

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ INFORMATYKI I
TELEKOMUNIKACJI

KIERUNEK: INFORMATYKA (TECHICZNA?)
SPECJALNOŚĆ: INŻYNIERIA SYSTEMÓW INFORMATYCZNYCH

PRACA DYPLOMOWA
INŻYNIERSKA

Projekt aplikacji webowej służącej do
muzycznego samokształcenia

Design of a web application for musical
self-education

AUTOR:

Jacek Kozicki

PROWADZĄCY PRACĘ:

dr inż. Tomasz Kubik, Katedra Informatyki Tech-
nicznej

Opracował: Tomasz Kubik <tomasz.kubik@pwr.edu.pl>
Data: maj 2021



Tekst zawarty w niniejszym szablonie jest udostępniany na licencji Creative Commons: *Uznanie autorstwa – Użycie niekomercyjne – Na tych samych warunkach, 3.0 Polska*, Wrocław 2021. Oznacza to, że wszystkie przekazane treści można kopiować i wykorzystywać do celów niekomercyjnych, a także tworzyć na ich podstawie utwory zależne pod warunkiem podania autora i nazwy licencjodawcy oraz udzielania na utwory zależne takiej samej licencji. Tekst licencji jest dostępny pod adresem: <http://creativecommons.org/licenses/by-nc-sa/3.0/pl/>. Podczas redakcji pracy dyplomowej stronę tę można usunąć. Licencja dotyczy bowiem zredagowanego opisu, a nie samego latexowego szablonu. Latexowy szablon można wykorzystywać bez wzmiankowania o jego autorze.

Streszczenie

Streszczenie w języku polskim powinno zmieścić się na połowie strony (drugą połowę powinien zająć abstract w języku angielskim).

Lorem ipsum dolor sit amet eleifend et, congue arcu. Morbi tellus sit amet, massa. Vivamus est id risus. Sed sit amet, libero. Aenean ac ipsum. Mauris vel lectus.

Nam id nulla a adipiscing tortor, dictum ut, lobortis urna. Donec non dui. Cras tempus orci ipsum, molestie quis, lacinia varius nunc, rhoncus purus, consectetur congue risus.

Słowa kluczowe: raz, dwa, trzy, cztery

Abstract

Streszczenie in Polish should fit on the half of the page (the other half should be covered by the abstract in English).

Lorem ipsum dolor sit amet eleifend et, congue arcu. Morbi tellus sit amet, massa. Vivamus est id risus. Sed sit amet, libero. Aenean ac ipsum. Mauris vel lectus.

Nam id nulla a adipiscing tortor, dictum ut, lobortis urna. Donec non dui. Cras tempus orci ipsum, molestie quis, lacinia varius nunc, rhoncus purus, consectetur congue risus.

Keywords: one, two, three, four

Spis treści

1. Wstęp	10
1.1. Wprowadzenie	10
1.2. Cel i zakres pracy	10
1.2.1. Cel pracy	10
1.2.2. Zakres pracy	11
1.3. Układ pracy	11
2. Użyte technologie	12
2.1. Technologie frontendowe	12
2.1.1. Wybór technologii	12
2.1.2. Opis wybranej technologii	14
2.2. technologie backendowe	14
2.2.1. Wybór technologii	14
2.2.2. Opis wybranej technologii	14
2.3. Wersjonowanie oraz pozostałe narzędzia	14
2.3.1. Wersjonowanie	14
2.3.2. Środowiska programistyczne	15
3. Analiza wymagań	16
3.1. Wymagania funkcjonalne	16
3.2. Wymagania niefunkcjonalne	16
3.3. Omówienie przypadków użycia	17
3.3.1. Diagram przypadków użycia	17
3.3.2. Scenariusze wybranych przypadków użycia	17
4. Projekt aplikacji	19
4.1. Architektura systemu	19
4.2. Architektura aplikacji klienta	20
5. Implementacja aplikacji	22
5.1. Implementacja Backendu	22
5.2. Implementacja Frontendu	22
5.2.1. Komunikacja z API	24
5.2.2. Podstrony z wirtualnymi instrumentami	25
5.2.3. Trening słuchu	26
5.2.4. Quiz	26
6. Podsumowanie	28
6.1. Sekcja poziomu 1	28
6.1.1. Sekcja poziomu 2	28
6.2. Sekcja poziomu 1	28
Literatura	29
A. Instrukcja wdrożeniowa	30

B. Opis załączonej płyty CD/DVD	31
--	-----------

Spis rysunków

2.1. Statystyki wykorzystania frameworka Angular na stronach generujących największy ruch[2]	13
2.2. Statystyki wykorzystania frameworka React na stronach generujących największy ruch[6]	13
3.1. Diagram przypadków użycia aplikacji	17
4.1. Architektura komunikacji w systemie opartym na REST API[7]	19
4.2. Diagram podstrony z wirtualnym instrumentem z wyróżnieniem komponentów i zależności między nimi	20
5.1. Schemat projektu aplikacji Frontendowej	23
5.2. Katalogi audio-player oraz guitar-page pokazują schemat plików odpowiednio serwisu i komponentu	24
5.3. Widok strony z gitarą basową przed wykonaniem zapytania	25
5.4. Widok strony z gitarą basową po wykonaniu zapytania	26
5.5. Widok strony z gitarą basową po wykonaniu zapytania i przełączeniu opcji hover	26
5.6. Widok strony z gitarą akustyczną po wykonaniu zapytania	27

Spis tabel

Spis listingów

5.1.	Klasa Scale wykorzystywana do zapytań i przechowywania danych o skalach . . .	24
5.2.	Metoda klasy TheoryApiService wysyłająca zapytanie o dźwięki akordu do API . .	24
5.3.	Wstrzykiwanie zależności do komponentu GuitarPageComponent	25

Skróty

OGC (ang. *Open Geospatial Consortium*)
XML (ang. *eXtensible Markup Language*)
SOAP (ang. *Simple Object Access Protocol*)
WSDL (ang. *Web Services Description Language*)
UDDI (ang. *Universal Description Discovery and Integration*)
GIS (ang. *Geographical Information System*)
SDI (ang. *Spatial Data Infrastructure*)
ISO (ang. *International Standards Organization*)
WMS (ang. *Web Map Service*)
WFS (ang. *Web Feature Service*)
WPS (ang. *Web Processing Service*)
GML (ang. *Geography Markup Language*)
SRG (ang. *Seeded Region Growing*)
SOA (ang. *Service Oriented Architecture*)
IT (ang. *Information Technology*)

Rozdział 1

Wstęp

1.1. Wprowadzenie

Wraz ze wzrostem dostępu do internetu, wzrosła również pula darmowych zasobów do nauki. Wg danych na rok 2021[10] ponad 66% ludności na świecie ma dostęp do internetu. W Polsce natomiast, szacuje się że jest to ponad 90% gospodarstw domowych[11]. Aktualne możliwości samorozwoju są więc bardzo duże, szczególnie w krajach takich jak Polska, gdzie dostęp do internetu jest powszechny.

Niestety, nie wszystkie materiały dostępne za darmo są wysokiej jakości, często trafiają się w nich błędy, są nieprzyjazne dla użytkownika lub niekompletne.

Aplikacja stworzona w ramach tej pracy również ma posłużyć jako darmowy zasób edukacyjny. Projekt skupiał się będzie w szczególności na gitarze basowej, ale docelowo, będzie łatwy do rozbudowania o inny rodzaj gitary oraz kolejne instrumenty, takie jak np. pianino. Niniejsza praca porusza więc dwa istotne zagadnienia: teorię muzyki oraz projektowanie aplikacji webowych.

Teoria muzyki jest często pomijana przez muzyków amatorów, którzy edukują się sami - oprócz tego, że same materiały mogą być słabej jakości, trudno jest zacząć przekładać teorię na praktykę. Na przeciw tym temu problemowi starają się wyjść aplikacje takie jak Rocksmith czy musicca, które są inspiracją dla niniejszej pracy. Rocksmith to gra pozwalająca na granie piosenek, do których tabulatura jest wyświetlana na ekranie w czasie rzeczywistym. Jako kontroler wykorzystuje się gitarę elektryczną lub basową ze specjalnym złączem. Oprócz tego, Rocksmith posiada również wiele przydatnych narzędzi, m.in. stroik czy efekty modyfikujące brzmienie gitary. Istotnymi wadami są natomiast: cena, wymaganie specjalnego złącza oraz duże obciążenie komputera. Aplikacja musicca jest dużą bliższą inspiracją - posiada interaktywny gryf i możliwość wyszukiwania akordów czy interwałów. W tej aplikacji za minus można uznać interfejs użytkownika - przejście z wyszukiwania interwałów do akordów wymaga zmiany widoku, a w kolejnym widoku trzeba ponownie wybrać instrument.

Aplikacja stworzona w ramach tej pracy to próba stworzenia przyjaznego użytkownikowi środowiska do nauki teorii muzyki, która wolna będzie od wyżej wymienionych problemów.

1.2. Cel i zakres pracy

1.2.1. Cel pracy

Celem pracy jest stworzenie aplikacji, która wspomaga muzyków w nauce teorii muzyki poprzez wizualizację oraz interaktywne ćwiczenia. Aplikacja powinna być łatwo rozszerzalna o kolejne aspekty teoretyczne oraz dodatkowe rodzaje ćwiczeń.

1.2.2. Zakres pracy

Zakres pracy obejmuje:

- wybór technologii,
- zaprojektowanie aplikacji webowej,
- implementację aplikacji,
- projekt i implementację testów aplikacji,
- udokumentowanie procesu w pracy dyplomowej.

1.3. Układ pracy

Opis zawartości kolejnych rozdziałów zostanie dodany kiedy rozdziały będą gotowe.

Rozdział 2

Użyte technologie

Na wybór technologii wpływ miały następujące czynniki:

- możliwość implementacji założonych funkcjonalności - technologia musi być dobrana odpowiednio do zadania,
- popularność technologii - wpływa na dostępność materiałów oraz pokazuje czy dana znajomość danej technologii ma szanse przydać się na rynku pracy,
- preferencja personalna - ostateczna decyzja zależała również od tego, na ile dobrze autor znał proponowane rozwiązania.

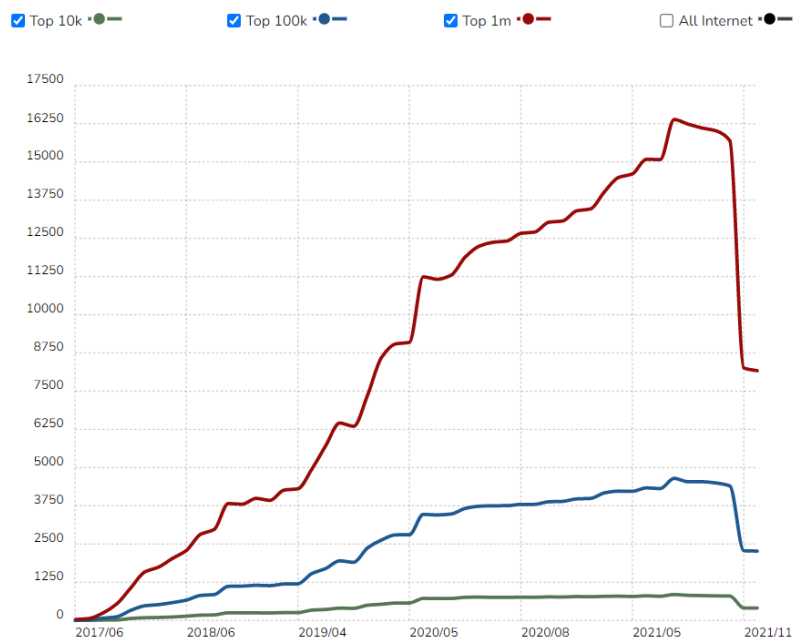
2.1. Technologie frontendowe

2.1.1. Wybór technologii

Podczas wyboru technologii frontendowej rozważane były przede wszystkim frameworki React oraz Angular.

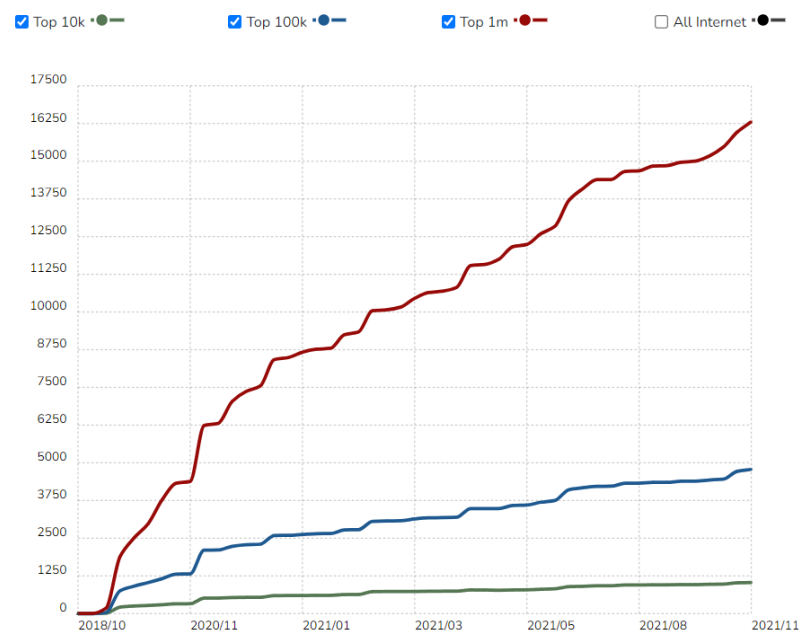
Wg trends.builtwith - strony analizującej trendy technologiczne w branży popularność obu frameworków jest zbliżona - na dzień 26.01.2022 295,780 stron internetowych zbudowanych jest z wykorzystaniem Angulara[2], a z wykorzystaniem Reacta 309,581[6]. Wykresy 2.1 oraz 2.2 przedstawiają popularność wspomnianych bibliotek w okresach 06.2017-10.2021 oraz 10.2018-10.2021 na podstawie liczby stron korzystających z danego rozwiązania spośród miliona, 100 tysięcy oraz 10 tysięcy stron generujących największy ruch.

Angular Usage Statistics



Rys. 2.1: Statystyki wykorzystania frameworka Angular na stronach generujących największy ruch[2]

React Redux Usage Statistics



Rys. 2.2: Statystyki wykorzystania frameworka React na stronach generujących największy ruch[6]

Na podstawie wykresów można łatwo zauważyć spadającą popularność angulara, należy jednak pamiętać, że decyzja o wyborze technologii, podejmowana była na początkowym etapie pracy, a więc tendencja spadkowa nie była wtedy jeszcze tak widoczna. Przed tak znaczącym spadkiem popularności Angulara obie technologie były wybierane z podobną częstotliwością, więc ostateczna decyzja sprowadziła się do preferencji autora - w ramach zajęć na uczelni oraz

prywatnych projektów miał on styczność z obydwoma bibliotekami, ale to z Angularem miał większe doświadczenie, w związku z czym wybrany został właśnie ten framework.

2.1.2. Opis wybranej technologii

Angular to kompleksowy framework do projektowania oraz tworzenia wydajnych aplikacji typu SPA (Single Page Application)[1]. Podstawowymi elementami składowymi Frameworka Angular są komponenty Angular, które są zorganizowane w NgModules. NgModules zbierają powiązany kod w zestawy funkcjonalne; aplikacja Angular jest definiowana przez zestaw NgModules. Aplikacja zawsze ma co najmniej moduł główny, który umożliwia ładowanie początkowe, i zazwyczaj ma o wiele więcej modułów funkcji.

Podczas implementacji korzystano głównie z komponentów i serwisów. Komponenty definiują widoki, które są zestawami elementów ekranu, które Angular może wybierać i modyfikować zgodnie z logiką programu i danymi. Serwis w Angularze jest zwykłą klasą TypeScript. Opatrzony jest dodatkowym dekoratorem, który może mieć różne właściwości konfiguracyjne. Serwis może działać jako singleton dla całej aplikacji lub może zostać powołany dla konkretnego komponentu.

2.2. technologie backendowe

2.2.1. Wybór technologii

Podczas wyboru technologii backendowej głównymi kandydatami były Python[5] z frameworkiem Flask[3] oraz Java[4] z frameworkiem Spring[9].

Tutaj głównym kryterium okazał się rozmiar aplikacji - w planowanej implementacji backend jest stosunkowo niewielki, większość implementacji leży po stronie frontendu, natomiast Flask jest lepiej przystosowany do tworzenia niewielkich aplikacji niż Spring.

2.2.2. Opis wybranej technologii

Flask określany jest mianem micro framework, co oznacza, że nie wymaga żadnych dodatkowych narzędzi ani bibliotek. Służy on do budowania aplikacji sieciowych i charakteryzuje się brakiem wbudowanych komponentów do obsługi bazy danych czy walidacji formularzy, ale zamiast tego jest on skonstruowany w taki sposób, aby był łatwo rozszerzalny o wszystkie potrzebne funkcjonalności.

Wśród zalet flaski wymieniane są przede wszystkim:

- pytoniczność kodu - kod pisany w tym frameworku pozwala na stosowanie dobrych praktyk programowania charakterystycznych dla języka Python,
- łatwość nauki - z uwagi na to, że kod bardzo przypomina ten pisany w czystym Pythonie a sam framework posiada niewiele komponentów, jest on uznawany za łatwy do przyswojenia.

2.3. Wersjonowanie oraz pozostałe narzędzia

2.3.1. Wersjonowanie

Do wersjonowania aplikacji wykorzystano system kontroli wersji GIT. Zdalne repozytorium zostało założone na platformie github. Podczas pracy z GITem korzystano przede wszystkim z konsoli git bash oraz interfejsów graficznych udostępnianych przez używane środowiska programistyczne.

2.3.2. Środowiska programistyczne

Do pracy z kodem pisanym w języku Python wykorzystano środowisko PyCharm - jest ono bardzo intuicyjne, wspiera formatowanie kodu według standardu PEP8, a oprócz tego posiada ono wbudowane wsparcie dla projektów opartych o framework Flask.

Do pracy z frameworkiem Angular wykorzystano edytor kodu Visual Studio Code. Edytor nie posiada wbudowanych narzędzi do pracy w tej technologii, ale jest otwarty na rozszerzenia, a z uwagi na swoją popularność, jest ich dużo i są dobrej jakości. W ramach projektu przydały się przede wszystkim rozszerzenia: SonarLint i Auto Rename Tag.

Rozdział 3

Analiza wymagań

3.1. Wymagania funkcjonalne

Dla implementowanej aplikacji zdefiniowano następujące wymagania funkcjonalne:

- Użytkownik może sprawdzić jakie dźwięki należą do wybranych skal,
- Użytkownik może sprawdzić jakie dźwięki tworzą dany interwał,
- Użytkownik może sprawdzić jakie dźwięki tworzą dany akord,
- Użytkownik może wchodzić w interakcję z wirtualnym gryfem gitary, aby odtworzyć dźwięki,
- Użytkownik może sprawdzić swoją wiedzę teoretyczną w interaktywnych ćwiczeniach,
- Użytkownik może trenować ucho muzyczne za pomocą specjalnych ćwiczeń.

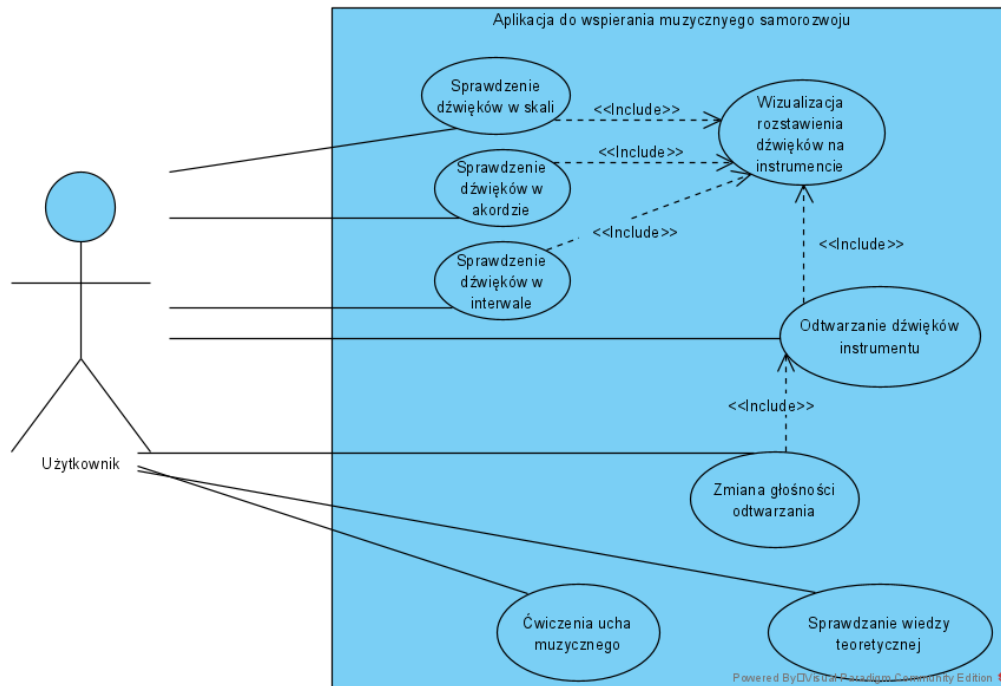
3.2. Wymagania нефunkcjonalne

Dla implementowanej aplikacji zdefiniowano następujące wymagania нефunkcjonalne:

- Aplikacja powinna działać na najpopularniejszych przeglądarkach internetowych,
- Aplikacja powinna być responsywna,
- Aplikacja nie wymaga od użytkownika zakładania konta,
- Pierwsze załadowanie aplikacji nie powinno trwać dłużej 2 sekundy,
- Aplikacja powinna być wykonana przy użyciu bibliotek Angular oraz Flask,
- Aplikacja powinna być zbudowana w taki sposób, aby jej dalszy rozwój był jak najprostszy,
- Aplikacja powinna posiadać intuicyjny interfejs użytkownika

3.3. Omówienie przypadków użycia

3.3.1. Diagram przypadków użycia



Rys. 3.1: Diagram przypadków użycia aplikacji

3.3.2. Scenariusze wybranych przypadków użycia

Poniżej opisano Scenariusze dla następujących przypadków użycia: ćwiczenie ucha muzycznego, sprawdzanie dźwięków w skali, odtwarzanie dźwięków instrumentu, zmiana głośności odtwarzania.

Ćwiczenie ucha muzycznego

1. Za pomocą paska nawigacji użytkownik wchodzi na podstronę z ćwiczeniem ucha muzycznego.
2. Użytkownik widzi swój aktualny wynik oraz guziki służące do obsługi ćwiczenia.
3. Po wciśnięciu przycisku, użytkownik słyszy wylosowany dźwięk. Guzik może być wciśnięty wielokrotnie w celu ponownego odsłuchania dźwięku.
4. Użytkownik wybiera odpowiedź z listy lub klika guzik reset.
 - 4.1. Po wybraniu poprawnej odpowiedzi losowany jest nowy dźwięk, a wynik zwiększa się o 1pkt.
 - 4.2. Po wybraniu złej odpowiedzi lub wciśnięciu guziku reset, losowany jest nowy dźwięk, a wynik ustawiany jest na 0pkt.

Sprawdzanie dźwięków w skali

1. Za pomocą paska nawigacji użytkownik wchodzi na podstronę z wybranym instrumentem.
2. Użytkownik widzi wirtualny instrument oraz selektor zapytań.
3. Użytkownik wybiera interesujące go zapytanie i zatwierdza to guzikiem.
4. Na wirtualnym instrumencie zaznaczają się odpowiednie dźwięki.

Odtwarzanie dźwięków instrumentu

1. Za pomocą paska nawigacji użytkownik wchodzi na podstronę z wybranym instrumentem.
2. Użytkownik widzi wirtualny instrument oraz selektor zapytań.
3. Użytkownik naciska na guzik odpowiadający wybranemu dźwiękowi na instrumencie.
4. Wybrany przez użytkownika dźwięk jest odtwarzany.

Zmiana głośności odtwarzania

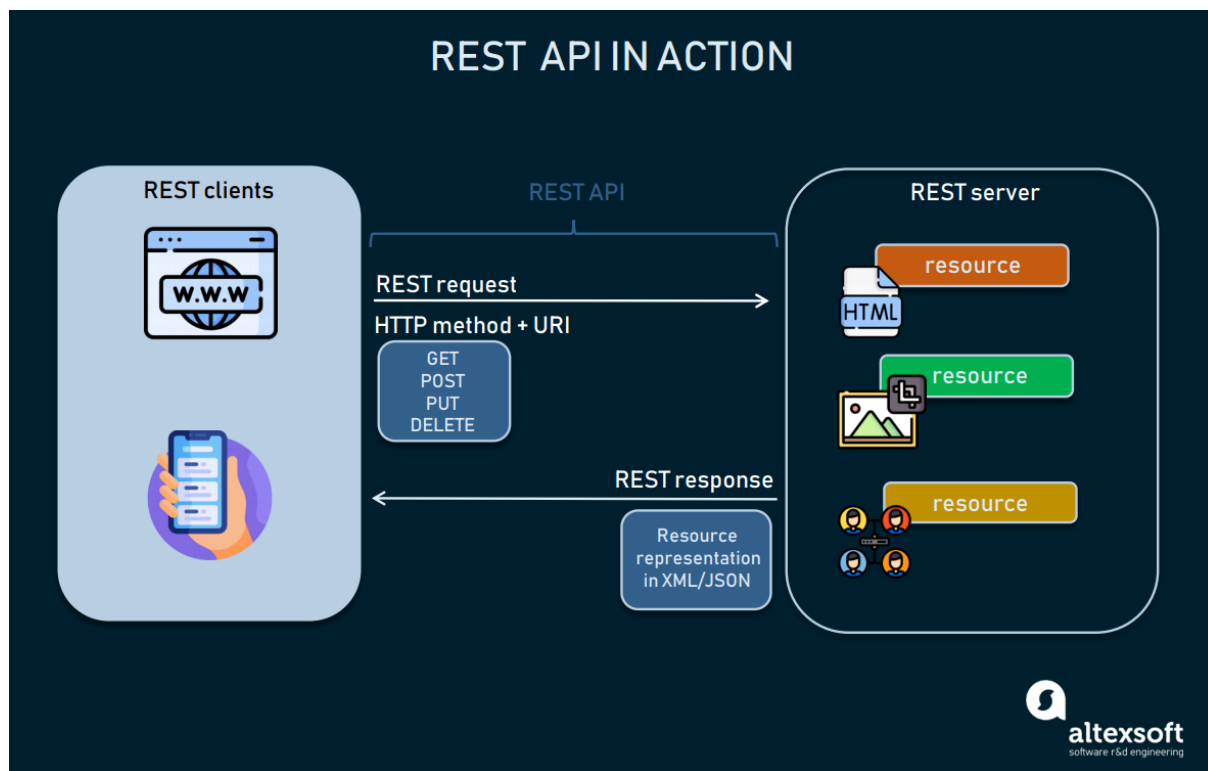
1. Za pomocą paska nawigacji użytkownik wchodzi na podstronę z wybranym instrumentem.
2. Użytkownik widzi wirtualny instrument oraz selektor zapytań.
3. Użytkownik odtwarza dowolny dźwięk poprzez kliknięcie na wirtualny instrument.
4. Użytkownik zmienia głośność instrumentu korzystając z suwaka regulacji głośności.
5. Użytkownik ponownie odtwarza dowolny dźwięk poprzez kliknięcie na wirtualny instrument - głośność jest dostosowana do jego oczekiwań.

Rozdział 4

Projekt aplikacji

4.1. Architektura systemu

Aplikacja bazuje na REST API - komunikacja pomiędzy aplikacją frontendową i backendową odbywa się w sposób przedstawiony na rysunku 4.1. Aplikacja kliencka wysyła zapytania HTTP na odpowiednie URI, a w odpowiedzi otrzymuje dane w formacie JSON. Typowy sposób tworze-



Rys. 4.1: Architektura komunikacji w systemie opartym na REST API[7]

nia endpointów w REST API to nazywanie endpointów nazwami kolekcji, a następnie uszczegóławianie zapytania w kolejnych częściach adresu[8]. Przykładowe endpointy w REST API powinny wyglądać następująco:

API/people - endpoint zwracający zawartość kolekcji “people”

API/people/:id - endpoint zwracający element kolekcji “people” o zadany id

API/people/:id/name - endpoint zwracający wartość pola “name” elementu kolekcji “people” o zadany id

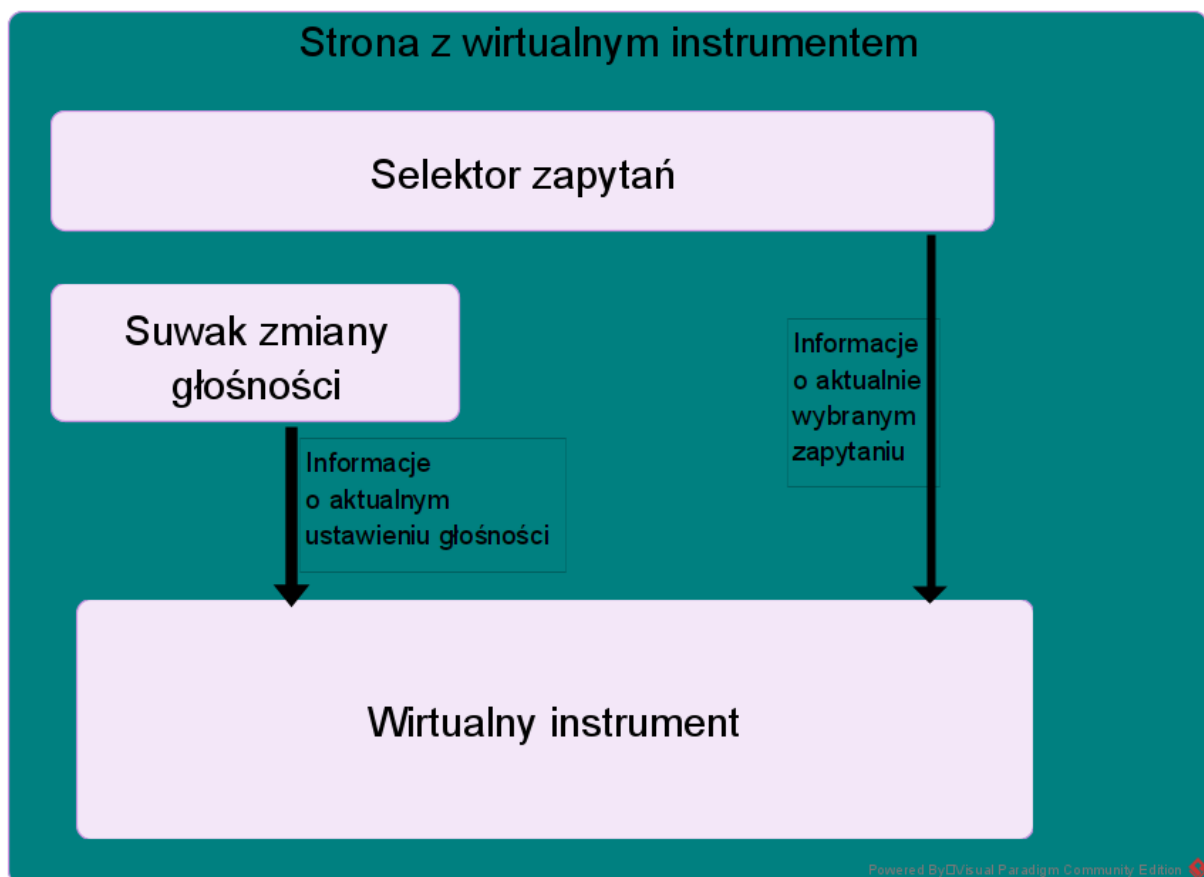
Rozwiązanie REST API pozwala na łatwe przesyłanie informacji o potrzebnych obiektach, takich jak skale czy akordy - JSON nie ogranicza wielkości wiadomości czy liczby elementów, więc takie obiekty świetnie nadają się do przekazywania tablic o nieznanych z góry rozmiarach. REST API pozwala również w prosty sposób zaimplementować mechanizmy obsługi błędów poprzez dołączanie do przekazywanego obiektu statusów HTTP, które opisują efekt zapytania czy nawet dodatkowej wiadomości opisującej błąd w dokładniejszy sposób.

Kolejną zaletą stosowania tego podejścia jest jego skalowalność. Każda funkcjonalność jest przypisana pod osobny adres, więc łatwo dodawać nowe poprzez tworzenie kolejnych endpointów.

4.2. Architektura aplikacji klienta

Korzystając z możliwości oferowanych przez framework Angular zdecydowano się rozbić aplikację na mniejsze elementy - charakterystyczne dla tej biblioteki komponenty oraz serwisy. Każda podstrona składa się z odpowiednich, połączonych ze sobą komponentów. Takie podejście pozwala na wielokrotne używanie tego samego kodu - ten sam komponent można wykorzystać w kilku miejscach na stronie lub na wielu różnych podstronach. Pozwala ono również na generowanie wielu podobnych podstron poprzez zamianę pojedynczych komponentów.

Diagram 4.2 przedstawia schemat jednej z podstron w aplikacji - podstrony zawierającej wirtu-



Rys. 4.2: Diagram podstrony z wirtualnym instrumentem z wyróżnieniem komponentów i zależności między nimi

alny instrument. Jak widać sam wirtualny instrument jest osobnym komponentem, co pozwala

na stworzenie części aplikacji dedykowanej dla gitary akustycznej oraz części aplikacji dedykowanej dla gitary basowej, a potencjalnie również dla dowolnych innych instrumentów. W tym celu należy jedynie zmienić komponent implementujący instrument.

Warto również zwrócić uwagę, na to, że oprócz widocznych na diagramie mniejszych komponentów, pojedynczy komponent może zawierać w sobie serwisy. Pozwalają one m.in. na pobieranie danych z API czy dostęp do zasobów lokalnych serwera.

Rozdział 5

Implementacja aplikacji

5.1. Implementacja Backendu

5.2. Implementacja Frontendu

Schemat projektu aplikacji frontendowej jest typowy dla aplikacji implementowanych w Angularze. Program podzielony jest na fragmenty zwane komponentami oraz serwisami. Każdy komponent oraz każdy serwis jest przechowywany w osobnym folderze. Katalog `assets` zawiera zasoby wykorzystywane w aplikacji, takie jak dźwięki czy obrazki. Oprócz komponentów stworzonych przez użytkownika wykorzystywane są również pliki wygenerowane za pomocą Angular CLI znajdujące się w katalogu głównym:

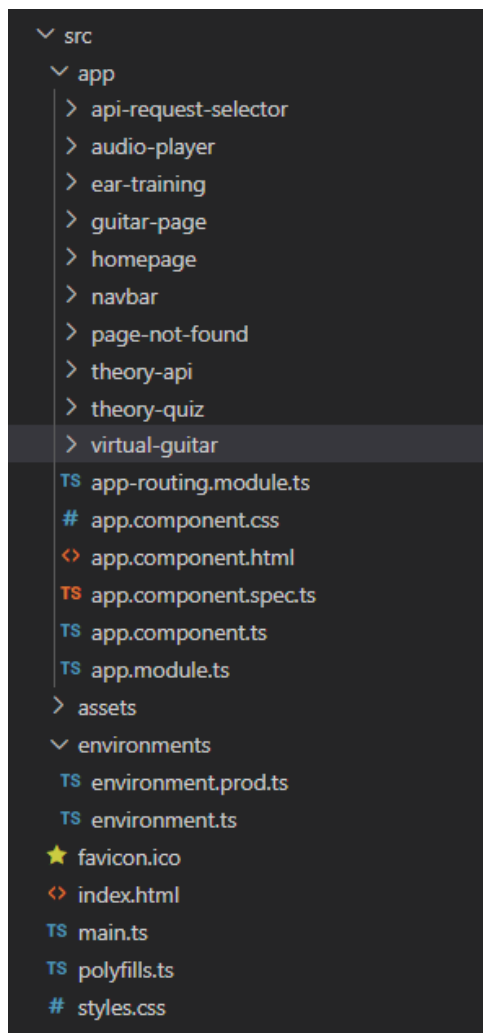
app-routing.module.ts - moduł pozwalający na zarządzanie routinguem wewnątrz projektu

app.module.ts - główny moduł aplikacji opisujący elementy importowane przez aplikację oraz deklarowane w niej komponenty i serwisy

app.* - pozostałe pliki komponentu głównego

environment.prod.ts - plik zawierające zmienne środowiskowe dla środowiska produkcyjnego

environment.ts - zmienne środowiskowe do wykorzystania podczas tworzenia i testowania aplikacji

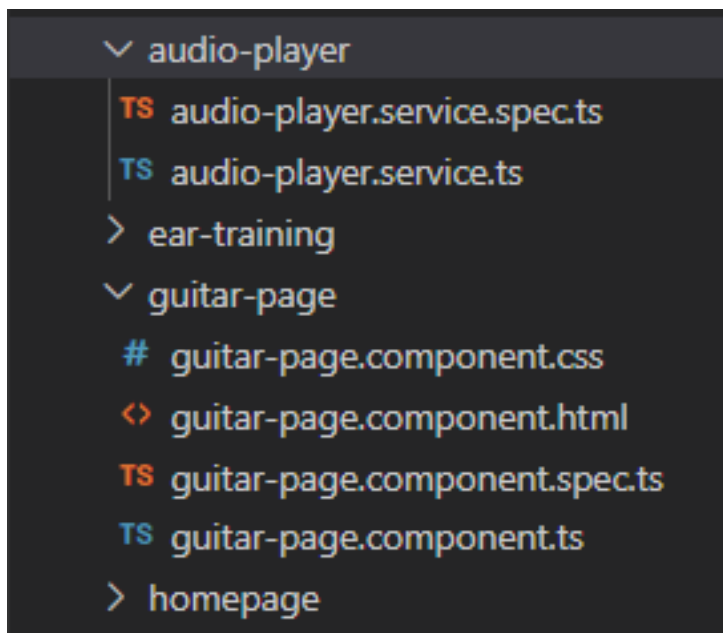


Rys. 5.1: Schemat projektu aplikacji Frontendowej

Każdy komponent składa się z 4 plików:

- component-name.css** - plik opisujący style wykorzystane w obrębie tego komponentu,
- component-name.html** - plik opisujący rozkład elementów w danym komponencie. Angular pozwala na wykorzystywanie tam selektorów innych komponentów oraz dodatkowych dyrektyw, takich jak np. `ngIf` czy `ngFor`,
- component-name.ts** - plik zawierający konstruktor komponentu oraz skrypty napisane przez użytkownika dla danego komponentu,
- component-name.spec.ts** - plik zawierający testy jednostkowe komponentu.

Serwisy natomiast składają się domyślnie z 2 plików: `service-name.ts` oraz `service-name.spec.ts`. Pełnią one takie same role jak w wypadku komponentów - pierwszy z nich zawiera klasę serwisu, natomiast drugi jej testy jednostkowe.



Rys. 5.2: Katalogi audio-player oraz guitar-page pokazują schemat plików odpowiednio serwisu i komponentu

5.2.1. Komunikacja z API

W celu komunikacji z API zaimplementowano serwis `TheoryApiService` oraz modele danych odpowiadające tym z API, czyli klasy: `Chord`, `Interval` oraz `Scale`. Klasę `Scale` przedstawiono na listingu 5.1. Serwis służy do wysyłania zapytań HTTP na zadany adres i rzutuje otrzymane dane na żądany typ.

Listing 5.1: Klasa `Scale` wykorzystywana do zapytań i przechowywania danych o skalach

```
export class Scale {
  constructor(
    public name: string,
    public notes: string[],
  ) { }
}
```

`TheoryApiService` wykorzystuje zmienną środowiskową `API_URL` pod którą przechowywany jest adres API, dzięki czemu można modyfikować adres API bez ingerencji w kod. Zastosowanie zmiennych środowiskowych jest szczególnie istotne kiedy dochodzi do dystrybucji aplikacji - adresy API mogą się zmieniać i nie powinno to powodować potrzeby aktualizacji aplikacji. Przykładowe zapytanie do API z wykorzystaniem zmiennej środowiskowej można zobaczyć na listingu 5.2.

Listing 5.2: Metoda klasy `TheoryApiService` wysyłająca zapytanie o dźwięki akordu do API

```
getChord(rootNote: string, name: string): Observable<Chord[]> {
  return this.http
    .get<Chord[]>(`${API_URL}/chord/${rootNote}/${name}`).pipe(
      tap(_ => console.log('fetched')),
      catchError(TheoryApiService._handleError)
    );
}
```


Takie rozwiązanie pozwala na wykorzystywanie tego samego serwisu w wielu komponentach poprzez wstrzykiwanie zależności. W Angularze proces ten odbywa się poprzez oznaczenie klasy serwisu dekoratorem `Injectable` oraz zadeklarowanie odpowiedniej zależności w konstruktorze komponentu (serwis przekazuje się jako argument konstruktora i przypisuje do odpowiedniego pola w klasie komponentu). Przykład wstrzykiwania zależności można zobaczyć na listingu 5.3, do komponentu są tam wstrzykiwane dwie zależności: `TheoryApiService` oraz `ActivatedRoute`.

Listing 5.3: Wstrzykiwanie zależności do komponentu `GuitarPageComponent`

```
constructor(api: TheoryApiService, route: ActivatedRoute) {
  this.theoryApi = api;
  this.route = route;
}
```

5.2.2. Podstrony z wirtualnymi instrumentami

Podstrony z wirtualnymi instrumentami zostały stworzone poprzez połączenie ze sobą mniejszych komponentów: selektora zapytań, suwaka regulującego głośność oraz wirtualnego instrumentu.

Rys. 5.3: Widok strony z gitarą basową przed wykonaniem zapytania

W aplikacji zostały zaimplementowane dwa instrumenty: gitarę akustyczną oraz gitarę basową. Bazują one na tym samym komponencie, ponieważ na poziomie abstrakcji, na którym operowano podczas implementacji różnica między nimi to tylko liczba strun.

Na każdym z instrumentów po wybraniu zapytania podświetlają się odpowiednie dźwięki. Wynik przykładowego zapytania można zobaczyć na rysunkach 5.4, 5.5 i 5.6. Wybrano zapytanie o skalę F Natural Major (F-dur). W skład tej skali wchodzi dźwięki: F,G,A,A#,C,D,E i te dźwięki są zaznaczone kolorem czerwonym.

Jeśli użytkownika interesują tylko dźwięki należące do wybranego zapytania, może on wyłączyć wyświetlanie pozostałych dźwięków przyciskając guzik “show notes on hover” - powoduje

on ukrycie wszystkich dźwięków, które nie należą do aktualnego zapytania i wyświetlanie ich dopiero po najechaniu kursorem.



Rys. 5.4: Widok strony z gitarą basową po wykonaniu zapytania



Rys. 5.5: Widok strony z gitarą basową po wykonaniu zapytania i przełączeniu opcji hover

5.2.3. Trening słuchu

5.2.4. Quiz

Choose a request

F scale Natural Major Calculate

Volume

Show notes on hover

E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F
B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C
G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#
D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#
A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#
E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F

Rys. 5.6: Widok strony z gitarą akustyczną po wykonaniu zapytania

Rozdział 6

Podsumowanie

Podsumowanie jest miejscem, w którym należy zamieścić syntetyczny opis tego, o czym jest dokument. W szczególności w pracach dyplomowych w podsumowaniu powinno znaleźć się jawnie podane stwierdzenie dotyczące stopnia realizacji celu. Czyli powinny pojawić się w niej akapity ze zdaniami typu: „Podczas realizacji pracy udało się zrealizować wszystkie postawione cele”. Ponadto powinna pojawić się dyskusja na temat napotkanych przeszkód i sposobów ich pokonania, perspektyw dalszego rozwoju, możliwych zastosowań wyników pracy itp.

6.1. Sekcja poziomu 1

Lorem ipsum dolor sit amet eleifend et, congue arcu. Morbi tellus sit amet, massa. Vivamus est id risus. Sed sit amet, libero. Aenean ac ipsum. Mauris vel lectus.

Nam id nulla a adipiscing tortor, dictum ut, lobortis urna. Donec non dui. Cras tempus orci ipsum, molestie quis, lacinia varius nunc, rhoncus purus, consectetur congue risus.

6.1.1. Sekcja poziomu 2

Lorem ipsum dolor sit amet eleifend et, congue arcu. Morbi tellus sit amet, massa. Vivamus est id risus. Sed sit amet, libero. Aenean ac ipsum. Mauris vel lectus.

Sekcja poziomu 3

Lorem ipsum dolor sit amet eleifend et, congue arcu. Morbi tellus sit amet, massa. Vivamus est id risus. Sed sit amet, libero. Aenean ac ipsum. Mauris vel lectus.

Paragraf 4 Lorem ipsum dolor sit amet eleifend et, congue arcu. Morbi tellus sit amet, massa. Vivamus est id risus. Sed sit amet, libero. Aenean ac ipsum. Mauris vel lectus.

6.2. Sekcja poziomu 1

Lorem ipsum dolor sit amet eleifend et, congue arcu. Morbi tellus sit amet, massa. Vivamus est id risus. Sed sit amet, libero. Aenean ac ipsum. Mauris vel lectus.

Literatura

- [1] Angular documentation. *trends.builtwith.com*, Sty. 2022. <https://angular.io/docs>.
- [2] Angular usage statistics. *trends.builtwith.com*, Sty. 2022. <https://trends.builtwith.com/framework/Angular>.
- [3] Flask documentation. <https://flask.palletsprojects.com/en/2.0.x/>, Sty. 2022. <https://flask.palletsprojects.com/en/2.0.x/>.
- [4] Java documentation. <https://docs.oracle.com/en/java/>, Sty. 2022. <https://angular.io/docs>.
- [5] Python 3.9 documentation. <https://docs.python.org/3.9/>, Sty. 2022. <https://docs.python.org/3.9/>.
- [6] React redux usage statistics. *trends.builtwith.com*, Sty. 2022. <https://trends.builtwith.com/javascript/React-Redux>.
- [7] Rest api design. *alexsoft.com*, Sty. 2022. <https://www.altexsoft.com/blog/rest-api-design/>.
- [8] Rest api design. *alexsoft.com*, Sty. 2022. <https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/>.
- [9] Spring documentation. <https://docs.spring.io/spring-framework/docs/current/reference/html/overview.html>, Sty. 2022. <https://docs.spring.io/spring-framework/docs/current/reference/html/overview.html>.
- [10] Y. Lin. 10 internet statistics every marketer should know in 2021. *oberlo.com*, Maj 2021. <https://www.oberlo.com/blog/internet-statistics>.
- [11] M. Marszycki. Gus: Ponad 90 *itwiz.pl*, Gru. 2021. <https://itwiz.pl/gus-ponad-90-gospodarstw-domowych-w-polsce-ma-dostep-do-internetu/>.

Dodatek A

Instrukcja wdrożeniowa

Jeśli praca skończyła się wykonaniem jakiegoś oprogramowania, to w dodatku powinna pojawić się instrukcja wdrożeniowa (o tym jak skompilować/zainstalować to oprogramowanie). Przydałoby się również krótkie how to (jak uruchomić system i coś w nim zrobić – zademonstrowane na jakimś najprostszym przypadku użycia). Można z tego zrobić osobny dodatek,

Dodatek B

Opis załączonej płyty CD/DVD

Tutaj jest miejsce na zamieszczenie opisu zawartości załączonej płyty. Należy wymienić, co zawiera.