|  |  |
| --- | --- |
| **ZUT_2** | **WI_1** |

Szymon Jarząb

nr albumu: 40598

kierunek studiów: Inżynieria cyfryzacji

specjalność: Zastosowania informatyki

forma studiów*: stacjonarne*

APLIKACJA WEBOWA DO ŁĄCZENIA I SYNCHRONIZOWANIA NAGRAŃ ŚCIEŻEK DŹWIĘKOWYCH INSTRUMENTÓW MUZYCZNYCH

A WEB APPLICATION FOR CONNECTING AND SYNCHRONIZING SOUND TRACK RECORDS OF MUSICAL INSTRUMENTS

praca dyplomowa inżynierska

napisana pod kierunkiem:

dr inż. Agnieszki Konys

Katedra Inżynierii Oprogramowania i Cyberbezpieczeństwa

Data wydania tematu pracy: 31.01.2021

Data dopuszczenia pracy do egzaminu: …………………………………………………………………  
(uzupełnia pisemnie Dziekanat)

Szczecin, 2021

Spis treści

[1. Wprowadzenie 3](#_Toc60831916)

[1.1. Wstęp 3](#_Toc60831917)

[1.2. Cel pracy 4](#_Toc60831918)

[1.3. Zakres pracy 5](#_Toc60831919)

[1.4. Przegląd i analiza funkcjonalności istniejących rozwiązań 6](#_Toc60831920)

[2. Realizacja celu 7](#_Toc60831921)

[2.1. Uzasadnienie wyboru środowiska programistycznego 7](#_Toc60831922)

[2.2. Analiza porównawcza istniejących rozwiązań 8](#_Toc60831923)

[2.3. Specyfikacja wymagań 9](#_Toc60831924)

[2.4. Projekt aplikacji 10](#_Toc60831925)

[2.5. Implementacja aplikacji 11](#_Toc60831926)

[2.6. Testy 12](#_Toc60831927)

[3. Sprawozdanie z realizacji celu 13](#_Toc60831928)

[3.1. Stopień realizacji celu 13](#_Toc60831929)

[3.2. Wykorzystane algorytmy 14](#_Toc60831930)

[3.3. Napotkane problemy 15](#_Toc60831931)

[4. Podsumowanie 16](#_Toc60831932)

[4.1. Wnioski 16](#_Toc60831933)

[4.2. Słownik terminologiczny 17](#_Toc60831934)

[4.3. Źródła 18](#_Toc60831935)

# Wprowadzenie

## Wstęp

Sytuacja na świecie w latach 2020/21 zmusiła wiele branż do zmiany utartych schematów funkcjonowania. Pod groźbą zawieszenia działalności wiele firm czy ugrupowań musiało znaleźć nowe sposoby na prowadzenie codziennej rutyny.

Niektóre, wcześniej nierozpowszechnione rozwiązania, takie jak nauczanie zdalne lub …… zyskały na tym, zdobywając przynajmniej tymczasową popularność, ale co ważniejsze, ich pojawienie się może wymusić na opinii publicznej przyznanie, że w jakimś stopniu wchodzimy w nową erę, którą w dodatku sami mamy okazję budować. Wielu może twierdzić, że spoczywa na nas wielka odpowiedzialność, jednak dużo przyjemniej myśleć o tym jako o olbrzymich możliwościach. W obecnej sytuacji to od nas wszystkich zależy jak będzie wyglądało jutro, które możemy tworzyć w oparciu o nasze potrzeby, których tak wiele ujawniło się w tak krótkim czasie. Rozwiązania, do których jesteśmy zmuszeni przez ograniczenie kontaktów międzyludzkich nie muszą przecież zostać z nami na zawsze. Po ustabilizowaniu się sytuacji na świecie zostaną możliwością, z której dobrowolnie będziemy mogli korzystać.

Część branż była w jakimś stopniu zdolna do dostosowania się, jak na przykład gastronomia, która pomimo ograniczenia jej roli do dostarczania jedzenia, z pominięciem oferowania miejsca spotkań czy odskoczni od codziennej rutyny mogła dalej funkcjonować na akceptowalnym poziomie.

Zostają jeszcze te gałęzie naszego życia, które przy przestrzeganiu wprowadzonych obostrzeń niemal całkowicie tracą możliwość istnienia. Wystarczy wyobrazić sobie wizytę w filharmonii, kiedy na scenie może zmieścić się zaledwie kilku muzyków, a ilość ludzi na widowni nie pozwala myśleć, że jest się uczestnikiem wyjątkowego wydarzenia. Oczywistym jest stwierdzenie, że koncert straci wiele na uroku, można nawet posunąć się do stwierdzenia, że nie warto na niego iść.

To właśnie problemów filharmonii, a ogólniej mówiąc różnego rodzaju orkiestr będzie dotyczyła ta praca.

Próbą rozwiązania powyższego problemu, a dokładniej mówiąc umożliwienia pracy orkiestrom pomimo braku możliwości spotykania się w większej grupie, było nagrywanie muzyków osobno i łączenie nagrań w jeden utwór. Tutaj pojawia się kolejny problem. Kiedy muzyk gra na scenie jako część całej orkiestry, dużo łatwiej jest mu wyczuć rytm, natomiast nagrywany osobno z wielu względów gra nierówno. Połączenie nagrań bez odpowiedniego przygotowania\* ich daje efekt daleki od oczekiwanego. Większość nagrań musi być przed połączeniem przetworzona w taki sposób, żeby poszczególne dźwięki w docelowym utworze były zagrane w odpowiednim momencie. Ręczne przetwarzanie pojedynczego nagrania trwa, zależnie od jakości nagrania (tutaj jakość dotyczy tego, ile dźwięków w nagraniu trzeba poprawić) od kilku minut do nawet godziny dla pojedynczego utworu. Oczywistym jest, że przy koncercie składającym się z kilkunastu utworów granym przez orkiestrę składającą się czasami nawet ze stu muzyków, czas potrzebny do ręcznego przetworzenia nagrań jest tak długi, że uniemożliwia płynne funkcjonowanie orkiestry. Dodatkowym problemem jest znikoma ilość osób, które zajmują się takim przetwarzaniem, nie wspominając nawet o jakości ich usług, która często okazuje się dużo niższa od wymaganej. Tytułowa aplikacja ma na celu zmniejszenie ilości czasu potrzebnego na przetwarzanie nagrań.

Praca, pomimo jej naukowego charakteru, będzie napisana w możliwie najprostszym języku, aby każdy, nawet niezaznajomiony z tematem był w stanie zrozumieć ją jak największym stopniu. W rozumieniu pomoże słownik pojęć, w którym docelowo znajdą się pojęcia, które w jakiś sposób mogą być niezrozumiałe. Dodatkowo, w granicach możliwości, treść pracy będzie wspomagana przykładami, zapewne najczęściej słownymi, jednak czasami też, jako że zagadnienie dotyczy muzyki, nagraniami pozwalającymi lepiej zrozumieć opisywaną w danym momencie sytuację czy problem. Praca będzie miała na celu zaprogramowanie aplikacji, jednak biorąc pod uwagę fakt, że nie każdy musi znać się na programowaniu, fragmenty kodu zawarte w dokumencie będą w miarę możliwości zastępowane rysunkami poglądowymi czy opisem słownym. Cały projekt, wliczając postępy pracy, kod oraz dokumentację, będzie zapisywany za pośrednictwem serwisu GitHub oraz, po jego ukończeniu, zostanie zapisany na fizycznym nośniku w celu archiwizacji.

Zarówno kod jak i zawarta w nim dokumentacja kodu będą napisane w języku angielskim, zgodnie z powszechnie panującymi standardami.

## Cel pracy

Celem pracy jest stworzenie aplikacji webowej pozwalającej na łączenie i synchronizację ścieżek dźwiękowych pojedynczych instrumentów muzycznych\*, wspierającą zdalną pracę orkiestr przez ograniczanie czasu potrzebnego na ręczną synchronizację.

## Zakres pracy

Podstawą do uznania celu pracy za osiągnięty jest zaprogramowanie i opisanie aplikacji, która w chociażby minimalnym stopniu zmniejsza czas pracy potrzebny do ręcznego przetworzenia ścieżek dźwiękowych i połączenia ich w pojedynczy utwór.

Cel ze względu na wiele czynników, w tym jego złożoność i innowacyjność może okazać się niemożliwy do osiągnięcia środkami dostępnymi w ramach pisania pracy. W takim wypadku docelowa praca będzie skupiała się przede wszystkim na opisaniu napotkanych problemów, omówieniu potencjalnych rozwiązań i wyciąganiu wniosków.

Jeśli cel zostanie osiągnięty, kolejnym krokiem będzie ogólny rozwój aplikacji, na przykład optymalizacja, udoskonalanie niezawodności czy przystępności użytkowania.

## Przegląd i analiza funkcjonalności istniejących rozwiązań

Innowacyjność oraz dopiero niedawne pojawienie się zapotrzebowania na funkcjonalności zawarte w projekcie powodują, że przewidziana w ramach pracy aplikacja nie tylko wypełnia niszę powstałą na rynku, ale równocześnie ma szansę na uzyskanie miana pioniera w tej dziedzinie.

# Realizacja celu

## Uzasadnienie wyboru środowiska programistycznego

Na wybór środowiska programistycznego może wpływać mnóstwo różnych czynników. Przystępnym sposobem na przedstawienie argumentów przemawiających za dokonanym wyborem będzie podzielenie ich na poszczególne grupy, które następnie podsumowane nie powinny pozostawić wątpliwości, że wybór jest prawidłowy i najlepszy z możliwych.

# Znajomość języka, popularność języka, dostępne narzędzia, prostota języka, szybkość implementacji, czytelność języka, wydajność (pypy), skalowalność projektu, testy, python jako narzędzie naukowe, optymalizacja gotowych bibliotek

Oczywiście na początku należałoby sprecyzować wybór. Przede wszystkim jest to język Python 3, który będzie stanowił zdecydowaną większość końcowego kodu. Dodatkowo, jako że aplikacja docelowo ma być webową, naturalnym będzie, że użyte zostaną HTML5 oraz CSS. Frameworkiem webowym będzie Django.

Najważniejszym argumentem, który sam mógłby przesądzić o dokonanym wyborze jest znajomość języka. Pomimo używania różnych języków, Python był tym, z którym miałem styczność przez większość czasu spędzonego na programowaniu.

<https://www.tiobe.com/tiobe-index/>

<https://pypl.github.io/PYPL.html>

<https://redmonk.com/sogrady/category/programming-languages/>

Za Pythonem przemawia również jego popularność. Najczęściej jest przedstawiany jako jeden z najpopularniejszych języków programowania, chociaż niektóre źródła stawiają go na pierwszym miejscu w rankingu. Sama popularność jest istotna z powodu możliwego wsparcia ze strony innych programistów. Przy takich rozmiarach społeczności Pythona zdecydowana większość problemów, które można spotkać podczas programowania została już przez kogoś napotkana i omówiona. To pozwala na upłynnienie procesu implementacji, ograniczając czas potrzebny na samodzielne rozwiązanie każdego kolejnego problemu. Dodatkowo, jeśli jakiś problem nie został jeszcze opisany, a rozwiązanie go samemu okazuje się zbyt trudne, dobrym wyjściem będzie prośba o pomoc. Często odpowiedzi na zadane pytania pojawiają się dosłownie w kilka minut.

Istotny jest także dostęp do bibliotek. Python jako narzędzie open-source dzięki olbrzymiej społeczności używającej go ma mnóstwo bibliotek, bez których stworzenie tytułowej aplikacji byłoby niemożliwe. Przy projekcie w ramach którego przewidziane są odczytywanie, edycja czy tworzenie plików oraz zarządzanie ich formatami niezbędny jest dostęp do bibliotek, które te czynności umożliwiają, ponieważ gdyby takich bibliotek nie było, to samo napisanie ich od zera mogłoby stanowić osobny projekt. Przy istniejącym zestawie bibliotek projekt jest możliwy do implementacji bez tworzenia nadmiernie zajmujących czas funkcjonalności, które nie dotyczą bezpośrednio problemu opisywanego w pracy.

Prostota języka, która jest jedną z myśli przewodnich Pythona zdecydowanie pomaga przy wszelkich projektach. Oczywistym jest, że na poziomie pracy inżynierskiej możliwości są ograniczone posiadaną wiedzą. W trakcie tworzenia i rozwijania aplikacji bez żadnych wątpliwości ujawnią się braki wiedzy. Ważne jest to, aby te braki były łatwe do nadrobienia, ponieważ nie jest możliwym przewidzenie wszystkiego, co będzie wymagane w ramach projektu. Python jako jeden z przystępniejszych do nauki języków jest świetnym wyborem, dodatkowo jego prostota nie wpływa na jego użyteczność, co również jest atutem.

Szybkość implementacji jest istotna w dosłownie każdym projekcie. Wpływa pozytywnie na rozwój całego procesu tworzenia, pozwalając na większe pokrycie kodu testami zwiększając tym samym niezawodność wersji końcowej, dając więcej czasu na wprowadzenie poprawek czy nowych funkcjonalności udoskonalających produkt czy po prostu skracając czas potrzebny na osiągnięcie celu. Python ze swoim zestawem narzędzi wspiera szybki rozwój wszelkich projektów, dostarczając mnóstwo gotowych rozwiązań.

Podczas programowania rozbudowanego projektu istotna jest wysoka czytelność kodu. Jest ona możliwa do osiągnięcia na różne sposoby, na przykład przez styl pisania programisty, nazewnictwo funkcji czy zmiennych czy utrzymana w całym kodzie spójność formatowania. Python jako język wysokiego poziomu\* wspiera w dużym stopniu czytelność kodu. Upłynnia to proces programowania, ułatwiając rozwój istniejących elementów pozwalając na szybsze przypomnienie sobie, jak zostały zaimplementowane czy po prostu pozwalając na łatwiejsze tworzenie aplikacji przez przybliżenie składni języka do ludzkiego.

Chociaż wiele osób postrzega Pythona jako język o niskiej wydajności, to jest to mylne wrażenie. O ile podstawowa implementacja Pythona może być dość wolna, tak inna, PyPy, może przyspieszyć wykonywanie kilkukrotnie. Nadal nie będzie to prędkość dorównująca najwydajniejszym językom programowania, jednak w znacznym stopniu niweluje to argument, który mógłby być użyty przeciwko używaniu Pythona.

Prawdopodobne jest, że aplikacja będzie rozwijana w przyszłości, a w takim wypadku istotna jest jak największa jej skalowalność. Odpowiednie projektowanie aplikacji, wysoka czytelność kodu czy orientacja obiektowa są wspierane przez Pythona, a każde z nich wpływa znacząco na skalowalność aplikacji.

Po tworzenia aplikacji ważna jest jej niezawodność. Wysokie pokrycie projektu testami pozwala na znaczne zmniejszenie jego wadliwości. Pomimo tego, że temat pracy nie dotyczy w żaden sposób testowania, warto przetestować chociażby najważniejsze funkcjonalności. Python jest pomocny również tutaj, pozwalając na przetestowanie aplikacji z pomocą takich narzędzi jak unittest, pytest czy Selenium.

Naukowy charakter projektu, przewidziane w nim przetwarzanie danych i stworzenie aplikacji to kolejne powody, by użyć Pythona. Dostępne biblioteki naukowe takie jak SciPy czy NumPy mogą w dużym stopniu ułatwić pracę z projektem, a dzięki frameworkom webowym Pythona możliwe będzie wykorzystanie zaimplementowanych algorytmów w zaplanowanej aplikacji.

Ostatnim argumentem przemawiającym za Pythonem będzie jakość dostępnych w nim narzędzi. Cały projekt zyskuje na wydajności im więcej gotowych narzędzi używa, ponieważ za każdym z nich stoją osoby, które nie tylko je stworzyły, ale także testowały i optymalizowały. Zdecydowana większość wykorzystywanych bibliotek dostarcza funkcjonalności tak dopracowanych, że samodzielny programista, tym bardziej skupiający się przede wszystkim na jakości ogólnej projektu, nie byłby w stanie zastąpić ich własnym kodem zachowując przy tym jakość i wydajność.

Te argumenty nie powinny pozostawiać najmniejszych wątpliwości, że Python będzie bardzo dobrym narzędziem do wykorzystania w tym projekcie. Jego ogromnym atutem jest to, że będzie występował w projekcie od początku do końca, dzięki czemu łatwiej będzie się skupić na tworzeniu projektu bez konieczności stosowania za dużej ilości innych narzędzi.

HTML5 oraz CSS są oczywistym wyborem, dla którego nie ma żadnych wartych rozważenia alternatyw, więc nie miałoby sensu rozwodzenie się nad tym, że ich wybór jest właściwy.

Wybór frameworka webowego nie jest wielkim wyzwaniem. Trzymając się tego, że projekt będzie pisany w Pythonie, najlepszymi możliwościami będą Django oraz Flask. Końcowym wyborem jest Django, który zawiera wiele gotowych rozwiązań, wspiera uporządkowanie plików zawartych w projekcie i, najprościej mówiąc, jest dużo bardziej przyjazny dla programisty niż Flask.

## Specyfikacja wymagań

### Wymagania funkcjonalne

|  |  |
| --- | --- |
| RF01 | System pozwala na synchronizację ścieżek dźwiękowych |
| RF02 | System pozwala na łączenie ścieżek dźwiękowych |
|  | System pozwala na wygenerowanie metronomu |
|  | System posiada panel administratora |
|  | System pozwala na rejestrację użytkowników |
|  | System pozwala na logowanie się użytkowników |
|  |  |
|  |  |
| RF0 | System pozwala na odczytywanie obsługiwanych formatów plików |
| RF0 | System pozwala na zarządzanie obsługiwanymi formatami plików |
| RF0 |  |
| RF0 | System pozwala na tworzenie plików formatu wav |
|  | System pozwala na wyświetlenie danych odczytanego pliku w formie wykresu |
|  | System pozwala na edycję danych odczytanego pliku |
|  |  |
| RF0 | System pozwala na wygenerowanie metronomu o wskazanych parametrach częstotliwości, tempa, czasu trwania oraz dźwięku |
| RF0 | System pozwala na zapisanie wygenerowanego metronomu w formacie wav |
| RF0 |  |
| RF0 |  |
| RF0 |  |

### Wymagania niefunkcjonalne

|  |  |
| --- | --- |
| RNF0 | System jest oparty głównie na języku Python |
| RNF0 | Aplikacja jest oparta na frameworku webowym Django |
| RNF0 | Frontend aplikacji ogranicza się do języków HTML5 i CSS |
| RNF0 | System korzysta z bazy danych PostgreSQL |
| RNF0 | Wykorzystywany system kontroli wersji to Git/GitHub |
| RNF0 | Główny framework testowy to unittest |
| RNF0 | Testy akceptacyjne wykonane za pomocą Selenium |
| RNF0 | Dokumentacja kodu pozwala na zrozumienie go |
| RNF0 | System korzysta z protokołu HTTP |
| RNF0 | Kod jest formatowany zgodnie z zasadami PEP 8 |
|  |  |
| RNF0 | System jest w formie umożliwiającej zaprezentowanie jego funkcjonalności |
| RNF0 | System nie posiada błędów uniemożliwiających korzystanie z niego |
| RNF0 | Interfejs użytkownika jest intuicyjny |
|  |  |
| RNF0 | System obsługuje formaty plików wav, mp3, m4a, flv, raw, ogg |
|  |  |
|  |  |
|  |  |

## Projekt aplikacji

## Implementacja aplikacji

## Testy

# Sprawozdanie z realizacji celu

## Stopień realizacji celu

## Wykorzystane algorytmy

## Napotkane problemy

# Podsumowanie

## Wnioski

## Słownik terminologiczny

Pojedyncza ścieżka dźwiękowa instrumentu muzycznego – pojedynczy głos instrumentu muzycznego, przeznaczony do synchronizacji i połączenia w utwór (jeśli w orkiestrze grają cztery flety, to pojęcie odnosi się do nagrania dokładnie jednego z nich)

Łączenie ścieżek dźwiękowych – połączenie kilku plików dźwiękowych w jeden

Synchronizowanie ścieżek dźwiękowych – modyfikacja ścieżek dźwiękowych mająca na celu dopasowanie nut do oczekiwanego rytmu

## Źródła