Spis treści

[Wstęp i cel pracy 2](#_Toc27504431)

[Opis rozwiązywanego problemu 3](#_Toc27504432)

[Specyfikacja problemu 3](#_Toc27504433)

[Przegląd istniejących rozwiązań 4](#_Toc27504434)

[Aspekty technologiczne i metodyka realizacji projektu 5](#_Toc27504435)

[Wykorzystane technologie 5](#_Toc27504436)

[Android Studio 5](#_Toc27504437)

[Java 6](#_Toc27504438)

[Gradle 6](#_Toc27504439)

[ADB 6](#_Toc27504440)

[Firebase 7](#_Toc27504441)

[GIMP 7](#_Toc27504442)

[Dodatkowe biblioteki 7](#_Toc27504443)

[Biblioteka TarsosDSP 7](#_Toc27504444)

[Biblioteka Picasso 8](#_Toc27504445)

[Specyfikacja wymagań funkcjonalnych aplikacji 8](#_Toc27504446)

[Implementacja systemu 10](#_Toc27504447)

[Przedstawienie systemu 10](#_Toc27504448)

[Strojenie gitary 10](#_Toc27504449)

[Metronom 10](#_Toc27504450)

[Profil użytkownika 11](#_Toc27504451)

[Nauka akordów 12](#_Toc27504452)

[Ranking 12](#_Toc27504453)

[Kompozycja struktury folderów 13](#_Toc27504454)

[Implementacja stroika 15](#_Toc27504455)

[Implementacja metronomu 16](#_Toc27504456)

[Tworzenie bazy i uzupełnienie bazy danych z poziomu aplikacji 17](#_Toc27504457)

[Administracja użytkownikami 20](#_Toc27504458)

[Sprawdzenie poprawności akordów 23](#_Toc27504459)

[Wyświetlenie rankingu z podziałem na poszczególne grupy 25](#_Toc27504460)

[Prezentacja systemu 27](#_Toc27504461)

[Obsługa aplikacji przez użytkownika niezalogowanego 27](#_Toc27504462)

[Obsługa aplikacji przez użytkownika zalogowanego 27](#_Toc27504463)

[Podsumowanie 27](#_Toc27504464)

# Wstęp i cel pracy

Co raz więcej czasu poświęcamy telefonom komórkowym co nie uchodzi uwadze projektantom systemów informatycznych. Rosnąca ilość nowych aplikacji mobilnych, dostępnych chociażby na platformie Google Play, oraz stale rosnąca ilość pobrań aplikacji zdaje się potwierdzać, że ściągamy i korzystamy z coraz liczniejszych aplikacji. Przeglądając platformę Google Play natknąć się można na wiele kategorii aplikacji i gier. Możemy tam znaleźć aplikacje do zarządzania czasem, wspomagające zapamiętywanie, do nauki języków. Przeglądając nowe aplikacje można zauważyć, że aplikacje do nauki gry na jakimś instrumencie zdają się pojawiać co raz częściej.

Największą popularnością wśród instrumentów od wielu lat cieszy się gitara. Wpływ na taki stan rzeczy może mieć fakt, że gitara jest bardzo prosta w swojej budowie, stosunkowo tania oraz szybkość opanowania nowych akordów na tym instrumencie jest dość szybka. Problemem jaki mogą napotkać gitarzyści samouki może być w nastrojeniu gitary, trzymanie tempa granych akordów/dźwięków czy nawet poprawność trzymania akordu. Ostatni problem można podzielić na dwa mniejsze problemy. Otóż pierwszy z nich to ułożenie palców na gryfie, a drugi to siła z jaką przykładane są palce do gryfu. Jeśli zbyt słabo naciśniemy strunę to dźwięk, który wydobędzie się z pudła rezonansowego będzie niewłaściwy.

Celem projektu jest stworzenie aplikacji, która ułatwi niedoświadczonym gitarzystom stawić pierwsze kroki do nauki na prostym instrumencie, którym jest gitara. Głównym zadaniem ma być pomoc przy nastrojeniu instrumentu posługując się dźwiękami nagrywanym przez urządzenie. Przy strojeniu gitary, potrzebne są obie ręce. Jedna z nich szarpie za struny, a druga odpowiednio manewruje kluczami. Z tego też powodu aplikacja powinna pozwolić na nastrojenie gitary, bez trzymania telefonu w ręce. Drugim ważnym aspektem przy nauce gry na gitarze jest trzymanie odpowiedniego tempa. Pomocnym urządzeniem do tego zadania jest metronom, który wybija dźwięk w odpowiednim tempie. Aplikacja powinna mieć możliwość zasymulowania metronomu, która jak w poprzednim wypadku, powinna być obsługiwana bez potrzeby trzymania telefonu w dłoni. Trzecim aspektem są akordy. Zaczynając od łatwych do chwycenia aż do tych, które trzeba poświęcić więcej czasu na poprawne chwycenie – każdy gitarzysta powinien znać podstawowe akordy. To one w głównej mierze odpowiadają za melodie graną w utworach. Tak więc aplikacja powinna wspomóc młodego gitarzystę na tych trzech płaszczyznach. Powinna pomóc mu nastroić gitarę, nauczyć podstawowych akordów oraz nauczyć trzymać odpowiedniego tempach podczas grania.

# Opis rozwiązywanego problemu

## Specyfikacja problemu

Stawiając pierwsze kroki w nauce gry na instrumencie wielu zastanawiało się jak to zrobić. Oczywiście mam ty na myśli tych młodych muzyków, którzy chcieli się nauczyć samemu grać na wybranym instrumencie. Przed młodym muzykiem stawiane są trudne, jak na razie, działania.

Jednym z takich działań jest nastrojenie gitary. Jest to najważniejszy z punktów, który musi przejść każdy muzyk. Jest to działanie, które musi wykonać zawsze przed zaczęciem grania na instrumencie. Jest to zadanie czasem bardzo trudnym, a zwłaszcza dla niedoświadczonych muzyków. Pierwszym punktem strojenia jest wybranie strojenia (naciągnięcie strun, w taki sposób aby dźwięk na strunie, którą szarpiemy, ale nie przyciskamy na żadnym z progów, dała odpowiedni dźwięk). W zależności od gatunku muzycznego jakiego chcemy zagrać, gitara otrzymuje inne strojenie. Gdy już wybierze odpowiedni strój gitary przechodzi do właściwego strojenia naciągając struny na klucze umieszczone na główce gitary. Młody gitarzysta, który nie ma wyrobionego na tyle słuchu aby samodzielnie nastroić instrument ma utrudnione zadanie. Musi wiedzieć jaki dźwięk chce uzyskać, a nie znając go na tyle dobrze aby określić czy dźwięk jest prawidłowy, to zadanie staje się trudne, jeśli nie awykonalne. Z pomocą przychodzą stroiki, które określając częstotliwość dźwięku ustalają dźwięk, który wydobył się z gitary. Manewrując kluczami gitarzysta przybliża się do wymaganego dźwięku.

Drugą trudnością z jaką spotka się gitarzysta-samouk jest trzymanie tempa. Jest ono bardzo istotną częścią podczas nie tylko gry na instrumencie, co nawet śpiewania czy nawet w tańcu. Przykładem tego mogą być same piosenki. Wiele z nich ma ten sam układ akordów, jednak bicie i tempo grania jest inne co sprawia, że słyszalna melodia jest inna. I w tym przypadku są odpowiednie narzędzie – metronom. Działa on na zasadzie wybijania dźwięku co jakiś określony czas. Nowoczesne narzędzia opierają się właśnie na tej zasadzie. Jednak jest to kolejne narzędzie które trzeba mieć podczas nauki gry.

Trzecią sprawą jest to, na co wszyscy młodzi gitarzyści czekają – nauka akordów. Jest to bardzo ważna część grania na gitarze. Trzymając odpowiednio palce na gryfie gitary i przyciskając dostatecznie mocno struny do gryfu, ale jednocześnie nie dotykając w żaden sposób innych strun wydobywamy upragniony dźwięk z gitary szarpiąc odpowiedni struny. Jest to dość skomplikowana rzecz do nauki. Jedna gdy opanuje się podstawy, czyli siłę nacisku na struny przy danej gitarze oraz sposób ułożenia palców, aby nie dotykały przez przypadek innych strun, nauka akordów staje się łatwiejsza. Jednak aby dalej przejść trzeba wiedzieć jak ułożyć palce na gryfie. Tu potrzebne są schematy akordów, zazwyczaj zebrane w mała książeczkę. Jednak od którego akordów zacząć? Jakie warto nauczyć się na początku, a jakie zostawić na później? Są to bardzo ważne kwestie, ponieważ zbyt trudne akordy na początku mogą zniechęcić młodych gitarzystów.

Podsumowując te trzy punkty. Mamy w kieszeni stroik, metronom oraz książeczkę z schematami akordów. Jest to bardzo dobry początek. Co jednak jak stroik albo metronom zgubi się gdzieś? Te nowe urządzenia są zazwyczaj małe, więc i zgubić je jest bardzo łatwo. Otwiera książeczkę akordów i zobaczywszy ich masę oraz dziwne i niezrozumiałe znaki od razu zamyka książkę. Gdyby tak tylko to ułatwić. Dlatego też powstają aplikacje takie jak ta. Mająca stroik, metronom i spis akordów w aplikacji dostępnych na wyciągnięcie ręki w telefonie komórkowym typu smartphone.

Taki zestaw dla początkujących jest nie raz wybawieniem, jednak potrzeba czegoś więcej. Trzeba ich zachęcić, potrzymać zapał do nauki gry na gitarze. Jak to zrobić? Nie od dziś wiadomo, że rywalizacja jest najlepszym sposobem, aby zmotywować kogoś do działania. Tak jesteśmy stworzeni – kiedyś determinowało to o naszym przetrwaniu (rywalizacja o jedzenie, o miejsce do spania, o miejsce w hierarchii). Dlatego też uważam, że to jest bardzo dobry sposób do zachęcenia nowych gitarzystów do nauki gry na tym bardzo prostym instrumencie jakim jest gitara. W swojej aplikacji wprowadzę system rankingu dla zarejestrowanych użytkowników aplikacji z podziałem na grupy akordów (od najłatwiejszych do tych bardziej skomplikowanych akordów) oraz podsumowanie punktów dla wszystkich grup.

## Przegląd istniejących rozwiązań

Aplikacją cieszącą się największą popularnością dostępną na platformie Google Play jest „GuitarTuna” od „Yousician Ltd.”. Korzystałam z niej dość duży okres i mogę stwierdzić, że do tej pory jest ona najlepsza wśród dostępnych aplikacji. Posiada ona stroik gitarowy, metronom ze zmiennymi parametrami oraz naukę akordów. Wszystkie funkcjonalności są bardzo dobrze zrobione. Aby nastroić gitarę nie trzeba zmieniać strun, nauka akordów wskazuje, która struna brzmi inaczej niż powinna.

Aplikacja ta jest bardzo dobrze wykonana, jednak ma kilka wad. Pierwszą z nich jest brak dostępu do wszystkich funkcjonalności. Nauka akordów jest ograniczona do kilku podstawowych akordów, reszta jest dostępna po wykupieniu usługi premium. Drugą bardzo poważną wadą jest waga aplikacji. Dla urządzeń mobilny opartych o systemy Android jest to bardzo poważna wada. Ciągłe informacje o braku miejsca na urządzeniu sprawiają, że użytkownicy zmuszeni są do kupna nowego urządzenia lub usunięcia aplikacji lub do zmiany systemu. Ciągłe wyświetlanie reklam jest problemem większości aplikacji dostępnych na platformie Google Play. Jest to aż tak duży i częsty problemem, że firma Google postanowiła walczyć z twórcami aplikacji, którzy nadużywają wyświetlania reklam.

Kolejną aplikacją wartą uwagi jest „Ultimate Tuner” od „Tabs4Acoustic – Free guitar tools”. Ta aplikacja skupia się wyłącznie na strojeniu. Mamy do wyboru większość strojeń gitarowych z przeróżnych typów muzycznych oraz różne gitary od gitary klasycznej do hawajskiej włącznie). Po kliknięciu w struny dostajemy dźwięk nastrojonej gitary i tak możemy nastroić ją ze słuchu. Aplikacja ta jest bardzo rozwinięta od strony strojenia, jednak jest to nie wystarczające jeśli chodzi o narzędzia wspomagające naukę gry na gitarze. Z tego też powodu powstaje problem z miejscem na urządzeniu, na którym trzeba zainstalować kolejne aplikacje.

Wiele innych aplikacji nie oferują strojenia poprzez mikrofon przykładem może być aplikacja „Gutar Tuner” od „appsmz”. Działa ona na zasadzie puszczania dźwięku po kliknięciu w daną strunę. Dźwięk można zapętlić poprzez kliknięcie w przycisk w górnej części ekranu. Dla początkujących jest to duże utrudnienie , ponieważ nie są w stanie stwierdzić czy dźwięk puszczany z aplikacji jest taki sam jak dźwięk uzyskany po szarpnięciu za strunę gitary, którą trzyma w ręku. Takie typy aplikacji do strojenia gitary są raczej dla zaawansowanych gitarzystów, jednak dla młodych w ogóle się nie sprawdzają.

Przeglądając Sklep Play pod kątem aplikacji wspomagających gry na gitarze odebrałam wrażenie, że nie ma takiej aplikacji, która zachęcałaby do kontynuacji nauki gry na gitarze. Są aplikacje rozbudowanie i kuszące funkcjami nauki gry akordów, są też takie, które oferują tylko dane funkcjonalności rozwijając je do potężnych narzędzi. Jednak żaden z tworzących tych aplikacji nie zastanowił się nad przyciągnięciem użytkownika nie tyle co do aplikacji, ale do samego grania na instrumencie co sprawiłoby, że ich aplikacja byłaby uruchamiana częściej z znacznie większym zapałem. Użytkownicy mając takie narzędzie, które motywowało by ich do stawiania nowych kroków do ich celów, staliby się lojalnymi odbiorcami, którzy pomogli by w rozwoju samej aplikacji podpowiadając kierunki w jakim dana aplikacja mogłaby się rozwinąć.

# Aspekty technologiczne i metodyka realizacji projektu

## Wykorzystane technologie

### Android Studio

Android Studio to najpopularniejsze IDE wykorzystywane w procesie tworzenia aplikacji mobilnych w technologii Android. Narzędzie te w procesie tworzenia aktywności oraz fragmentów proponuje użytkownikowi wiele szablonów. Decydując się na stworzenie aktywności bądź fragmentu z dostępnych szablonów generuje się kod, który jest otoczony potrzebnymi zdarzeniami w samym kodzie Javy lub Kotlina (w zależności jaki język wybraliśmy podczas tworzenia projektu) oraz powiązane pliki xml.

Do tworzenia layoutów wykorzystuje się język XML, jednak Android Studio oferuje tworzenie poprzez graficzny interfejs. Po lewej stronie posiada spis wszystkich atrybutów, co znacznie ułatwia prace nad wyglądem aplikacji.

Jako, że Android Studio oparty jest na IDE od JetBrains posiada takie same funkcjonalności jak inne IDE od wydawcy. Nagodna uwagę funkcjonalność jest łatwe zarządzanie repozytorium zdalnym od GitHuba. Możemy łatwo utworzyć repozytorium, commitować i pushować zmiany, ale też możemy cofać wersje do odpowiedniego commita.

Android Studio dostarcza też możliwość otworzenia aplikacji na emulatorze. Aby to zrobić najpierw tworzymy urządzenie o wybranych przez nas parametrach, wybieramy mu system operacyjny. Następnie możemy uruchomić bildowanie i uruchomienie aplikacji poprzez naciśnięcie przycisku „Run” zaraz koło listy rozwijanej z dostępnymi urządzeniami.

Kolejną funkcjonalnością wartą uwagi jest asystent Firebase’a. Z jego pomocą utworzymy projekt na Firebase, nawiążemy połączenie z istniejącym projektem, dodamy odpowiednie zależności do pliku gradle i dostaniemy przykładowy kod na łączenie się z funkcjonalnościami z Firebase’a.

### Java

Wysoko poziomowy język programowania utworzony przez Sun Microsystems w 1995 roku. Język ten wykorzystywany jest w aplikacjach deskoptowych, aplikacjach mobilnych oraz aplikacjach webowych, czy nawet dla systemów wbudowanych takich jak karty SIM, telefony VOIP czy odtwarzacze dysków Blu-ray. Często jest wykorzystywana w systemach bankowych co tylko potwierdza bezpieczeństwo jakie daje ten język programowania. Jest usadowiony na czołowych miejscach w ostatnich latach w rankingu najpopularniejszych języków programowania.

Java odznacza się szybkością, bezpieczeństwem oraz solidnością. Język ten opiera się głównie na obiektowości z wyjątkiem kilku struktur prostych.

Java polepszyła swoją wydajność poprzez zastosowanie kompilatora JIT (Just in time), który te część kodu który jest najczęściej używany potrafi zoptymalizować. Pomiędzy mechanizmami, które umożliwiają takie działania jest dobrze znany Garbage collection. Działa aby zwolnić pamięć urządzenia z niepotrzebnych obiektów. Niepotrzebne obiekty uważa za takie obiekty, do których nie ma dostępu. Brak referencji oznacza nie potrzebność danego obiektu. Te działanie wykonuje się automatycznie nie zależnie od użytkownika aplikacji czy nawet programisty. W aplikacji używana będzie Java w wersji 8. Jest to bardzo powszechna praktyka, aby tą wersje Javy łączyć z aplikacjami na systemy Android.

### Gradle

Jest to narzędzie służące do zautomatyzowania procesu budowania projektu. Za pomocą jednej linii zaciągane są odpowiednie zależności oraz w szybki sposób uruchamiane są testy aplikacji. Gradle jest najczęściej wybieranym narzędziem do zautomatyzowanego budowania w aplikacjach androidowskich. Inne narzędzia podobne do Gradle to równie popularny Maven, Ant oraz Make.

### ADB

Ten interfejs pozwala na wiele operacji na urządzeniu mobilnym. Jednak najważniejszą rzeczą, do której najczęściej wykorzystuje się ADB to możliwość instalowania aplikacji bezpośrednio na urządzenie mobilne. Wystarczy włączyć na urządzeniu tryb debugowania i uruchomić w oknie poleceń komendę adb install <ścieżka do pliku apk>. Jednak Android Studio pozwala na uruchomienie komendy poprzez jeden przycisk „Run” zaraz koło listy rozwijanej z dostępnymi urządzeniami. Interfejs ten pozwolił na łatwe testowanie aplikacji przez użytkowników. Pomógł też w sprawdzeniu poprawności dźwięku. Na symulatorze telefonu komórkowego na komputerze, aplikacja nie była w stanie działać poprawnie. Dźwięk, który symulator odbierał był zestawem cyfrowych sygnałów, w większości przypadków o jednej częstotliwości co uniemożliwiało przetestowanie działania stroika. Dlatego też należało aplikacje zainstalować na urządzeniu fizycznym i sprawdzić działanie aplikacji.

### Firebase

Jest to platforma wspierana posiadająca wiele funkcjonalności, które wspomagają pracę nad aplikacjami mobilnymi oraz webowymi. Łatwość zarządzania danymi przez intuicyjny graficzny interfejs sprawia, platformą interesuje się coraz większe grono programistów.

Jest ona klasyfikowana jako BaaS (z ang. Backend as a Service). Dostarcza on przeróżne narzędzia wspomagające tworzenie aplikacji. Dzięki temu nie musimy na nowo tworzyć systemu logowania i rejestracji czy wysyłki wiadomości email. Platforma ta udostępniając takie przydatne funkcje jak Cloud Function, Authentication, Hosting czy nawet Cloud Messaging przyspiesza tworzenie oprogramowania przez dostarczenie przez te funkcje podstawowe funkcjonalności każdej współczesnej aplikacji. Najważniejszą zaletą tej platformy jest widoczny rozwój platformy, przez co tak szybko zdobywa popularność.

W aplikacji użyłam takie funkcje jak: Authentication (pozwala na łatwe zarządzanie zarejestrowanymi użytkownikami), „Cloud Firestoire” (jest to baza danych, która pozwala na ładowanie danych w czasie rzeczywistym, jest to baza NoSQL) oraz „FirebaseStorage” (jest to miejsce, w którym możemy przechowywać pliki, które później możemy pobierać w naszej aplikacji).

### GIMP

Pierwsza wersja tego programu została wydana w listopadzie 1995 r. i nosił nazwę „The GIMP” przez Petera Mattisa. Program został oparty na bibliotece Motif jednak autor zaznaczał, że późniejsze wersje nie będą oparte na tej bibliotece. Tak też się stało. W wydaniu rozwojowym rozpoczęto korzystanie z autorskich bibliotek. Warto też wspomnieć, że GIMP pozwala na zaimplementowanie własnych funkcjonalności poprzez skrypty w językach Scheme, Perl, Ruby i Python.

Jest to darmowa aplikacja służąca do obróbki plików graficznych. Licencja tej aplikacji pozwala na korzystanie z programu przez użytkownika prywatnego jak i do wykorzystania do celu komercyjnych co jest największą zaletą tego programu. To środowisko rozwijane przez wielu programistów i jest na tyle rozbudowane, że stało się bardzo poważną konkurencją dla tego środowiska jak Adobe Photoshop. Kolejną zaletą tego środowiska jest przyjemny dla użytkownika interfejs co było w założeniu tworzenia projektu. GIMP posiada podstawowe funkcje jak malowanie, zaznaczanie, mazanie/usuwanie, ale też te bardziej złożone jak warstwy, złożone pędzle do malowania, skalowanie, usuwanie szumów z obrazów czy operacje na plikach. Za pomocą jednego kliknięcia jesteśmy wstanie wyeksportować zdjęcie do wszystkich podstawowych rozszerzeń graficznych.

W projekcie, GIMP został użyty do stworzenia schematów akordów gitarowych.

## Dodatkowe biblioteki

### Biblioteka TarsosDSP

Jest to biblioteka typu opensource do przetwarzania dźwięku. Cała biblioteka, jeśli było to możliwe, opiera się na czystej Javie. Na repozytorium tej biblioteki możemy uzyskać lik do przykładowych programów, stworzonych na podstawie tej biblioteki. Możemy zobaczyć takie aplikacje jak: detektor głośności dźwięku, spektrogram oraz gra polegająca na zaśpiewaniu dźwięku najlepiej jak to możliwe. Aby dołączyć tą bibliotekę należy ze strony githuba projektu pobrać najnowszą wersję biblioteki i zapisać ją do folderu libs w folderze app. Następnie do pliku build.gradle w module app dopisać tą linię:



Ważne aby dopisać to w obiekcie dependencies w którym zapisane są wszystkie zależności w projekcie.

### Biblioteka Picasso

Jest to biblioteka służąca do dołączania obrazów do aplikacji. Dołączane zdjęcia mogą znajdować się w folderze zasobów aplikacji lub na zewnętrznym serwerze. Dołączanie zdjęć do aplikacji staje się znacznie łatwiejsze niż przez standardowy plik XML. W przypadku gdy chcemy dołączyć zdjęcie bezpośrednio do znacznika XML plik z obrazkiem musi znajdować się w folderze zasób aplikacji lub androida. Używając biblioteki Picasso dołączanie i ustawianie wartości dla zdjęcia odbywa się w często jednej linii kodu. Standardowym kodem jest:



Aby dołączyć tą bibliotekę wystarczy do pliku build.gradle w module app dopisać tą linię, gdzie liczbami na końcu oznaczona jest aktualna, podczas tworzenia aplikacji, wersja biblioteki:



Ważne aby ta linia znalazła się w obiekcie dependencies w którym zapisane są wszystkie zależności w projekcie.

## Specyfikacja wymagań funkcjonalnych aplikacji

Najważniejszą funkcją aplikacji jest możliwość interakcji z nią bez użycia rąk. Jest to ważne ponieważ strojąc gitarę użytkownik nie jest wstanie trzymać telefonu w dłoni i wykonywać jakikolwiek działań na telefonie. Drugą kwestią jest przejrzysty interfejs. Jest to motywowane tymi samymi pobudkami. Strojąc gitarę użytkownik aplikacji położy telefon komórkowy w dogodnej dla niego odległości. Trzeba umożliwić użytkownikowi na swobodne działanie nie ograniczając go przywiązując go do telefonu.

Aplikacja powinna oferować stroik dla kilku strojeń. Pierwszy z nich powinien być standardowy, kolejne muszą być najpopularniejsze w swoim gatunku muzycznym. Po przeprowadzeniu rozpoznania oprócz strojenia standardowego (Standard – EADGHe) będzie dostępny strój E-Flat (EbG#C#F#BbEb – dźwięk obniżony o półtonu) oraz Drop-D (DGCFAD – dźwięk obniżony o cały ton). Działanie tego stroika powinno mieć dwa sposoby działań. Pierwszy z nich to strojenie jednej struny. Powinno być to możliwe po wyborze struny. Poprawność dźwięku dla tej struny powinna być mierzona cały czas dopóki nie zmieni się struny lub nie wejdzie się w drugi sposób działania stroika. Drugie działanie powinno umożliwiać nastrojenie wszystkich strun bez konieczności przełączania się między strunami. W tym trybie sprawdzanie poprawności dźwięku powinno odbywać się dla wszystkich strun wskazując poprawność dla poszczególnych. Jeśli dźwięk który użytkownik zagra jest bardzo zbliżony do którejś ze strun aplikacja powinna zaznaczyć tą właśnie strunę i wskazać o ile trzeba zmienić dźwięk (manewrując przy kluczach gitary) aby uzyskać odpowiedni strój dla danej struny.

Kolejną funkcją która powinna oferować aplikacja jest metronom. Metronom jako proste narzędzie do mierzenia tempa jest wręcz niezbędny do nauki gry na gitarze. Metronom powinien oferować możliwość zmiany tempa nawet w trakcie swojego działania. Dużym ułatwieniem byłoby wyboru metrumdzięki czemu niedoświadczony gitarzysta będzie w stanie określić kiedy powinien zmienić dźwięk który gra pomagając jednocześnie grać płynnie. Stanie się tak poprzez zastosowanie dwóch różnych dźwięków, które by oznaczały początek taktu lub jego środek. Wybierając metrum 4/4 w cyklu będą cztery dźwięki. Pierwszy z nich będzie odznaczał się zmianą dźwięku, a pozostałe trzy będą miały ten sam dźwięk. Analogicznie dziać się będzie się tak dziać dla metrum np. 5/8. Będzie pięć dźwięków w jednym cyklu i tylko jeden z nich, oznaczający początek nowego cyklu, będzie oznaczał się innym dźwiękiem. Metronom ten powinien mieć możliwość zmiany tempa wybijanych dźwięków poprzez wybór pomiędzy dostępnymi tempami (np. Largo, Allegro, Presto) oraz pozwolić użytkownikowi łatwo określić własne tempo poprzez suwak lub przyciski +/-.

Funkcją skłaniającą do dalszej nauki będzie ranking zalogowanych użytkowników aplikacji. Będzie ona wyświetlać ranking w zależności od grupy akordów lub podsumowujący. Będzie on wyświetlał dane użytkowników takie jak nazwa użytkownika, ikona użytkownika i jego punkty zebrane w danej kategorii. Każdy użytkownik zalogowany będzie miał możliwość przeglądnięcia poszczególnych kategorii rankingów.

Użytkownik aplikacji powinien mieć możliwość zarządzanie własnym profilem. Aplikacja powinna umożliwić zalogowanym użytkownikom na zamianę takich danych jak nazwę oraz ikonę. Nazwa będzie zmieniana poprzez udostępnienie pola tekstowego z wpisanym aktualną nazwą, a ikona będzie możliwa do zmiany poprzez kliknięcie w aktualną ikonę i wybór pomiędzy dostępnymi ikonami.

Początkujący gitarzysta powinien mieć możliwość nauki akordów. Taki tryb zostanie dostarczony przez aplikacje. Po wyborze odpowiedniej grupy, następnie danego akordu, użytkownikowi wyświetli się schemat akordu oraz 6 pasków symbolizujące struny. Po porwanym zagraniu dźwięku odpowiednia struna zmieni kolor na zielony. Jeśli wszystkie struny zmienią kolor na zielony użytkownik zostanie poinformowany o poprawnym zagraniu akordu.

Użytkownik chcący brać udział w rankingu będzie spytany przy wyborze akordu w który tryb chce wejść. Wybierając ranking czas zostanie puszczony wraz z każdą próbą, a schemat zostanie ukryty. Klikając w przycisk użytkownik może wyświetlić schemat, jednak punkt możliwe do zdobyć zostaną zmniejszone. W zależności od czasu jaki użytkownik będzie potrzebował na zagranie danego akordu punkty zostaną odpowiednio przydzielone. To samo dotyczy się ilości podejść do jednego akordu. Im dłużej chcemy podejść do jednego akordu tym punkty dostępne do zdobycia są mniejsze. Będzie to miało na celu wyeliminowanie problemu z nieuczciwymi użytkownikami, którzy poprzez granie jednego akordu z grupy będą chcieli zdobywać punkty, co byłoby nieuczciwe względem innych użytków.

# Implementacja systemu

## Przedstawienie systemu

### Strojenie gitary

Zgodnie z diagramem **todo**, użytkownik uruchamiając okno aplikacji odpowiedzialne za nastrojenie gitary może wybrać rodzaj strojenia. Aby nastroić gitarę ma dwie możliwości. Pierwszą z nich jest nastrojenie gitary w sposób taki, że wybierze daną strunę sprawdzi jej częstotliwość i gdy częstotliwość będzie zadowalająca zmieni strojoną strunę, a w przypadku ostatniej zakończy proces strojenia. Drugim wyjściem jest włączenie trybu auto gdzie użytkownik może szarpać struny dowolnie i gdy częstotliwość na którejś z nich będzie zadowalająca to przejdzie do kolejnej bez konieczności operacji na urządzeniu lub wyłączy dane okno.

Działanie stroika dla strojenia pojedynczej struny i strojenia automatycznego różni się w działaniach wykonywanych przez aplikację (przedstawione na diagramie **todo** oraz **todo**). Najważniejszą zmianą jest sposób przypisania wymaganej częstotliwości. W pierwszym przypadku wymagana częstotliwość jest zapisywana w zmiennej, podczas gdy w drugim podejściu wymagane częstotliwości są przypisywane do danej struny. Umożliwia to później sprawdzenia granej przez użytkownika częstotliwości którego dźwięku jest najbliżej i zaznacza się odpowiednia struna.

Pliki, które oferują dane działania są zawarte w diagramie **todo.** Klasa oferująca wygląd okna i jego działanie jest zawarta w klasie TunerActivity, który dziedzicząc po klasie Activity pozwala na użycie metod działających w danym stanie (np. metoda onCreate jest wykorzystywana podczas tworzenia wyglądu aplikacji łącząc kod języka Javy z plikiem XML tworzącego rozmieszczenie elementów). Do tej klasy dołączane są inne klasy oferujące działanie w poszczególnych aspektach. Z klasy Notes pobierana jest lista strun połączona z wymaganą częstotliwością, w klasie Permission oferowane są działania odpowiedzialne za poprawne działanie aplikacji pod kątem pozwoleń użytkownika. Klasa Premission odpowiada za sprawdzenie czy dane pozwolenie jest włączone. W innym przypadku prosi o pozwolenie, jeśli zostanie odmówione blokuje działanie aplikacji. W klasie NavigationBottom znajduje się szereg działań sprawiających, że dolna nawigacja w aplikacji działa poprawnie.

### Metronom

Zgodnie z diagramem **todo**, użytkownik ma możliwość włączyć oraz wyłączyć działanie metronomu. Podczas działania metronomu użytkownik ma możliwość zmiany tempa (w zakresie od 40 do 208) oraz zmiany metrum wśród podanych.

Główne działanie aplikacji nie może być zaburzone lub zablokowane, dlatego też zgodnie z diagramem **todo** metronom powinien zostać uruchomiony w nowym wątku. Dzięki czemu możliwa jest zmiana tempa i metrum w czasie działania wątku z metronomem. Dlatego też klasa Metronome, przedstawiony na diagramie **todo**) implementuje interfejs Runnable dzięki czemu w klasie MetronomeActivity może zostać uruchomiony wątek, który działa w sposób opisany w klasie Metronome. Klasami pomocniczymi w tym zestawieniu są TempoRange oraz TempoStatistic. W klasie TempoStatistic określane są dane na temat danego wyróżnianego tempa, natomiast w TempoRange tworzona jest lista tych obiektów reprezentująca 8 podstawowych temp. Dzięki pracy tych dwóch klas użytkownik może wybrać wśród podanych tempo i zostaje ono ustawione w wątku metronomu jako najmniejsza wartość w danym przedziale. Do MetronomeActivity dołączana jest klasa NavigationBottom odpowiedzialna za nawigacje dolną aplikacji.

### Profil użytkownika

Wyróżniane są dwa rodzaje użytkowników (diagram **todo**). Pierwszy z nich to użytkownik niezalogowany. Może on się zalogować, zarejestrować oraz korzystać z funkcji strojenia oraz metronomu. Drugich z nich oprócz działań, które może wykonywać użytkownik niezalogowany ma dostęp do zarządzania swoim profilem (wyświetlanie profilu, zmianę nazwę oraz awatara), wylogować się oraz korzystać z nauki akordów i rankingu.

Aby dodać nowego użytkownika (diagram **todo**) należy po wypełnieniu trzech pól (mail oraz dwa razy wpisać hasło) następuje zapytanie do bazy czy podany mail już istnieje, jeśli tak zostaje zwrócona informacja o nie powodzeniu autentyfikacji, w innym przypadku użytkownik zostaje przekierowany do okna profilu w aplikacji. Do logowania zostaje zastosowany ten sam schemat działania z tym, hasło nie jest powtórzone (diagram **todo**).

Użytkownik może dokonać dwóch zmian w swoim profilu. Pierwszym z nich jest dokonanie zmian w nazwie użytkownika (diagram **todo**). Nazwa użytkownika jest wykorzystywana w wyświetleniu rankingu. Po wpisaniu nową nazwę użytkownik zatwierdza zmianę poprzez kliknięcie w przycisk co wywołuje zapytanie do bazy. Jeśli zapytanie zostanie poprawnie wykonane to na urządzeniu użytkownika wyświetli się informacja o poprawnym przebiegu zmian, w innym przypadku zostanie poinformowany o błędzie. Kolejną rzeczą możliwą do zmiany oraz wykorzystywaną do wyświetlenia rankingu jest awatar użytkownika (diagram **todo**). Może on zmieniać swoją ikonę z dostępnych. Odbywa się to na podobnej zasadzie co zmiana nazwy użytkownika. Jednak użytkownik nie zatwierdza zmiana poprzez kliknięcie w przycisk a w zdjęcie, które wysyła do bazy adres URL danej ikony, który zostaje przypisany do danego użytkownika.

Na diagramie **todo** przedstawione są klasy biorące udział w prezentacji i działaniu opisanych sytuacji. Za wygląd okien aplikacji odpowiadają klasy LoginActivity (odpowiada za okno logowania), RegistrationActivity (odpowiada za okno rejestracji), ProfilActivity (odpowiada za wyświetlenie okna profilu użytkownika) oraz SelectAvatarActivity (wyświetla dostępne ikony użytkownika – do zmiany awatara użytkownika) wszystkie dziedziczą po klasie Activity. Do każdej z podanych klas (oprócz klasy ProfileActivity) dołączane są klasy NavigationBottom (odpowiedzialny za nawigacje dolną aplikacji) oraz NavigationTop (odpowiedzialną za nawigację górną aplikacji). Na tym schemacie widoczne są też klasy pomocnicze. Są to klasy User, IconUser oraz AuthFirebase. Pierwsza z nich odpowiada za stworzenie obiektu użytkownika, który posiada informacje o emailu, ikonie oraz nazwie użytkownika. IconUser odpowiada za prezentacje dostępnych ikon, łącząc adres URL z miejscem wyświetlanym w oknie SelectAvatarAcitivity. Ostatnią z pomocniczych klas jest AuthFirebase. Jest on odpowiedzialna za komunikacje pomiędzy funkcją autoryzacji na platformie Firebase z aplikacją. Ostatnią klasą przedstawioną na diagramie jest UserColelction. Jest to klasa łącząca bezę danych z aplikacją. Dzięki niej można wyciągnąć dane o użytkowniku oraz zmieniać informacje w bazie danych o danym użytkowniku.

### Nauka akordów

Do okien, które oferują naukę akordów dostęp mają zalogowani, ale też niezalogowani użytkownicy. Jednak w przypadku niezalogowanego użytkownika dostęp do funkcji jest ograniczony. Może on tylko wyświetlić listę dostępnych akordów oraz listę z podziałem na grupy. Zalogowany użytkownik może sprawdzać poprawność granych akordów w typie nauki lub w trybie rankingu (diagram **todo**). Tryb rankingu różni się od trybu nauki tym, że przydziela użytkownikowi punkty za dany akord.

Przed rozpoczęciem głównego działania zostają pobrane dane z bazy danych o wybranym przez użytkownika akordzie (diagram **todo**). W klasie łączącej bazę danych z aplikacją pobierane są dane i tworzony jest obiekt akordu z wymaganymi częstotliwościami na odpowiednich strunach oraz adres URL do schematu akordu. Następnie tworzony jest wątek sprawdzający poprawność granych dźwięków i zaznaczane są struny, których częstotliwość wymagana i zagrana jest taka sama lub różnica mieści się w granicy błędu. Gdy już wszystkie struny zostają oznaczone jako poprawne to wyświetlana jest informacja o poprawnym zagraniu oraz wyłączany jest wątek nasłuchujący poprawność zagrania. W przypadku trybu rankingu (diagram **todo**), gdy wszystkie struny zostają zaznaczone jako poprawne to naliczane zostają punkty, zwiększana liczba prób oraz zostaje wyświetlona informacja o punktach zdobytych w próbie oraz w całej rundzie. Gdy liczba prób osiągnie 4 to zostaje zakończona runda, a dane o zdobytych punktach zostają przesłane do bazy danych.

Na diagramie **todo** przedstawiono współprace klas realizujące działanie nauki akordów. Okna oferujące wygląd aplikacji i posiadające ich działanie to klasy BasicLearningActivity (lista grup akordów), ListOfChordsActivity (spis akordów w grupie oraz ShowChordActivity (okno z danym akordem) wszystkie dziedziczą po klasie Activity i dołączane są do nich klasy pomocnicze takie jak NavigationTop (nawigacja górna aplikacji) czy NavigationBottom (nawigacja dolna aplikacji). Do poprawnego wyświetlania listy zostały wykorzystane klasy adapterów ChordsAdapter (lista akordów) oraz ChordsGroupAdapter (lista grup akordów), które dziedziczą odpowiednio z klas ArrayAdapter<Chords> oraz ArrayAdapter<ChordsGroup> oraz klasy pomocnicze Chords (zawiera nazwę akordu) i ChordsGroup (łączy nazwę grupy z listą akordów). Do okna ShowChordActivity dołączane są klasy pomocnicze Permission, Chord (tworzy powiązanie nazwy akordu z wymaganymi częstotliwościami, adresem URL do schematu akordu oraz liczbą prób zagrania akordu w trybie rankingu przez danego użytkownika) oraz AuthFirebase. Do połączenia się z bazą służą klasy ChordsCollection oraz LeaderboardCollecrtion. Pierwsza z nich odpowiada za działaniach na kolekcji akordów. Posiada ona informacje o podziale na grupy, wymagane częstotliwości dla podanego akordu oraz jego schemat. Druga zaś działa na kolekcji leaderboard, która zbiera informację o punktach danego użytkownika w grupie akordów, danym akordzie oraz podsumowaniu wszystkich grup.

### Ranking

Do okna z rankingiem dostęp mają tylko zalogowani użytkownicy aplikacji. Gdy użytkownik wejdzie w okno, które wyświetla ranking to zostaje wysłane zapytanie do bazy o spis użytkowników i ich punkty. Gdy baza wyśle takie dane aplikacja łączy użytkownika z danymi punktami, sortuje je ze względu na punkty oraz wyświetla je w formie listy. Sortowanie danych odbywa się za każdym razem gdy użytkownik zmieni grupę akordów lub gdy wybierze kategorie wszystkie , w którym punkty są sumowane ze wszystkich grup dla danego użytkownika.

Na diagramie **todo** przedstawiono klasy odpowiedzialne za funkcjonowanie rankingu. Za pomocą klasy LeaderboardActivity powiązany zostaje wygląd okna z jego działaniem rozszerzając klasę Activity. Do LeaderboardActivity dołączane są klasy pomocnicze takie jak NavigationTop, NavigationBottom odpowiedzialne za nawigacje górną oraz dolną w aplikacji oraz NavigationLeaderboard odpowiedzialną za przełączanie się pomiędzy kategoriami rankingu. Do poprawnego wyświetlania listy został zastosowany adapter, dziedziczący po ArrayAdapter, LeaderboardAdapter. Do połączenia z bazą danych użyta została klasa LeaderboardCollection oraz klasy pomocnicze takie jak User oraz Leaderboard. Ostatnia zawiera informacje o UID użytkownika, jego punktach w poszczególnych kategoriach oraz obiekt typu User informujący o jego nazwie i ikonie.

## Kompozycja struktury folderów

Po włączeniu programu za pomocą narzędzia Android Studio po lewej stronie ujrzymy taką strukturę folderów. Jest to wygląd folderów stworzonych przez IDE pod Androida. W przypadku gdy ustawimy, że chcemy zobaczyć faktyczny wygląd struktury folderów wygląd ten nieco się zmieni. Przykładem tego może być folder z układem elementów na ekranie (folder layout). W rzeczywistości jest tam folder o nazwie „layouts” a w nim poszczególne foldery, które odpowiadają za poszczególne elementy – (items – wygląd elementów dołączanych do poszczególnych aktywności, chords – wszystkie układy elementów dla czynności związanych z poprawnością akordów, profile – wszystkie wkłady aktywności dla okien dotyczących logowania, rejestracji i zarządzania kontem oraz activity – definiuje wygląd pozostałych aktywności).

Folder „java” przetrzymuje wszystkie pliki Javy odpowiedzialne za poprawne działanie aplikacji (funkcje oraz wprowadzające odpowiednie zmiany w wyglądzie okien). Starałam się aby struktura w tym folderze była spójna – tak więc widoczne są foldery, które przechowują wszystkie potrzebne pliki dla poszczególnych okien (leaderboard – rankingi, learnig – wszystkie elementy potrzebne do utworzenia nauki akordów, metronome – pliki potrzebne w okien metronomu, profile – profil użytkownika oraz logowanie i rejestracja, tuner- pliki odpowiedzialne za działanie stroika oraz folder utility w którym znajdują się pliki potrzebne w kilku miejscach, albo nie niosące sobą funkcjonalności jednak są potrzebne w poprawnym działaniu aplikacji). Folder collections zawiera pod foldery chords, leaderboard oraz user, a w nich są zawarte pliki Javy potrzebne w komunikacji aplikacji z bazą danych Firebase.



## Implementacja stroika

Stroik jest dostępny zarówno dla użytkowników zalogowanych jak i dla niezalogowanych. Wygląd tego okna jest determinowany za pomocą pliku XML o nazwie activity\_tuner. Do niego dołączany jest plik XML z nawigacją dolną aplikacji (plik bottom\_nav).

Stroik powinien pozwolić na nastrojenie gitary przy wybranym strojeniu oraz pomóc w nastrojeniu jednej struny jak i wszystkich jednocześnie. Przy czym podczas strojenia wszystkich strun aplikacja powinna pozwolić użytkownikowi na strojenie gitary bez konieczności wykonywania czynności na telefonie.

Używając biblioteki TarsosDSP możemy uzyskać informacje o częstotliwości dźwięku, który dotarł do mikrofonu urządzenia. Z jego pomocą mona utworzyć funkcjonalny stroik gitarowy. Funkcja która na to pozwala to znajdująca się w klasie PitchDetectionResult getPitch, która określa częstotliwość za pomocą algorytmu **FFT** (ang. Fast Fourier Transform). Funkcja ta jest uruchamiana na osobnym wątku aby nie blokowała ona działania aplikacji.

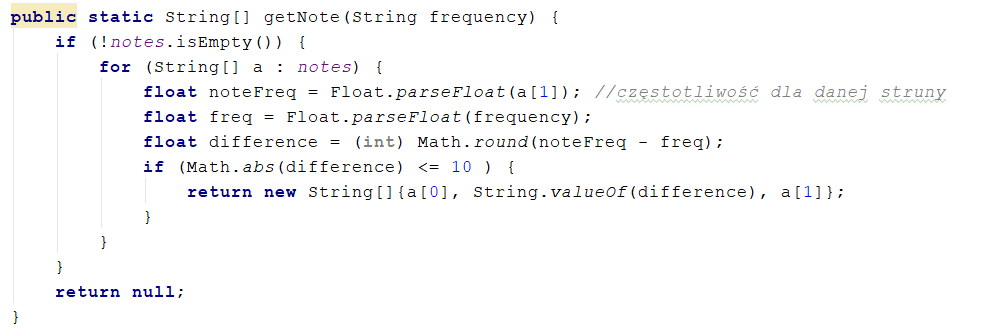
Zaczynając od strojenia jednej struny aplikacja czeka aż użytkownik określi, którą strunę chce nastroić. Gdy już to uczyni aplikacja sprawdza które strojenie zostało wybrane. W każdej chwili użytkownik może zmienić wybrany strój jak i strunę, którą chce nastroić. Po wyborze struny sprawdzany zostaje element w liście strun z odpowiednim ID przypisanym przy tworzeniu ułożenia strun podczas tworzenia wyglądu (przy pierwszym uruchomieniu okna lub po zmianie strojenia). Gdy struna zostanie znaleziona w liście pobrana zostaje wymagana częstotliwość i zapisana do zmiennej pomocniczej. Gdy mamy już wybraną strunę, więc i wymaganą częstotliwość, można przejść do kolejnego punktu jakim jest utworzenie nowego wątku i sprawdzenie częstotliwości dla dźwięku pobieranego przez mikrofon urządzenia.



Otwieramy wątek w metodzie getPitch dostępną w klasie TunerActivity. W tej metodzie nadpisujemy funkcje dostępną w bibliotece TarsosDSP, która będzie przekazywać na bieżąco wyniki działania algorytmu FFT. Kolejnym krokiem jest sprawdzenie czy odebrana częstotliwość jest zgodna z wymaganą częstotliwością.

Działanie aplikacji stojąc wszystkie struny jednocześnie ustawiamy przełącznik „Auto” w pozycji włączonej. Jeśli wcześniej został uruchomiony wątek strojenia gitary po jednej strunie to zostanie on wyłączony. Do momentu uruchomienia nowego wątku działanie aplikacji jest takie samo z wyjątkiem wybierania strun. W tym przypadku nie wybieramy strun a włączamy automatyczne dobieranie strun poprzez włączenie przełącznika „Auto”. Wątek po odczytaniu częstotliwości sprawdza której najbliżej znajduj się dźwięk. Każda ze strun na przypisana częstotliwość dzięki czemu aplikacja jest w stanie określić, do której ze częstotliwości jest najbardziej przybliżona częstotliwość odczytana przez mikrofon. Następnie aplikacja wskazuje która struna, według częstotliwości, jest grana i wskazuje jak bardzo ta częstotliwość jest inna do wymaganej (to samo dzieje się dla działania aplikacji w trybie strojenia jednej struny).

Sprawdzenie czy dany dźwięk jest najbardziej zbliżony do którejś ze częstotliwości przypisanych do strun odbywa się poprzez metodę dostępną w klasie Notes nazwaną „getNote”.



Na początku sprawdzone zostaje czy w tablicy strun zostały wczytane struny (podczas zmiany strojenia). Następnie zostają utworzone zmienne dla każdej ze strun w tablicy, każda ma typ float i są kolejno odpowiedzialne za częstotliwość aktualnie sprawdzanej struny, sprawdzana częstotliwość oraz różnica tych dwóch zmiennych. Ta różnica zostaje zaokrąglona, a później zostaje pozbawiona znaku. Różnica ta jest sprawdzana pod kątem wielkości i tak jeśli różnica będzie większa niż 20 to zostanie zbadana kolejna struna, a w pozostałych przypadkach zostanie zwrócona tablica z nazwą struny, różnicą oraz poprawną częstotliwością dla danej struny.

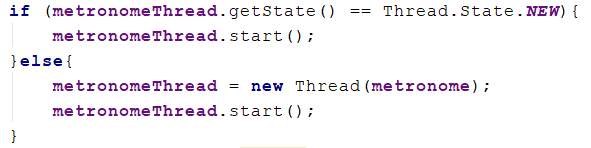
## Implementacja metronomu

Metronom jest dostępny dla użytkowników zalogowanych oraz niezalogowanych. Wygląd okna jest determinowany przez plik XML o nazwie activity\_metronome. Do niego dołączany jest plik XML z nawigacją dolną aplikacji (plik bottom\_nav).

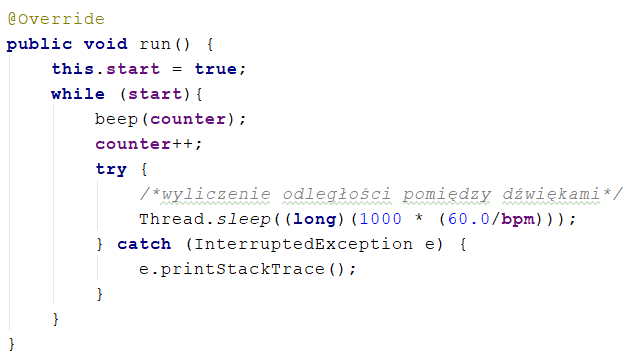
Metronom powinien pozwalać na uruchomienie metronomu oraz pozwalać na zmianę tempa oraz metrum. Uruchomienie metronomu powinno odbywać się na osobnym wątku w aplikacji aby nie zatrzymać działania całej aplikacji. Ponadto dostępne powinny być dwa różne sygnały dźwiękowe.

Uruchomienie działania metronomu odbywa się poprzez kliknięcie w przycisk START. Uruchamia on wątek, który sprawdzając aktualny stan ustawień odpowiednio ustawia odległość kolejnego dźwięku. W ustawieniach możemy określić metro oraz tempo w jakim ma działać metronom. Do wyboru są przygotowane osiem możliwych temp oraz jedenaście **metrów**. Jeśli użytkownik po kliknięciu przycisku START zmieni któryś z tych ustawień metronom zmieni odległość dźwięków i dostosuje ich rodzaj. Działanie wątku zakończy się tylko wtedy gdy użytkownik kliknie przycisk STOP lub przejdzie do innego okna.

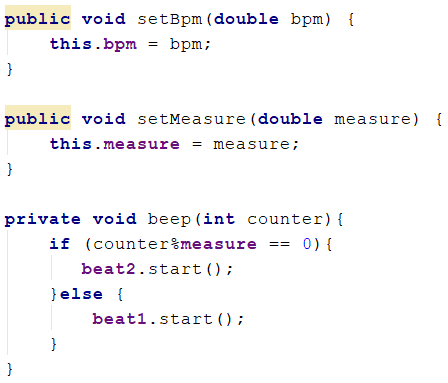
Aby uniknąć błędów spowodowanych przez brak odpowiednich danych jak i też przepełnieniem pamięci wykorzystany został stan wątku.



Działanie nieprzerwane metronomu jest uzyskanie poprzez budowę odpowiedniej klasy implementujący interfejs Runnable i nadpisanie metody run.



Klasa ta pozwala na zmianę tempa oraz metrum w dowolnej chwili co powoduje zmianę odległości między dźwiękami oraz wyboru dźwięku, który w danym momencie ma zostać odtworzony

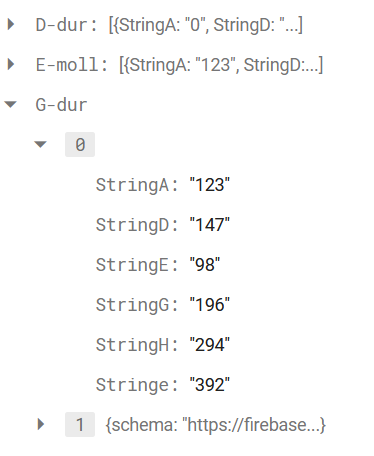


## Tworzenie bazy i uzupełnienie bazy danych z poziomu aplikacji

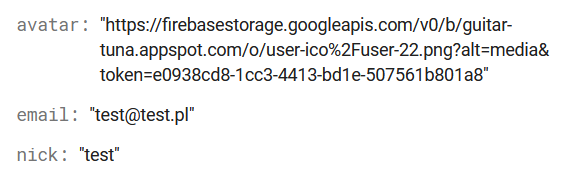
Baza danych, jak i inne działania na serwerze, jest umieszczona na platformie Firebase od Google i jest ona typu NoSQL.

/\*zalety baz typu NoSQL\*/

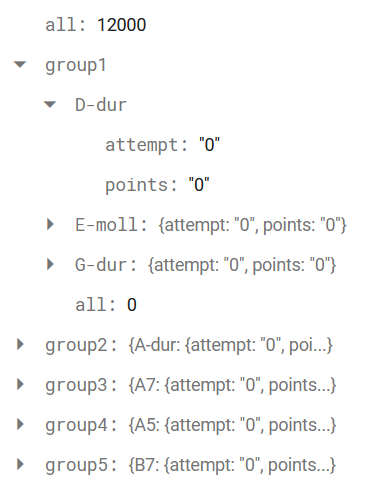
Na platformie Firebase przyjęto model dokumentów. Każdy dokument ma swój unikalny klucz i jest przechowywany w zbiorze nazywanym kolekcjami. W tym przypadku są dostępne 3 kolekcje. Pierwsza z nich to kolekcja chords(zdjęcie todo). W niej znajdują się aktualnie 5 dokumentów nazwanych grupami akordów. W każdej z grup znajdują się minimum 3 akordy. Ich klucze odpowiadają nazwą akordów i zawierają tablice. Pierwszy element z tablicy zawiera listę zbudowaną jako klucz-wartość i zawiera nazwę struny oraz wymaganą częstotliwość. Drugim elementem w tablicy jest schemat akordu, a dokładnie adres URL wskazujący na zasób umieszczony w Firebase Storage.



Kolekcja chords jest wykorzystywana do funkcjonalności aplikacji zajmujących się nauka akordów zarówno w trybie nauki jak i w trybie rankingu.



Kolejną kolekcją jest kolekcja users(zdjęcie todo), która jako klucz przyjmują wygenerowane podczas rejestracji UID. Wewnątrz dokumentu znajdują się informacje o adresie email, nazwie użytkownika oraz adres URL ikony użytkownika. Ta kolekcja jest wykorzystywana do wyświetlania i edycji danych w oknie profilu oraz, współpracując z kolekcją leaderboard, do poprawnego wyświetlania danych na temat rankingu użytkowników.



Zawartość kolekcji leaderboard(zdjęcie todo) służy do pobrania takich informacji jak liczba prób w trybie ranking dla danego akordu, ogólna liczba punktów uzyskana podczas grania danego akordu, w danej grupie czy ogólna liczba punktów uzyskana we wszystkich akordach. Aby to ułatwić akordy są podzielone w taki sam sposób jak w kolekcji chords jednak z tym wyjątkiem, że zamiast zawierać informacje o wymaganych częstotliwościach czy adresie URL schematu zawierają tylko informacje o punktach i próbach. Do szybszego pobrania danych zastosowano jeszcze jedno pole dla każdej grupy oraz dla podsumowania które zawierają informacje odpowiednio o wszystkich punktach w grupie i liczbie punktów ogółem. Dla rozpoznania, dla którego gracza są to statystyki zastosowano jako klucz dokumentu UID użytkownika.



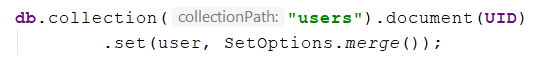
Połączenie z bazą oferują klasy ChordsCollection, LeaderboardCollection oraz UsersCollection. Aby można było dokonać jakikolwiek działań na bazie danych najpierw należy nawiązać połączeni poprzez użycie metody ujętej na zdjęciu todo, a następnie zapisać ją do zmiennej typu FirebaseFirestore.

Firebase korzysta z dokumentów, które oferują obiekty oparte na zasadzie klucz-wartość. Z tego też powodu podczas działań na bazie też powinno korzystać się z obiektów opartych o schemat klucz-wartość. Najlepszym wyborem zdaje się być przechowywania danych w mapach. Tak też są skonstruowane przetłumaczenie danych na wspomniane wcześniej mapy. Weźmy na przykład kawałek kodu odpowiedzialny za stworzenie nowego dokumentu w kolekcji users pod kluczem UID stworzonego właśnie użytkownika.

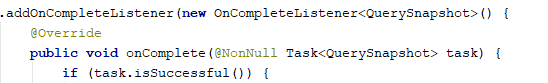


Na początku tworzymy obiekt mapy określając przechowywany typ klucza na string, a wartość na typ object (zdjęcie todo). Następnie do tego obiektu dodajemy kolejne obiekty. Na zdjęciu todo dodaje pod kluczem email wartość określająca adres email użytkownika.





Ostatnim krokiem (zdjęcie todo) do dodania nowego dokumentu jest na obiekcie bazy danych (w tym przypadku zmienna db) wywołać metodę określająca, w której kolekcji mają nastąpić zmiany oraz poprzez metodę document określamy w argumencie klucz dokumentu. Następnie wybieramy akcje, w tym przypadku wybrana zostaje metoda set i w parametrze określany zostaje obiekt user jako, że to właśnie w tym obiekcie są przechowane dane, które należy umieścić w bazie. W przypadku gdy chcemy zmodyfikować niektóre dane w danym dokumencie należy jako drugi parametr określić jako SetOptions.merge(). W takim przypadku, gdy nie będzie któregoś z kluczy, który jest w bazie to nie zostanie on nadpisany. Gdyby zabrakło drugiego parametru w metodzie set to wskazany dokument zostałby całkowicie zmieniony przez ten, który został stworzony w obiekcie user.





Do pobrania dokumentów służy metoda get. Jest ona bezparametrowa i zwraca całą kolekcje. Dlatego też, po sprawdzeniu czy dana kolekcja się pobrała (zdjęcie todo), wyszukiwany jest interesujący dokument przez sprawdzenie jego klucza (zdjęcie todo).

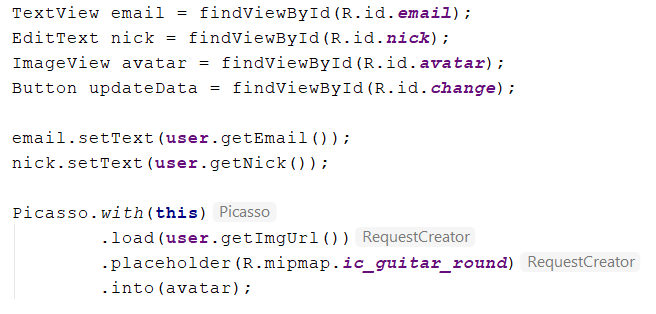
## Administracja użytkownikami

Zalogowany użytkownik zdobywa dostęp do nowych funkcjonalności. Aby się zalogować użytkownik musi podać adres email, którym się rejestrował oraz hasło pasujące do tego zapisanego w bazie danych. Okno logowania jest determinowany przez plik XML z nazwą activity\_login, a rejestracji przez plik activity\_registration. Rejestracja użytkownika przebiega w następujących krokach. Pierwszym z nich jest przejście przez użytkownika do okna rejestracji oraz wypełnieni formularza, na który składa się adres email oraz dwa pola z hasłem które muszą się zgadzać oraz posiadać minimum 6 znaków. W kodzie pierwszym krokiem, który dzieje się zaraz po uruchomieniu okna rejestracji jest nawiązanie połączenia z funkcją Firebase Authentication (tu zmienna firebaseAuth), która jest odpowiedzialna za autentyfikacje użytkowników oraz po stronie konsoli Firebase’a ułatwić zarządzanie nimi. Po kliknięciu w przycisk rejestracji na ekranie uruchamiana jest funkcja registerUser znajdująca się w klasie AuthFirebase. Po usunięciu białych znaków i sprawdzeniu poprawności danych przychodzi czas na wywołanie odpowiednich metod z klas Firebasa.

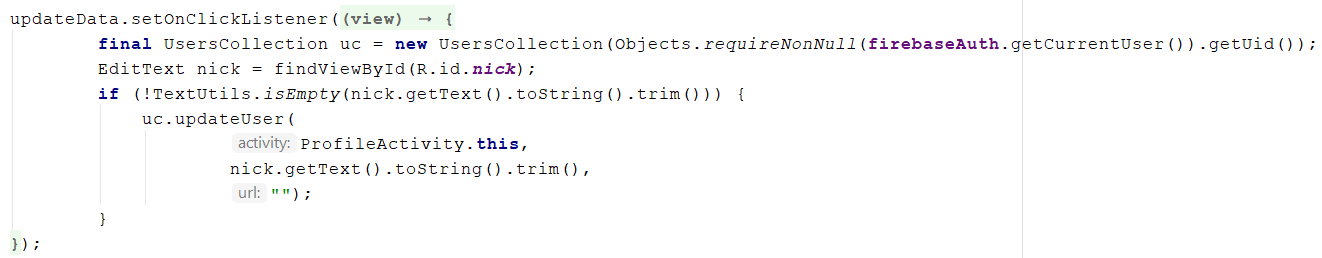


Metodą createUserWithEmailAndPassword i przekazując wpisane przez użytkownika dane otwieramy konto użytkownikowi. Po poprawnym utworzeniu konta (sprawdzone to zostaje poprzez sprawdzenie warunku task.isSuccessful) tworzone zostają dokumenty w kolekcji users oraz leaderboard. Następnie użytkownik poprzez metodę updateUI zostaje przekierowany do widoku swojego profilu. Jednak gdy z jakiegoś powodu nie można było stworzyć konta (np. podany adres email jest przypisany do innego konta) zostanie wyświetlona informacja o niepowodzeniu procesu autentyfikacji.

Wygląd okna profilu wykonywany jest za pomocą pliku XML activity\_profile. Za pomocą metody onCreate klasa ProfileActivity zostaje powiązana z wspomnianym wcześniej widokiem. Do poprawnego działania tego okna potrzebne jest uzyskanie informacji o użytkowniku. W tym celu zostaje uruchomiona funkcja readUser, która pobiera dane o użytkowniku z kolekcji users. Dane zawarte w tej kolekcji to adres URL odnoszący się ikony użytkownika, adres email przypisany do konta oraz nazwa użytkownika. Te dane zostają pobrane i wyświetlone za pomocą tej części kodu:

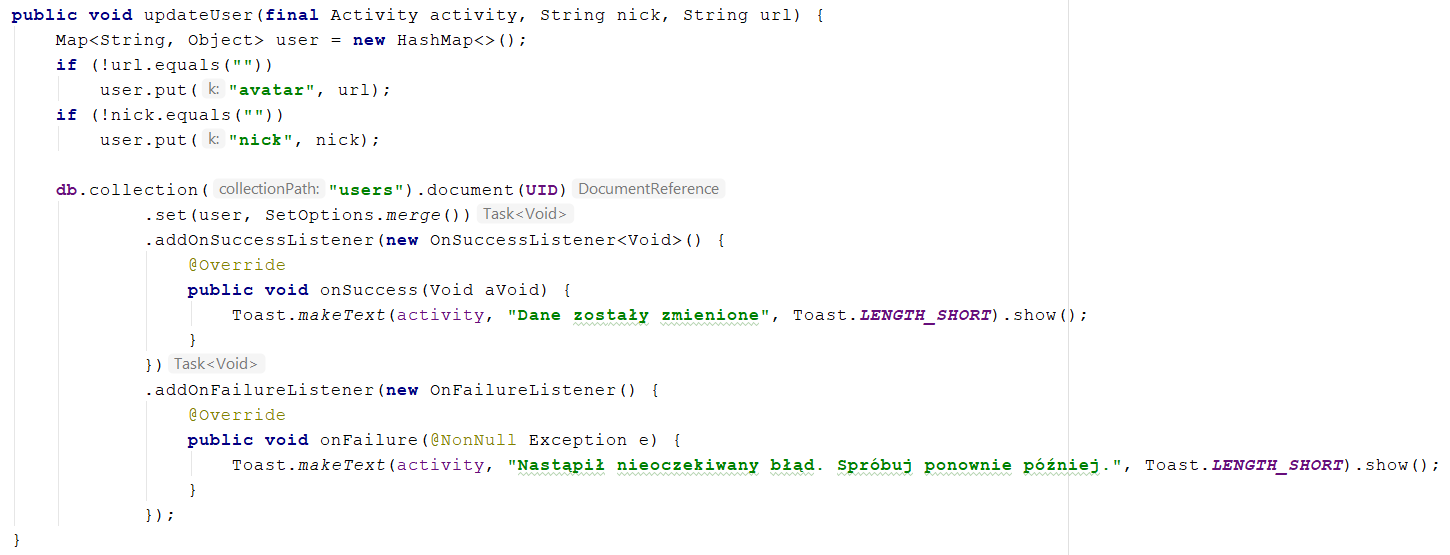


Do ikony użytkownika (zmienna avatar) oraz przycisku (zmienna updateData) zostają przypisane słuchacze. Są one uruchamiane przy naciśnięciu w dany obszar.

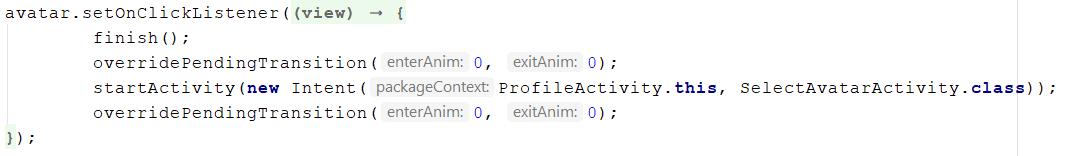


W przypadku przycisku zostaje pobrany tekst w polu nick (okno do wpisywania danych z klawiatury urządzenia) i przekazany zostaje jako parametr do metody updateUser, która zmieni zawartość danych w bazie danych w kolekcji users dla zalogowanego użytkownika.

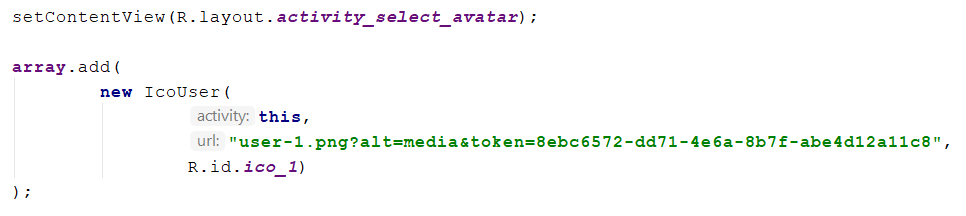
Zmienna updateUser przyjmuje 3 parametry. Pierwszy z nich to kontekst aplikacji, w tym przypadku to klasa ProfileActivity. Następnie przyjmuje dwa obiekty typu string. Pierwszy z nich odpowiada za nową nazwę użytkownika, a dugi za nowy adres URL ikony użytkownika.



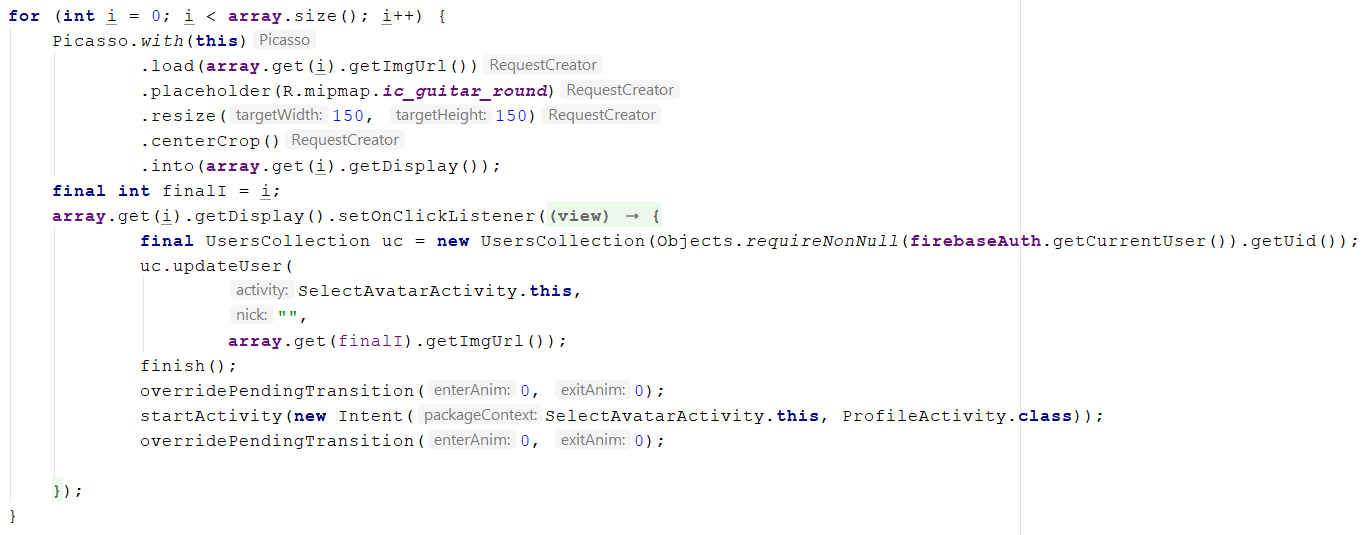
Po sprawdzeniu który z nich jest wartością pustą i odpowiednio dodaniu do wartości do obiektu zostaje uruchomiona metoda set (dla dokumentu w kolekcji users, który klucz ma taki jak UID zalogowanego użytkownika), która przyjmuje jako parametr nowy obiekt. Ważne jest aby w set jako drugi parametr ustawić SetOptions.merge() co sprawi że dokument nie nadpisze się, a zmienią się tylko te dane które mają inną wartość ustawiając to co, w tym przypadku, znajduje się w obiekcie user.



W przypadku chęci zmiany ikony użytkownika trzeba najpierw wywołać (poprzez kliknięcie w aktualną ikonę użytkownika) okno z dostępnymi ikonami - SelectAvatarActivity. Dzieje się to za pomocą metody startActivity. Pierwszy parametr to aktualny kontekst a drugi to klasa, która ma zostać wywołana. Reszta odpowiada za animacje zmiany okna.



Po powiązaniu pliku JAVA z plikiem XML odpowiadającym za ułożenie elementów (activity\_select\_avatar) następuje wpisanie do listy obiektów IcoUser łączący miejsce wyświetlania i adres URL obrazka.

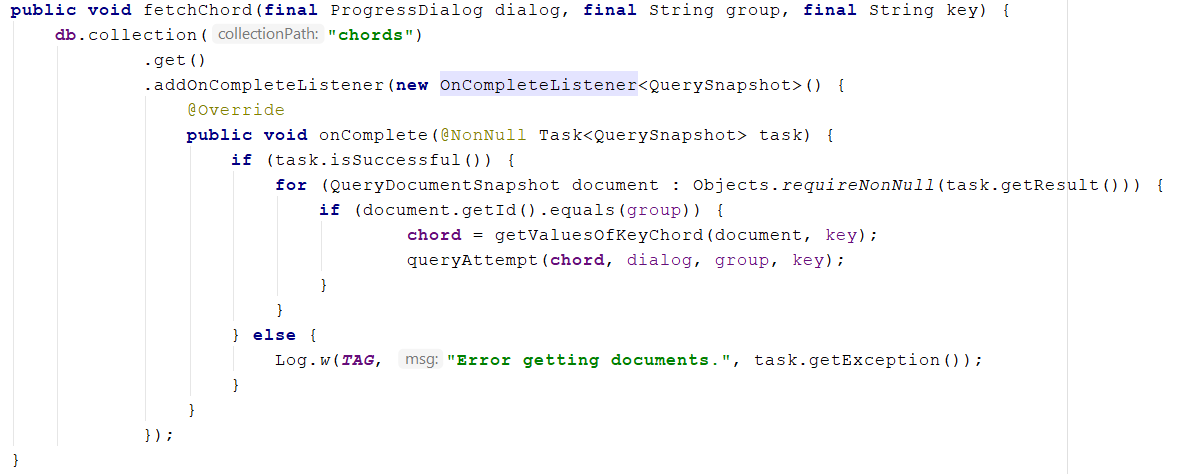


Następnie do każdego obiektu zostaje wyświetlony obraz oraz dołączony słuchacz pozwalający na zmianę adres URL ikony dla zalogowanego użytkownika. W tym celu znowu została zastosowana metoda updateUser, tylko jako nowa nazwa użytkownika został przekazany pusty string, a dla nowego adresu został wyciągnięty adres z klikniętego obiektu ikony i przekazany do metody jako parametr URL. Następnie zostało zastosowane zmiana okna i jego animacja.

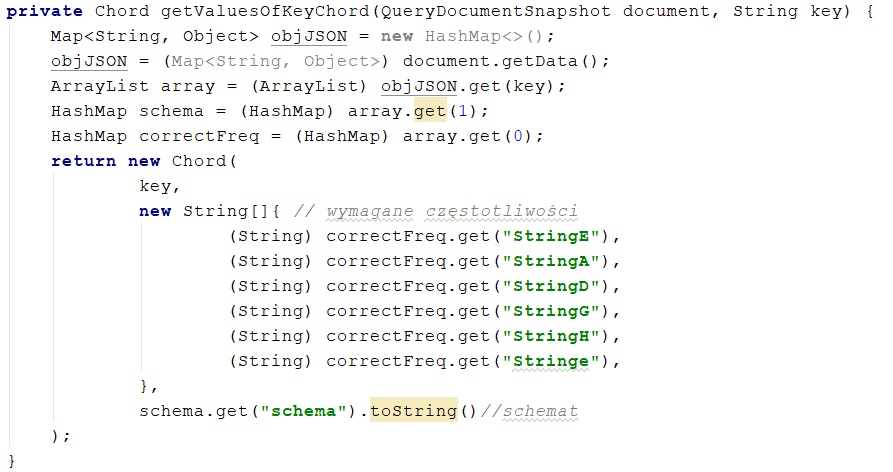
## Sprawdzenie poprawności akordów

Funkcjonalność dotycząca nauki akordów jest dostępna tylko dla zalogowanych użytkowników, a dla niezalogowanych jest możliwość tylko przeglądnięcia dostępnych akordów. Wygląd dla okien spełniających tę funkcjonalność to pliki XML o nazwie activity\_basic\_learning, activity\_show\_chord, item\_chords\_group (adapter dla obiektu ListView, odpowiada za poprawne wyświetlenie listy grupy akordów), item\_chords (adapter dla obiektu ListView, odpowiada za poprawne wyświetlenie listy akordów w grupie). Do niego dołączany jest plik XML z nawigacją dolną aplikacji (plik bottom\_nav) oraz z nawigacją górną aplikacji (plik top\_nav).

Sprawdzenie akordów odbywa się poprzez sprawdzenie częstotliwości dla poszczególnych strun i porównanie z pobranych wymaganych częstotliwościach z bazy danych. Dane zostają pobrane dopiero w chwili wyboru, przez użytkownika, danego akordu.



Dane o akordzie pobierane zostają z kolekcji chords przy pomocy powyższej funkcji, a takie dane jak informacja o ilości prób lub punkty w kategoriach akord, grupa akordów lub podsumowanie pobierane są z kolekcji leaderboard (na zdjęciu metoda queryAttempt). Nadpisanie metody addOnCompileListener daje nam dostęp do danych w momencie powodzenia pobrania kolekcji. Jest to bardzo pomocne ponieważ już teraz kod jest zabezpieczony przed działaniem na pustych obiektach. Następnie przechodzi po kolekcjach i wybiera ten dokument którego kluczem jest odpowiednia nazwa grupy akordu. Kolejno są wywołane funkcje odpowiedzialne za tworzenie akordu z wymaganymi częstotliwościami i schematem danego akordu oraz funkcja odpowiedzialna za dane o postępach użytkownika.



Obie funkcje są dość podobne do siebie. Dokumenty są dzielone na mniejsze obiekty i rzutowane zostają na mapy typu HashMap dzięki czemu łatwiej można pobrać dane. Wystarczy wywołać metodę get a jako parametr podać klucz, który w tym przypadku jest też kluczem w dokumencie.

Po pobraniu danych o akordzie (i wybraniu wcześniejszym trybu – w tym przypadku jest to tryb nauka) zostaje uruchomiony wątek który sprawdza poprawność dźwięku na strunie. Jeśli dźwięk jest poprawny to odpowiednia struna zostaje zaznaczona na kolor zielony. W kolejnej iteracji pętli ta struna zostaje pominięta w sprawdzaniu. Jeśli wszystkie struny zostają oznaczone jako prawidłowe pętla kończy się i użytkownik zostaje poinformowany o poprawnym zagraniu akordu.

W przypadku trybu ranking następują trzy zmiany. Otóż pierwsza rzucająca się zmiana to zniknięcie schematu i pojawienie się przycisku „pokaż schemat”. Pozwala ona na określenie punktów w dalszej części działania programu. Gdy schemat zostaje wyświetlony poprzez kliknięcie w przycisk „Pokaż schemat” zostaje on włączy przez resztę działania rundy (składa się z 4 prób) nie zostanie naliczone 100 punktów. Kolejną rzucająca się w oczy zmianą jest pojawienie się licznika czasu. Jest on bardzo ważny w procesie przyznawania punktów. Trzecim składnikiem w przyznawaniu punktów jest ogólna liczba podejść przez użytkownika. Wyróżnione są 3 etapy. Jeśli użytkownik podchodzi do danego akordu pierwszy, drugi lub trzeci raz to liczba punktów nadana w próbie jest mnożona odpowiednio 5, 4, 3 a w pozostałych przypadkach mnożnik punktów nie występuje. Poniżej znajduje się tabela z punktami.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| czas [s] | <8 | 8-9 | 10-19 | 20-29 | 30-39 | 40-49 | 50-59 | 60-120 |
| Liczba punktów | 200 | 150 | 140 | 130 | 120 | 110 | 100 | 50 |

Kolejną zmiana w trybie rankingu jest okienko, które pojawia się po zagraniu wszystkich strun poprawnie. Otóż użytkownik zostaje poinformowany o numerze próby w danej rudzie oraz o punktach zdobytych podczas trwania próby i podsumowanie punktów zdobytych w rundzie.

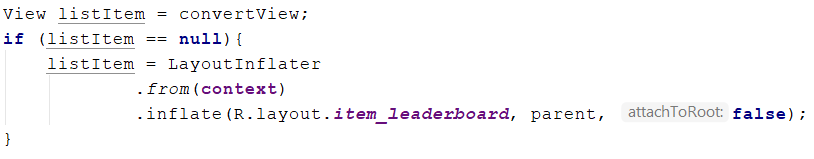
## Wyświetlenie rankingu z podziałem na poszczególne grupy

Funkcjonalność dotycząca rankingu jest dostępna tylko dla zalogowanych użytkowników. Wygląd dla okien spełniających funkcje wyświetlania listy rankingowej to pliki XML o nazwie activity\_leaderboard oraz item\_leaderboard (adapter dla obiektu ListView, odpowiada za poprawne wyświetlenie listy rankingowej). Do niego dołączany jest plik XML z nawigacją dolną aplikacji (plik bottom\_nav), nawigacją górną aplikacji (plik top\_nav) oraz z nawigacją rankingu czyli wybór odpowiedniej kategorii rankingu (plik leaderboard\_nav).

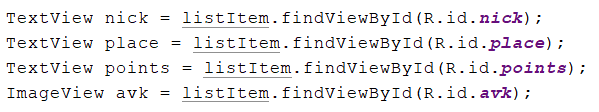
Wyświetlanie listy rankingowej odbywa się poprzez połączenia z danymi otrzymanych z bazy danych i poprzez adapter wyświetlona zostaje lista posortowanych użytkowników ze względu na liczbę zdobytych punktów.

Po kliknięciu w przycisk uruchamiający okno z rankingiem otwierane zostaje okno ładowania. Które zamyka się dopiero gdy uzyskamy kompletne dane z dwóch kolekcji – „leaderboard” oraz „users”. Gdy dane zostają pobrane i zwrócona zostaje lista danych o użytkownikach i ich statystykach następuje wyświetlanie tej listy przez adapter. Dzieje się to za pomocą klasy LeaderboardAdapter (rozszerza ArrayAdapter), który w konstruktorze przyjmuje okno aktualnie wyświetlonej aktywności, listę z danymi oraz nazwę grupy której dane mają zostać wyświetlone. W przypadku podania pustego string lub wartości null wyświetlone dane będą dotyczyć danych podsumowujących. Aby wyświetlić dane w widoku dodanym do XML odpowiedzialnego za wygląd okna należy nadpisać metodę getView (dostępna z ArrayAdapter), która zwraca widok elementu listy.

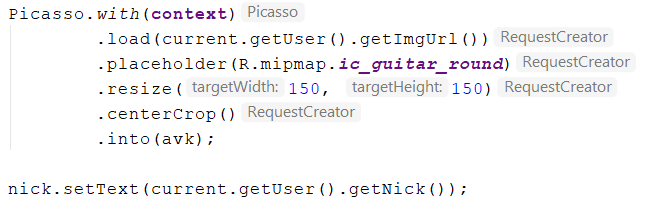
Zdecydowałam się na napisanie własnego adaptera ze względu na możliwość tworzenia własnego wyglądu listy. Jest tu na tyle ważne, ponieważ nazwy użytkowników mogą się powtórzyć, a żaden z dostępnych domyślnie adapterów listy nie był wstanie wyświetlić tylu informacji (avatar i nick użytkownika, numer pozycji w rankingu oraz punkty zdobyte przez użytkownika w danej kategorii) w przyjemnej i czytelnej formie. Dlatego więc w pliku item\_leaderboard.xml zapisany został wygląd pojedynczego elementu listy.



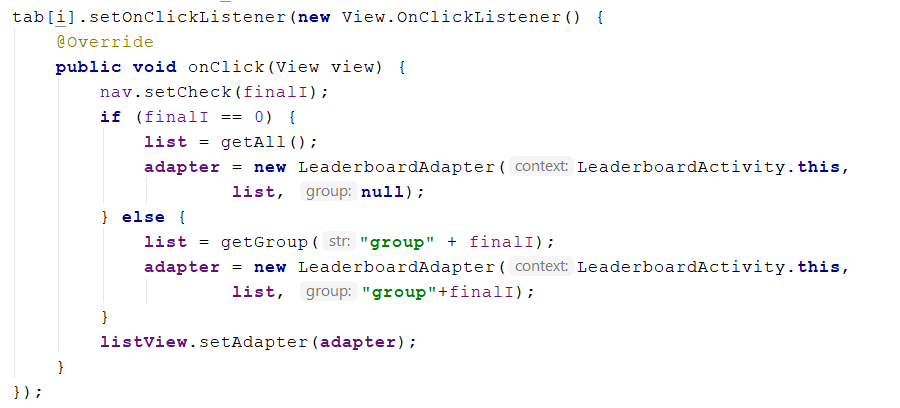
W pierwszym kroku należy podpisać pod zmienną aktualny widok listy. Jeśli wcześniej nie został wybrany należy podpisać wynikową wartość z LayoutInfleter, w tym miejscu wskazujemy widok dzięki któremu otrzymamy listę z własnym widokiem.



Kolejnym krokiem jest podpisanie pól w z pliku XML do zmiennych w pliku Java. Możemy tego dokonać ponieważ pod zmienną listIltem mamy aktualnie widok z pliku XML. Będzie tą zmienną traktować jako kontekst aplikacji co umożliwi wybór pola które chcemy podpisać do zmiennej.



Teraz gdy pod zmiennymi mamy obiekt z pliku XML można przejść do podpisywania danych z listy otrzymanej w konstruktorze klasy LeaderboardAdapter. Do obiektów typu TextView używa się metody setText, która w parametrze przyjmuje tekst jaki ustawi na ekranie. W przypadku obiektu ImageView wykorzystana została biblioteka Picasso, która w łatwy sposób ustawia obraz ze ścieżki url przy czym przed załadowaniem obrazka pokazuje się wcześniej ustalony obrazek. W tym przypadku to obraz, który jest ustalony jako ikona aplikacji (ic\_guitar\_round). Po ustaleniu co ma się wyświetlać w którym miejscu zwrócony zostaje obiekt listItem.



Aby wywołać taki adapter został zastosowany słuchacz akcji kliknięcia na nawigacje rankingu. Po kliknięciu w którykolwiek element obiekt adaptera zostaje nadpisany aby mogą zostać wyświetlona lista rankingowa dla odpowiednej kategorii. Do sortowania wyników zastosowany został algorytm bąbelkowy ukryty w metodach getAll oraz getGroup w zależności czy wyświetlane ma być dane o ogólnych punktach czy w danej grupie określanej przez parametr do metody getGroup. Gdy adapter zostaje już nadpisany do listy która wyświetla już listę na ekranie dołączany zostaje nowy adapter aby dane zostały odświeżone.

# Prezentacja systemu

## Obsługa aplikacji przez użytkownika niezalogowanego

## Obsługa aplikacji przez użytkownika zalogowanego

# Podsumowanie

Celem pracy było stworzenie aplikacji, która w znacznym stopniu ułatwi naukę gry na gitarze. Dzięki ocenie użytkowników można stwierdzić że cel ten został osiągnięty. Aplikacja została przetestowana na telefonach: FLOW 5 Kruger&Matz, Huawei P9 Lite 2018, TODO oraz na emulacji takich telefonów jak: TODO, przy czym funkcje bazujące na dźwiękach z otoczenia nie została sprawdzona. Wynika to z tego, że emulacja telefonu zbiera dźwięk zewnętrzny z wygenerowanych sygnałów komputera. Większość użytkowników stwierdziła, że aplikacja spełnia przedstawione wymagania oraz określiła przejrzystość aplikacji na bardzo dobry. Aplikacja została stworzona w taki sposób, aby w łatwy sposób dodać nowe funkcjonalności. W kwestii przyszłego rozwoju, na dzień dzisiejszy, widzę poprawę działania funkcji sprawdzania poprawności akordów. Chciałabym aby sprawdzanie odbywało się poprzez szarpnięcie wszystkich strun na raz oraz wskazanie, która ze strun jest zagrana nie prawidłowo. W tym momencie sprawdzanie akordów odbywa się poprzez zagranie struny po strunie. Kolejną możliwością rozwoju aplikacji jest poszerzenie bazy akordów oraz bazy dostępnych strojeń gitary. W przyszłości możliwe by było też ustawienie przez użytkownika własnych niestandardowych strojeń przez zmianę częstotliwości na wymaganych strunach.