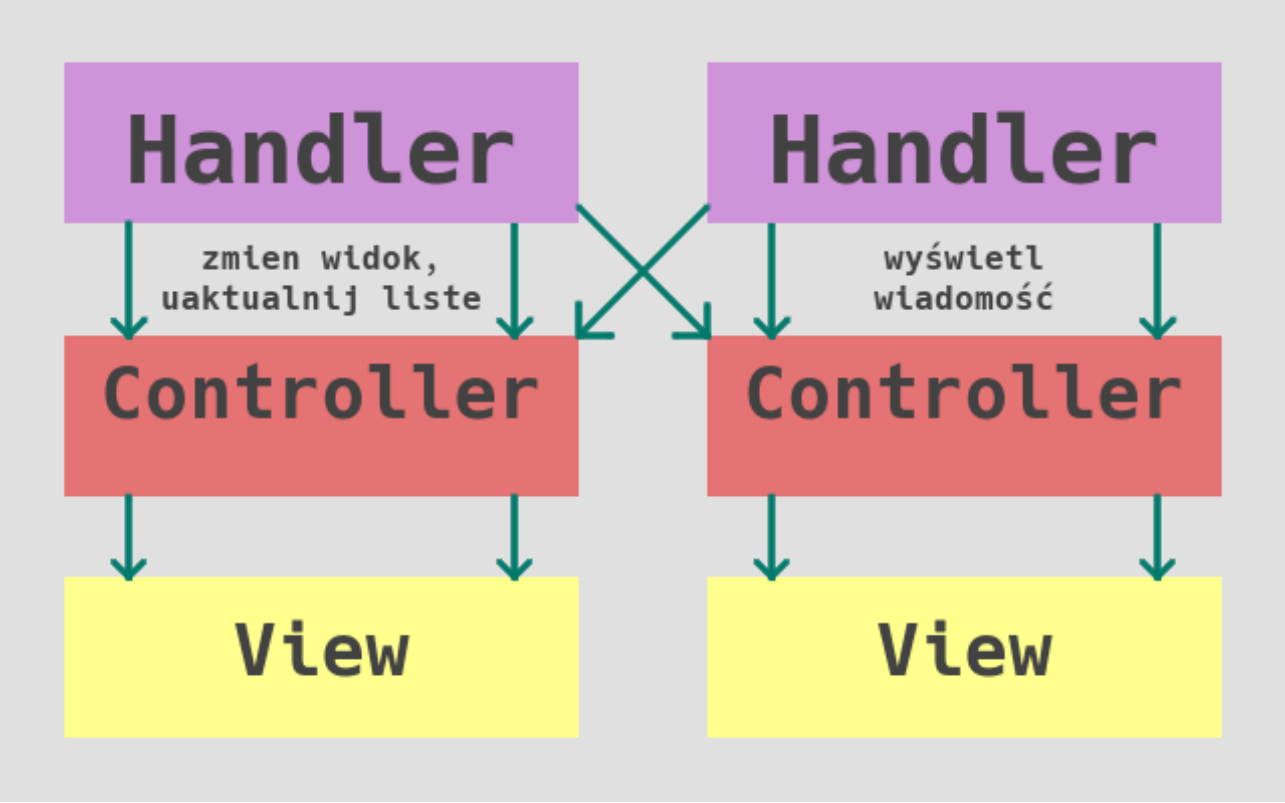
Ponieważ aplikacja jest zależna od zasobów zewnętrznych (w tym wypadku **Bluetooth**), musimy zapewnić, że w momencie gdy czekamy na reakcję z sieci (np. połączenie z użytkownikiem- co może potrwać od paru milisekund do paru minut) użytkownik cały czas musi mieć responsywną aplikację. Aby to zapewnić wszystkie akcje muszą być **asynchroniczne**.

Dlatego wprowadziliśmy do architektury tzw. **Handlery**. Aplikacja oczekuje na wykonanie zadania. W momencie gdy zadanie zostanie wykonane (np. użytkownik zostanie połączony z innym), wiadomość zostaje wysłana do handlera który obsługuje sytuację.3

W naszej aplikacji posiadamy dwa handlery.

- Pierwszy odpowiada za łączenie z innymi użytkownikami - **IConnectionHandler**

  Jego zadaniem jest obsługiwanie wydarzeń takich jak pobranie listy dostępnych w pobliżu użytkowników aplikacji oraz reakcja na połączenie z którymś z nich. Dla przykładu, w momencie gdy zostanie nawiązane połączenie **ConnectionHandler** powinien zmienić widok z listy dostępnych użytkowników na widok wysłanych i odebranych wiadomości.
- Kolejny to **IMessageHandler** - odpowiada za reakcję na wiadomości odebrane od innego użytkownika. Wiadomości takie powinny zostać wyświetlone w czytelnej formie na ekranie telefonu.



Sam layout czyli ułożenie przycisków i innych elementów UI będą wspólne dla obu platform. Ma to swoje zalety w postaci szybszego i łatwiejszego procesu tworzenia aplikacji.

Podział prac

- Rafał Ziembiński dostał za zadanie napisanie **IBluetoothManagera** dla systemu Android,
- Klaudia Głocka dostała za zadanie napisanie **IConnectionHandlera,**

- Konrad Chojnecki dostał za zadanie napisanie **IMessageHandlera,**
- Wojciech Polak dostał za zadanie napisanie **IBluetoothManagera** dla systemu iOS.