Interfejs głosowy dla aplikacji mobilnej Radio1.7

# Skład zespołu

Michał Kleszczyński

Beata Kotra

WIMIR, Inżynieria Akustyczna

# Cel projektu

Celem projektu było zaprojektowanie funkcjonalnego i intuicyjnego systemu obsługi dla aplikacji mobilnej na telefony z systemem Android umożliwiającej słuchanie interntowej studenckiej rozgłośni radiowej Radio1.7.

Grupą docelową aplikacji są młodzi użytkownicy. Głównie studenci, osoby w przedziale wiekowym pomiędzy 19 a 30 lat. System powinien umożliwiać włączenie lub wyłączenie odtwarzacza, zwiększenie lub zmniejszenie głośności systemowej dla multimediów oraz otrzymanie informacji o aktualnie odtwarzanym utworze.

# Założenia projektu

# Wykorzystywane systemy oraz architektura aplikacji

## Wykorzystywane systemy

Aplikacja jest napisana z myślą o systemie Android, stąd też całość systemu napisana jest w języku programowania Kotlin stworzonym przez producenta oprogramowania – firmę JetBrains. Dostęp do kodu źródłowego zdecydowanie ułatwił proces tworzenia całego systemu, gdyż nie trzeba było wykorzystywać zewnętrznych narzędzi do ingerencji w aplikację.

Do obsługi interfejsu głosowego wykorzystano **Android Speech API**, jako nakładkę programową na **Cloud Speech to Text** oraz **Cloud Text to Speech** od firmy Google. Aplikacja działa na telefonach z systemem Android, oczywistą decyzją było więc wybranie systemu rozpoznawania i syntezy mowy od firmy produkującej ten system. Jako że Speech API jest systemem rozwijanym od długiego czasu wybranie go jako system obsługujący aplikację było decyzją, która sprawiła że jakiekolwiek problemy, które pojawiły się podczas procesu tworzenia aplikacji były rozwiązywane w bardzo krótkim czasie.

## Opis architektury aplikacji

Aplikacja jest oparta o wzorzec MVVM (Model-View-ViewModel). Dodanie nowych funkcjonalności sprowadzało się do stworzenia odpowiednich metod i obsługi zdarzeń.

Do odtwarzacza został dodany przycisk z ikoną mikrofonu. Aktywowanie usługi sterowania głosowego obdywa się poprzez jego wciśnięcie. Od tego momentu aplikacja będzie reagować na wydawane jej polecenia, a ikona mikrofonu zmieni się na ikonę przekreślonego mikrofonu, tak aby można było w dowolnej chwili przerwać działanie usługi sterowania głosowego.

|  |  |
| --- | --- |
| baseline_mic_black_18dp.png  Rysunek 1. System nieaktywny. Wciśnij, aby włączyć. | baseline_mic_off_black_18dp.png  Rysunek 2. System aktywny. Wciśnij aby wyłączyć. |

## Dialog użytkownik – aplikacja

System reaguje na wszystkie opcje podane po ukośnikach. W przypadku odpowiedzi systemu, losowo odtwarzana jest jedna z kwestii wypisanych na powyższym schemacie.

# Osiągnięte sukcesy

# Napotkane problemy i ich rozwiązania

## Sposób aktywacji systemu

W trakcie planowania napotkano problem, w jaki sposób należy aktywować system. Brano pod uwagę następujące opcje:

* Poprzez specjalny przycisk aktywujący sterowanie głosowe – ta opcja trafiła do produktu końcowego
* Poprzed wypowiedzenie przez użytkownika odpowiedniej sekwencji słów

Obie opcje posiadają swoje wady i zalety. Pierwsza wymaga od użytkownika wykonania konkretnej akcji, czyli wciśnięcia przycisku na ekranie, przez co nie jest ona dostępna cały czas, na przykład gdy użytkownik znajduje się w pewnej odległości od telefonu lub nie ma możliwości wykonania akcji, na przykład podczas jazdy samochodem. Znacznie ogranicza to łatwość użycia oraz jej dostępność. Druga z opcji natomiast wiąże się z trudniejszą implementacją – wymagane jest ciągłe nasłuchiwanie w celu inentyfikacji hasła aktywującego system, a co za tym idzie – większe zużycie energii i wymuszenie na urządzeniu ciągle aktywowanego mikrofonu.

//Ostatecznie została wybrana pierwsza z opcji bo „No łatwiej, ale też coś madrego trzeba napisać”

## Aktywacja funkcji na podstawie frazy

Aplikacja powinna reagować na przynajmniej kilka wariantów poleceń (np. Włącz / Proszę włącz / Czy mógłbyś włączyć / Graj / Zacznij grać). Aby wykonać system parsujący tekst należałoby brać pod uwagę dziesiątki, a może nawet setki wariantów. Uwzględniałoby to każdy możliwy wariant, lecz byłoby wymagające pod kątem zasobów jak i pod względem skomplikowania kodu aplikacji.

Problem rozwiązano wykorzystując słowa kluczowe dla danego polecenia. Patrząc na powyższy przykład, słowami kluczowymi stały się „Włącz”, „Włączyć”, „Graj”. Jeżeli któreś z nich zostanie znalezione w tekście wypowiedzianym przez użytkownika system decyzyjny wykona odpowiednią akcję.

## Aktywacja kilku opcji jednocześnie

Kolejnym problemem była decyzja, jak powinien reagować system jeśli użykownik poda w polecniu dwa słowa klucze odnoszące się do różnych akcji. Przykładowo jeśli powie „start stop” – system powinien najpierw rozpocząć, a później zatrzymać odtwarzanie.

Rozwiązaniem problemu była hierarchizacja akcji. Akcje mają ustawiony priorytet, który prezentuje się w następujący sposób:

## Działanie na różnych wersjach systemu Android

Niektóre fragmenty aplikacji należało napisać w różny sposób dla różnych wersji systemu android z racji, iż system ten na przestrzeni lat zmieniał się, tak też zmieniało się API obsługi głosowej.

// Kurde, wiem o co mi chodzi ale za bardzo nie wiem jak to napisać

// Jak jeszce masz pomysła to pisz

# Wnioski z projektu

# Zgody

Wyrażamy zgodę na prezentację wyników projektu, którego dotyczy ten raport na stronie Zespołu Przetwarzania Sygnałów, stronie przedmiotu oraz w ramach wydarzeń promujących uczelnię, takich jak Noc Naukowców, Dzień Otwarty AGH, Festiwal Nauki.

Wyrażamy zgodę na publikację naszych imion i nazwisk podczas prezentacji wyników projektu, którego dotyczy ten rapory.