POLITECHNIKA GDAŃSKA | WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

Imię i nazwisko studenta: Hanna Banasiak
Nr albumu: 193078
Poziom kształcenia: studia pierwszego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Informatyka
Profil: Systemy geoinformatyczne

Imię i nazwisko studenta: Aleksandra Bujny
Nr albumu: 193186
Poziom kształcenia: studia pierwszego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Informatyka
Profil: Bazy danych

Imię i nazwisko studenta: Karol Zwierz
Nr albumu: 193603
Poziom kształcenia: studia pierwszego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Informatyka
Profil: Architektura systemów komputerowych

Imię i nazwisko studenta: Jakub Romanowski
Nr albumu: 193637
Poziom kształcenia: studia pierwszego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Informatyka
Profil: Architektura systemów komputerowych

PRACA DYPLOMOWA INŻYNIERSKA

Tytuł pracy w języku polskim: Aplikacja do treningu oddechowego

Tytuł pracy w języku angielskim: Breathing training application

Opiekun pracy: dr hab. inż. Julian Szymański

STRESZCZENIE

Kontekst: W obliczu rosnącej popularności technik oddechowych w sporcie, redukcji stresu oraz rehabilitacji pulmonologicznej, zidentyfikowano potrzebę stworzenia wysoce konfigurowalnego narzędzia wspierającego w ich praktykowaniu. Analiza aktualnego stanu wiedzy oraz istniejących rozwiązań pokazała, że obecnie dostępne rozwiązania często narzucają sztywne schematy ćwiczeń, ograniczając możliwość indywidualnego dostosowania treningu do potrzeb użytkownika.

Cel: Głównym celem pracy inżynierskiej jest opracowanie rozwiązania oraz implementacja wieloplatformowej mobilnej aplikacji umożliwiającej przeprowadzanie treningu oddechowego umożliwiającej przeprowadzenie wysoce konfigurowalnych treningów oddechowych. Zakres prac obejmował analizę wymagań, projekt architektury i interfejsu oraz implementację oprogramowania.

Metody: Aplikacja została zrealizowana w oparciu o platformę Flutter w języku Dart, co umożliwiło wsparcie dla zarówno systemu Android jak i iOS przy wykorzystaniu wspólnego kodu źródłowego. Zastosowano modularną architekturę oprogramowania oraz lokalną, nierelacyjną bazę danych Hive do zapewnienia trwałości danych. Projektowanie poprzedzono analizą porównawczą istniejących rozwiązań na rynku.

Wyniki: W ramach prac stworzono aplikację “ReSpire”, wyposażoną w kompleksowy edytor umożliwiający tworzenie wieloetapowych sekwencji treningowych. Zaimplementowano audiowizualny system przeprowadzania użytkownika, integrujący animowane wizualizacje przebiegu faz z wielowarstwową ścieżką dźwiękową — obsługującą tło muzyczne, wskazówki głosowe oraz generator dudnień różnicowych.

Wnioski: Opracowane rozwiązanie skutecznie łączy prostotę obsługi dla początkujących z elastycznością wymaganą przez użytkowników zaawansowanych, eliminując dostrzeżone na etapie analizy ograniczenia konkurencyjnych produktów.

Słowa kluczowe: trening oddechowy, zdrowie cyfrowe, elastyczność, konfigurowalność, Flutter, aplikacja mobilna, fazy oddechu, oprawa dźwiękowa, personalizacja treningu

Dziedzina nauki i techniki, zgodnie z wymogami OECD: Nauki inżynieryjne i techniczne, inżynieria informatyczna

ABSTRACT

Context: In light of the increasing popularity of breathing techniques in sport, stress reduction, and pulmonary rehabilitation, the need to develop a highly configurable tool to facilitate users' practice has been identified. An analysis of the state of the art and existing market solutions revealed that currently available tools often impose rigid exercise patterns, thereby limiting the scope for individual customisation of training to meet user needs.

Objective: The primary aim of this engineering thesis is to design and implement a cross-platform mobile application capable of facilitating highly configurable breathing exercises. The scope of work included requirements analysis, architecture and interface design, and software implementation.

Methods: The application was developed using the Flutter framework and the Dart language, which enabled support for both Android and iOS systems via a single codebase. A modular software architecture was employed alongside Hive, a local non-relational database, to ensure data persistence. The design phase was preceded by a comparative analysis of existing solutions available on the market.

Results: The project resulted in the creation of 'ReSpire', an application featuring a comprehensive editor that allows for the construction of multi-stage training sequences. An audiovisual user guidance system was implemented, integrating animated visualisations of phase progression with a multi-layered audio track that supports background music, voice instructions, and a binaural beats generator.

Conclusions: The developed solution effectively combines the ease of use for beginners with the flexibility required by advanced users, successfully overcoming the limitations of competing products identified during the analysis stage.

Keywords: breathing training, digital health, flexibility, configurability, Flutter, mobile application, breathing phases, audio setting, training personalization

SPIS TREŚCI

1. WSTĘP I CEL PRACY (Hanna Banasiak, Jakub Romanowski, Karol Zwierz) 8
1.1. Wprowadzenie (Hanna Banasiak) 8
1.2. Motywacje (Hanna Banasiak, Karol Zwierz) 8
1.3. Cel pracy (Hanna Banasiak) 9
1.4. Podział prac w zespole (Jakub Romanowski) 9
1.4.1. Hanna Banasiak 10
1.4.2. Aleksandra Bujny 10
1.4.3. Jakub Romanowski 10
1.4.4. Karol Zwierz 10
1.5. Struktura rozdziałów (Hanna Banasiak) 11
2. AKTUALNY STAN WIEDZY (Hanna Banasiak, Aleksandra Bujny, Jakub Romanowski, Karol Zwierz) 12
2.1. Kontekst naukowy i skuteczność aplikacji mobilnych (Karol Zwierz) 12
2.2. Istniejące rozwiązania wspierające trening oddechu (Jakub Romanowski) 12
2.3. Wim Hof Method firmy WHM Services (Hanna Banasiak) 12
2.3.1. Cel aplikacji 12
2.3.2. Funkcjonalności 13
2.3.3. Ograniczenia 15
2.4. Breathwrk: Breathing Exercises firmy Breathwrk Inc. (Aleksandra Bujny) 15
2.4.1. Cel aplikacji 15
2.4.2. Funkcjonalności 16
2.4.3. Ograniczenia 20
2.5. STAmina Apnea Trainer firmy Squarecrowd Apps SIA (Karol Zwierz) 20
2.5.1. Cel aplikacji 20
2.5.2. Funkcjonalności 20
2.5.3. Ograniczenia 22
2.6. Breathe firmy Havabee (Jakub Romanowski) 22
2.6.1. Cel aplikacji 22
2.6.2. Funkcjonalności 22
2.6.3. Ograniczenia 24
3. ZAŁOŻENIA PROJEKTOWE (Aleksandra Bujny, Karol Zwierz) 25
3.1. Dostarczane korzyści (Karol Zwierz) 25
3.2. Główne funkcje aplikacji (Karol Zwierz) 25
3.3. Przewidziani użytkownicy (Karol Zwierz) 26
3.4. Systemy współpracujące (Karol Zwierz) 26
4. ANALIZA WYMAGAŃ (Hanna Banasiak, Aleksandra Bujny) 27
4.1. Model przypadków użycia (Aleksandra Bujny) 27
4.2. Model sterowania dźwiękami (Hanna Banasiak) 32

5. PROJEKT ROZWIĄZANIA (Hanna Banasiak, Aleksandra Bujny, Jakub Romanowski) 34
5.1. Projekt architektury systemu (Aleksandra Bujny) 34
5.1.1. Architektura warstwowa 34
5.1.2. Architektura monolityczna 34
5.1.3. Modularny podział aplikacji 35
5.1.4. Model obiektów stron 35
5.1.5. Stan aplikacji i warstwa prezentacji 35
5.1.6. Model serwisów 35
5.1.7. Model treningu 37
5.2. Projekt interfejsu użytkownika (Hanna Banasiak, Aleksandra Bujny, Karol Zwierz) 38
5.2.1. Projekt w narzędziu Figma 39
5.2.2. Zrealizowany wygląd interfejsu użytkownika aplikacji 41
5.2.3. Identyfikacja wizualna aplikacji 43
5.3. Projekt struktury danych (Jakub Romanowski) 45
6. IMPLEMENTACJA APLIKACJI MOBILNEJ RESPIRE (Hanna Banasiak, Aleksandra Bujny, Jakub Romanowski, Karol Zwierz) 47
6.1. Schemat plików/klas/modułów (Aleksandra Bujny, Jakub Romanowski) 47
6.2. Edytor treningów (Karol Zwierz) 49
6.2.1. Menu - trening 49
6.2.2. Menu - dźwięki 49
6.2.3. Menu - inne 52
6.3. Importowanie oraz eksportowanie treningów (Karol Zwierz) 52
6.4. Przebieg treningu (Hanna Banasiak) 55
6.4.1. Klasa TrainingParser 55
6.4.2. Klasa TrainingController 58
6.4.3. Klasa AnimatedCircle 66
6.4.4. Klasa InstructionSlider 68
6.5. Dźwięki (Jakub Romanowski) 70
6.5.1. SoundManager 71
6.5.2. SingleSoundManager 72
6.5.3. PlaylistManager 72
6.5.4. AudioPlayerPool 73
6.6. Języki (Jakub Romanowski) 73
6.7. Generator list dźwięków (Aleksandra Bujny) 75
6.8. Baza danych (Karol Zwierz) 77
6.8.1. Architektura i generowanie kodu 77
6.8.2. Model danych 79
6.8.3. Organizacja danych w Box'ach 79
6.8.4. Inicjalizacja bazy danych 80
7. TESTOWANIE APLIKACJI (Hanna Banasiak) 82
7.1. Testowanie statyczne 82
7.2. Testowanie dynamiczne 83

8. PODSUMOWANIE (Aleksandra Bujny, Karol Zwierz) . . . 84
8.1. Osiągnięte rezultaty (Karol Zwierz) . . . 84
8.2. Napotkane problemy i ogarniczenia (Karol Zwierz) . . . 84
8.3. Wnioski z projektu (Karol Zwierz) . . . 84
8.4. Porównanie opracowanego rozwiązania z istniejącymi rozwiązaniami (Aleksandra Bujny) . 84
8.5. Możliwości rozwoju (Aleksandra Bujny) . . . 85
Bibliografia . . . 86
A. INSTRUKCJA UŻYTKOWANIA (Aleksandra Bujny) . . . 88
A.1. Instalowanie, odinstalowanie, uruchamianie i aktualizacja aplikacji . . . 88
A.1.1. Instalowanie aplikacji . . . 88
A.1.2. Uruchamianie aplikacji . . . 89
A.1.3. Odinstalowanie aplikacji . . . 89
A.1.4. Aktualizacja aplikacji . . . 90
A.2. Strona główna . . . 90
A.2.1. Lista treningów . . . 91
A.2.2. Pasek z opcjami . . . 91
A.2.3. Tryb zaznaczania . . . 92
A.3. Strona szczegółów treningu . . . 94
A.4. Strona edycji treningu . . . 95
A.4.1. Zakładka Trening . . . 96
A.4.2. Zakładka Dźwięki . . . 100
A.4.3. Zakładka Inne . . . 106
A.5. Strona treningu oddechowego . . . 108
A.6. Strona ustawień . . . 109

1. WSTĘP I CEL PRACY (Hanna Banasiak, Jakub Romanowski, Karol Zwierz)

1.1. Wprowadzenie (Hanna Banasiak)

W ostatnich latach obserwuje się wzrost zainteresowania technikami wspierającymi zdrowie psychiczne oraz treningi fizjologiczne. Ważną rolę w tym obszarze odgrywają ćwiczenia oddechowe wykorzystywane w sporcie, terapii i codziennym dbaniu o dobre samopoczucie. Aplikacje mobilne pełnią w tym zakresie istotną rolę, sprzyjając ich popularności. Umożliwiają użytkownikom stały dostęp do zindywidualizowanych programów treningowych z dowolnego miejsca i w dowolnym czasie, pozwalając użytkownikowi dopasować przebieg ćwiczenia do własnych preferencji i celów.

Niniejsza praca koncentruje się na zaprojektowaniu aplikacji mobilnej, która umożliwia wykonywanie treningów oddechowych z wykorzystaniem konfigurowalnych wzorców oddechowych i czytelnej prezentacji ich przebiegu.

1.2. Motywacje (Hanna Banasiak, Karol Zwierz)

Motywacją do realizacji projektu jest stworzenie narzędzia ułatwiającego wykonywanie ćwiczeń oddechowych, które mogą wspierać zarówno zdrowie psychiczne, jak i fizyczne. Regularne treningi oddechowe znajdują zastosowanie w wielu grupach odbiorców.

Badania naukowe wskazują na terapeutyczny potencjał regularnych ćwiczeń oddechowych. Systematyczne przeglądy literatury potwierdzają, że techniki takie jak *pursed lip breathing* (pol. oddychanie przez zaciśnięte usta), oddychanie przeponowe oraz techniki jogiczne (np. pranajama) przyczyniają się do istotnej poprawy jakości życia oraz zmniejszenia odczuwania duszności u pacjentów z przewlekłymi chorobami układu oddechowego, takimi jak POChP czy astma [1]. Analiza ta, obejmująca również metody takie jak technika Buteyko [2] czy metoda Papworth [3], wskazuje na wysoki poziom bezpieczeństwa tych interwencji — w przeglądanych badaniach nie odnotowano istotnych działań niepożądanych związanych z samym treningiem. Jest to kluczowy argument przemawiający za zasadnością udostępniania tych narzędzi w formie aplikacji mobilnej do samodzielnego stosowania.

Co więcej, poprawa *Health-Related Quality of Life* (pol. jakości życia związana]ej ze zdrowiem) obserwowana u pacjentów stosujących te techniki często przekraczała minimalną różnicę istotną klinicznie, co potwierdza, że korzyści te są odczuwalne i znaczące dla pacjenta w codziennym funkcjonowaniu. Wykazano również bezpośredni wpływ tych ćwiczeń na poprawę jakości snu, co ma szczególne znaczenie w rehabilitacji osób starszych cierpiących na przewlekłe schorzenia płuc [4]. Mechanizm ten wiązany jest ze stymulacją aktywności nerwu błędnego poprzez powolne, rytmiczne oddychanie, co sprzyja relaksacji i regulacji układu autonomicznego.

W kontekście zdrowia psychicznego, interwencje oparte na świadomej kontroli oddechu zyskują na znaczeniu jako metody wspierające leczenie zaburzeń nastroju. Badania kliniczne dowodzą, że zarówno specyficzne metody, takie jak metoda Wima Hofa (łącząca techniki oddechowe z ekspozycją na zimno), jak i tradycyjne powolne oddychanie, mogą skutecznie redukować objawy depresji, lęku oraz poziom odczuwanego stresu. W badaniu z udziałem kobiet z wysokim poziomem objawów depresyjnych odnotowano redukcję tych objawów o 24%, lęku o 27% oraz postrzeganego stresu o 20% bezpośrednio po interwencji. Co istotne, poprawa ta utrzymywała się

również w okresie obserwacji po 3 miesiącach, gdzie blisko połowa badanych zgłaszała łagodne objawy lub ich brak. Warto również odnotować, że niektóre z tych metod mogą być szczególnie skuteczne w zmniejszaniu ruminacji, czyli uporczywego nawracania negatywnych myśli [5].

Równocześnie, rozwój technologii mobilnych otwiera nowe możliwości w zakresie dostępności rehabilitacji oddechowej. Metaanalizy wskazują, że aplikacje mobilne wspierające samodzielny trening są skutecznym narzędziem w rehabilitacji pulmonologicznej, oferując efekty porównywalne do tradycyjnych metod. Kluczowym czynnikiem sukcesu jest tu systematyczność — wykazano korelację między częstotliwością korzystania z aplikacji a redukcją objawów choroby (mierzoną np. w skali CAT). Aplikacje umożliwiające personalizację planów treningowych oraz dostosowanie intensywności ćwiczeń do indywidualnych możliwości pacjenta są wskazywane jako najbardziej obiecujący kierunek rozwoju, pozwalający na utrzymanie długoterminowego zaangażowania użytkownika [6]. Aplikacja ReSpire wpisuje się w ten trend, oferując konfigurowalne wzorce oddechowe, które mogą być dostosowane do zaleceń medycznych i preferencji użytkownika.

Aplikacja ReSpire może służyć:

1. sportowcom — poprawienie wytrzymałości [7];
2. pacjentom z zespołami bólowymi i chorobami układu oddechowego takimi jak astma, przewlekła obturacyjna choroba płuc czy wady klatki piersiowej — zwiększenie tolerancji wysiłku, ogólnej wydolności fizycznej oraz maksymalnego poboru tlenu [1] [8];
3. osobom z zaburzeniami lękowymi lub doświadczającym przewlekłego stresu — ukojenie lęku i łagodzenie stresu [5] [9];
4. osobom zmagającym się z problemami ze snem — poprawa jakości snu [4] [10];
5. instrumentalistom dętym — polepszenie zadęcia;
6. każdej osobie chcącej pracować nad oddechem — polepszenie układu odpornościowego, zwiększenie ilości tlenu w organizmie.

Mobilny charakter aplikacji sprawia, że trening staje się bardziej dostępny. Użytkownik może wykonywać go samodzielnie, w dowolnym miejscu, w tempie dostosowanym do swoich potrzeb. To uzasadnia potrzebę stworzenia rozwiązania, które umożliwia wygodną konfigurację wzorców oddechowych, odpowiednie prowadzenie treningu oraz przejrzystą prezentację jego przebiegu.

1.3. Cel pracy (Hanna Banasiak)

Celem pracy jest opracowanie projektu oraz implementacja wieloplatformowej aplikacji mobilnej służącej do przeprowadzania treningu oddechowego opartego na konfigurowalnych wzorcach oddechowych obejmujących wdech, retencję, wydech i regenerację, z możliwością dostosowania parametrów dźwiękowych oraz językowych (polski i angielski), a także z wbudowaną wizualizacją przebiegu treningu.

1.4. Podział prac w zespole (Jakub Romanowski)

Prace nad projektem realizowane były przy pełnym zaangażowaniu całego zespołu. Członkowie grupy dynamicznie reagowali na informacje zwrotne, sprawnie wdrażając sugestie opiekuna oraz nowe koncepcje funkcjonalne. Dzięki bieżącej eliminacji błędów i iteracyjnemu wprowadzaniu udoskonaleń zapewniono ciągłość procesu wytwórczego oraz stabilny rozwój systemu, a jasno określony zakres obowiązków przyczynił się do lepszej koordynacji zadań.

1.4.1. Hanna Banasiak

Hanna odpowiadała za projekt oraz implementację kluczowych komponentów służących do realizacji pobierania i odtwarzania sesji treningowych — `TrainingController` (rozdział 6.4.2) oraz `TrainingParser` (rozdział 6.4.1). Opracowała również wstępny projekt interfejsu użytkownika w środowisku Figma (rozdział 5.2.1), który stanowił fundament dla dalszych prac deweloperskich. W warstwie widoku zaimplementowała moduły wizualizacji treningów — `AnimatedCircle` (rozdział 6.4.3), `InstructionSlider` (rozdział 6.4.4) — walidację danych wejściowych (w dodatku A, rysunek A.26), a także stworzyła widok szczegółów treningu (w dodatku A, rozdział A.3), stanowiący element nawigacyjny między stroną główną, a odtwarzaczem.

1.4.2. Aleksandra Bujny

Do zadań Aleksandry należało opracowanie kompletnej identyfikacji wizualnej systemu — zaprojektowanie logotypu (rysunek 5.8) oraz zdefiniowanie spójnego motywu graficznego aplikacji (rysunek 5.10), który stał się obowiązującym standardem dla wszystkich modułów aplikacji. W warstwie implementacyjnej przygotowała widok strony głównej (w dodatku A, rozdział A.2), a także wzbogaciła interfejs użytkownika o animacje, zwiększające dynamikę i interaktywność aplikacji. Dodatkowo odpowiadała za bieżące utrzymanie spójności i przejrzystości aplikacji — konsekwentnie usuwała błędy, korygowała niejednoznaczności wizualne i standaryzowała elementy interfejsu, co znacząco poprawiło czytelność oraz ogólną jakość systemu.

1.4.3. Jakub Romanowski

Zadania Jakuba związane były z zaprojektowaniem i implementacją całej warstwy obsługi dźwięku w aplikacji. Kluczowym elementem jego pracy było stworzenie architektury silnika audio, opartej na komponentach `SoundManager` (rozdział 6.5.1), `SingleSoundManager` (rozdział 6.5.2) oraz `PlaylistManager` (rozdział 6.5.3). Ponadto zaprojektował fundamentalne struktury danych (rysunek 5.3) niezbędne do funkcjonowania aplikacji, wdrożył mechanizm internacjonalizacji (rozdział 6.6) oraz funkcjonalność pozwalającą użytkownikom na import własnych plików dźwiękowych (w dodatku A, rysunek A.29). Dbał również o stosowanie optymalnych rozwiązań, ponowne wykorzystanie istniejących komponentów oraz utrzymanie wysokiej jakości i przejrzystości kodu. Nadto odpowiadał za zarządzanie repozytorium na platformie GitHub, koordynację zadań oraz nadzór nad pracami zespołu.

1.4.4. Karol Zwierz

Karol był odpowiedzialny za pierwotny projekt struktury danych, implementację mechanizmu wytwarzania oraz modyfikacji treningów w sposób intuicyjny dla użytkowników (w dodatku A, rozdział A.4.1), a także za funkcjonalność ich importu i eksportu (rozdział 6.3). W obszarze przetwarzania sygnałów opracował moduł generujący dudnienia różnicowe (binaural beats w dodatku A, rozdział A.4.2). Ponadto zajął się optymalizacją wydajności poprzez wdrożenie mechanizmu ładowania zasobów z wyprzedzeniem przed uruchomieniem treningu oraz dodaniem powiązanego z nim interfejsu ekranu ładowania (w dodatku A, rysunek A.41). W warstwie interfejsu stworzył edytor list odtwarzania plików audio (w dodatku A, rysunek A.34). Wykonał również analizę kontekstu naukowego naszego rozwiązania.

1.5. Struktura rozdziałów (Hanna Banasiak)

W celu przejrzystego przedstawienia realizacji projektu i ułatwienia odbioru pracy została ona podzielona na dziewięć rozdziałów opisanych poniżej.

W rozdziale pierwszym przedstawiono wprowadzenie do tematu pracy, przedstawiono problemy i motywacje podjęcia pracy, jej cel oraz podział obowiązków w zespole.

Rozdział drugi opisuje aktualny stan wiedzy, obejmujący istniejące rozwiązania aplikacji mobilnych wspierających trening oddechowy oraz aspekty medyczne i sportowe treningu oddechowego. Przedstawiono również opisy podsumowujący ważne elementy i brakujące funkcjonalności w tych rozwiązaniach.

W rozdziale trzecim określono główne założenia projektowe, w których przedstawiono grupę docelową, korzyści płynące z projektu oraz zakładane główne funkcjonalności aplikacji. Rozdział ten definiuje również przewidywanych użytkowników oraz podstawowe wymagania wobec systemu.

Rozdział czwarty zawiera analizę wymagań, w tym modele przypadków użycia, modele i diagramy klas oraz dźwięków wykorzystywanych w projekcie.

W rozdziale piątym przedstawiono projekt rozwiązania, obejmujący architekturę systemu, logikę aplikacji, interfejs użytkownika oraz strukturę danych.

Rozdział szósty opisuje implementację aplikacji, w tym strukturę plików i modułów oraz wybrane fragmenty kodu, obejmujące najważniejsze elementy projektu. Zwrócono w nim uwagę na zastosowane technologie i narzędzia programistyczne.

W rozdziale siódmym zamieszczono instrukcję użytkownika opisującą sposób korzystania z aplikacji.

Rozdział ósmy poświęcono testowaniu, obejmującemu testy jednostkowe, integracyjne modułów, systemowe i użytkowe. Zaprezentowano w nim wyniki testów oraz ocenę poprawności działania aplikacji w różnych scenariuszach użycia.

Dziewiąty rozdział, zawiera podsumowanie pracy, omówienie osiągniętych rezultatów, napotkanych trudności oraz wnioski dotyczące projektu. Wskazano również możliwe kierunki przyszłych usprawnień i rozszerzeń funkcjonalnych systemu.

2. AKTUALNY STAN WIEDZY (Hanna Banasiak, Aleksandra Bujny, Jakub Romanowski, Karol Zwierz)

Celem niniejszego rozdziału jest przegląd wybranych aplikacji mobilnych dedykowanych treningowi oddechowemu oraz przedstawienie naukowego uzasadnienia dla ich stosowania. W kolejnych podrozdziałach opisano kontekst medyczny wykorzystania technologii mobilnych w rehabilitacji oddechowej, a następnie przeanalizowano indywidualne aplikacje, zwracając uwagę na ich specyfikę funkcjonalną, zastosowane rozwiązania technologiczne, interfejs użytkownika oraz ograniczenia.

2.1. Kontekst naukowy i skuteczność aplikacji mobilnych (Karol Zwierz)

W dobie cyfryzacji opieki zdrowotnej, aplikacje mobilne stają się istotnym narzędziem wspierającym tradycyjne metody rehabilitacji. Przeglądy systematyczne wskazują, że samodzielna rehabilitacja pulmonologiczna wspierana przez aplikacje mobilne może być skuteczną alternatywą dla rehabilitacji stacjonarnej, szczególnie w przypadku pacjentów z przewlekłą obturacyjną chorobą płuc (POChP) [6]. Aplikacje te pozwalają na przezwyciężenie barier takich jak brak dostępu do specjalistycznych ośrodków, koszty czy problemy z transportem. Wykazano, że interwencje oparte na aplikacjach mogą prowadzić do poprawy wydolności fizycznej i łagodzenia objawów chorobowych, oferując jednocześnie spersonalizowane plany treningowe, co jest kluczowe dla utrzymania zaangażowania pacjenta.

2.2. Istniejące rozwiązania wspierające trening oddechu (Jakub Romanowski)

- Wim Hof Method firmy WHM Services
- Breathwrk: Breathing Exercises firmy Breathwrk Inc.
- STAmina Apnea Trainer firmy Squarecrowd Apps SIA
- Breathe firmy Havabee

2.3. Wim Hof Method firmy WHM Services (Hanna Banasiak)

Na szczególną uwagę wśród dostępnych rozwiązań zasługuje aplikacja Wim Hof Method [11] stworzona przez firmę WHM Services. Wykorzystuje ona techniki oddechowe opracowane przez holenderskiego sportowca i trenera, Wima Hofa. Wyróżnia się ona kompleksowym podejściem, łącząc ćwiczenia z ekspozycją na zimno oraz medytacją.

2.3.1. Cel aplikacji

Celem aplikacji jest udostępnienie użytkownikom dostępu do regularnych treningów oddechowych zgodnie z założeniami metody Wim Hofa. Została zaprojektowana z myślą o osobach szukających sposobów na poprawę zdrowia psychofizycznego, zwłaszcza poprzez codzienne ćwiczenia oddechowe i ekspozycję na zimno. Ze względu na zróżnicowany poziom wiedzy technicznej oraz wiedzy z zakresu zdrowia wśród użytkowników, kluczową cechą aplikacji jest prosty i intuicyjny interfejs oraz stopniowe, prowadzące użytkownika krok po kroku wdrożenie w proces ćwiczeń.

2.3.2. *Funkcjonalności*

Aplikacja podzielona jest na trzy główne zakładki: *Results* (pol. Rezultaty), *Home* (pol. Strona główna) oraz *Community* (pol. Społeczność).

Celem zakładki *Results* (pol. Rezultaty) jest przedstawienie wszystkich postępów użytkownika. Na podstronie *Bar Chart* (pol. Wykres słupkowy) znajdują się statystyki dotyczące wykonanych *Breathing sessions* (pol. sesji oddechowych), *Cold showers* (pol. zimnych pryszniców) czy *Ice baths* (pol. kąpieli w lodzie). Poniżej znajduje się także wykres słupkowy i podsumowanie ilościowe dotyczące konkretnego rodzaju ćwiczenia, które można wybrać z dostępnej listy. Aby zarejestrować zadanie, nie jest konieczne wykonywanie go wraz z aplikacją, można to zrobić, klikając w przycisk *Log exercise* (pol. Zarejestruj ćwiczenie).

Na podstronie *Calendar* (pol. Kalendarz) dostępny jest widok miesiąca z możliwością przeglądania szczegółów poszczególnych dni. Użytkownik może edytować lub usuwać zarejestrowane aktywności, jednak dodawanie nowych jest niedostępne. Istnieje także opcja dzielenia się wynikami ze społecznością.

Podstrona *Badges* (pol. Odznaki) motywuje użytkownika do dalszej pracy poprzez system osiągnięć i nagród za regularność i stopień trudności ćwiczeń. Na rysunku 2.1 przedstawiono podstrony *Bar Chart*, *Calendar* i *Badges* zakładki *Results* aplikacji Wim Hof Method.

Rysunek 2.1. Zakładka *Results* (*Bar Chart, Calendar, Badges*) w aplikacji Wim Hof Method

Zakładka *Home* (pol. Strona główna) oferuje użytkownikowi dostęp do różnych typów ćwiczeń: *Breathing Exercises* (pol. Trening oddechowy), *Cold Exposure* (pol. Ekspozycja na zimno) — obejmującej m.in. zimne prysznice i kąpiele w lodzie — oraz trening *Power of the Mind* (pol. Siły umysłu), który w praktyce stanowi zestaw ćwiczeń fizycznych. Oprócz głównych modułów treningowych, użytkownik ma dostęp do treści dodatkowych, takich jak nagrania audio (np. internetowe publikacje Wima Hofa), sesje medytacyjne, materiały internetowe wyjaśniające zasady

działania metody oraz różnego rodzaju wyzwania. Interfejs zakładki *Home* (pol. Strona główna) przestawiono na rysunku 2.2.

Zakładka z ćwiczeniami oddechowymi umożliwia użytkownikowi skorzystanie z gotowych zestawów treningowych oraz ich edycję. Po wybraniu konkretnego ćwiczenia dostępny jest szereg opcji personalizacji, takich jak: tempo oddechu, liczba rund, liczba oddechów przed wstrzymaniem, wybór podkładu muzycznego oraz wyłączenie głosu prowadzącego.

Po uruchomieniu sesji użytkownik widzi animację w formie sześciokąta, który sygnalizuje pozostały czas do ukończenia etapu oddechu. Wyświetlane są także liczba rund i liczba oddechów. W dowolnym momencie istnieje możliwość przerwania ćwiczenia i powrotu do ekranu głównego — postęp zostaje mimo to zapisany jako ukończona sesja. Dodatkowo, podwójne kliknięcie w trakcie ćwiczenia pozwala na przejście do fazy wstrzymania oddechu. Proces treningu został przedstawiony na rysunku 2.4.

Zakładka *Community* (pol. Społeczność) umożliwia śledzenie aktywności innych użytkowników aplikacji. Aby uzyskać dostęp do postępów konkretnej osoby, należy ją wyszukać i zaobserwować. Użytkownik może również udostępniać własne wyniki za pośrednictwem spersonalizowanego profilu, w którym można zamieścić takie informacje jak imię i nazwisko, zdjęcie profilowe oraz krótki opis. W profilu widoczne są również: lista obserwowanych i obserwujących użytkowników, zdobyte odznaki oraz liczba ukończonych treningów. Intrefejs zakładki *Community* (pol. Społeczność) został przedstawiony na rysunku 2.3.

Rysunek 2.2. Zakładka *Home* w aplikacji Wim Hof Method

Rysunek 2.3. Zakładka *Community* w aplikacji Wim Hof Method

Rysunek 2.4. Trening oddechowy w aplikacji Wim Hof Method

2.3.3. Ograniczenia

Pomimo szerokiego zakresu funkcjonalności, aplikacja posiada również pewne ograniczenia, które z perspektywy użytkownika mogą mieć istotne znaczenie.

Pierwszym z nich jest dostęp do wielu funkcji wyłącznie w wersji płatnej *Premium*. Ograniczenia te dotyczą m.in. wyboru rodzaju ćwiczeń, uczestnictwa w sesjach medytacyjnych oraz podejmowania wyzwań. Choć aplikacja jest dostępna i użyteczna także w wersji podstawowej, dla bardziej zaawansowanych użytkowników może nie oferować wystarczającego wsparcia rozwojowego. Warto jednak zauważyć, że model subskrypcyjny stanowi źródło przychodu.

Drugim ograniczeniem jest brak możliwości tworzenia własnych, w pełni spersonalizowanych treningów. Aplikacja koncentruje się na ściśle określonych założeniach metody Wima Hofa, co ogranicza elastyczność i możliwość dostosowania ćwiczeń do indywidualnych potrzeb. Dostępna personalizacja ogranicza się głównie do zmiany długości ćwiczeń, liczby rund czy tempa oddychania. Dla bardziej zaawansowanych użytkowników może to stanowić istotne utrudnienie.

2.4. Breathwrk: Breathing Exercises firmy Breathwrk Inc. (Aleksandra Bujny)

Breathwrk [12] to aplikacja, która w 2022 roku zwyciężyła w kategorii *Najlepsza aplikacja do rozwoju osobistego* w *Google Google Play's Best of 2022*. Jest reklamowana jako aplikacja numer jeden do ćwiczeń oddechowych. Posiada wysoką ocenę zarówno w *Sklepie Play* - 4.1, jak i w *App Store* - 4.8. Według twórców posiada około dwa miliony użytkowników.

2.4.1. Cel aplikacji

Aplikacja skupia się na pomocy użytkownikowi poprzez trening oddechowy w czterech głównych obszarach: śnie, stresie, koncentracji oraz energii. Oferuje setki ćwiczeń oraz zajęć, czyli sesji z doświadczonymi instruktorami, którzy przeprowadzają użytkownika przez starannie do-

braną serię ćwiczeń. Breathwrk ma także służyć jako pomoc w budowaniu nawyku regularnych treningów oddechowych, dzięki przypomnieniom według ustalonych harmonogramów. Elementem wyróżniającym są także codziennie dodawane, nowe zajęcia immersyjne oferujące ciekawe wrażenia wizualne. Interesującym dodatkiem są również wibracje telefonu podczas treningu, jednak ustawienie to nie jest dostępne na wszystkich urządzeniach. Możliwy jest dostęp offline do aplikacji, a jej językiem głównym jest angielski.

2.4.2. Funkcjonalności

Aplikacja składa się z czterech głównych zakładek dostępnych na pasku nawigacyjnym na dole aplikacji: *Now* (pol. Teraz), *Explore* (pol. Odkrywaj), *Schedule* (pol. Zaplanuj), *Me* (pol. Ja).

W prawym, górnym rogu znajduje się okrągła ikona ze zdjęciem profilowym lub wybranym z biblioteki zdjęciem zwierzęcia, liczbą oznaczającą poziom rankingowy oraz postęp w osiągnięciu kolejnego poziomu rankingowego. Poziom rankingowy można zwiększać poprzez wykonywanie ćwiczeń. Po kliknięciu na ikonę pokazuje się okno *Shortcuts* (pol. Skróty) z dwoma zakładkami. W jednej z nich znajdują się treningi oznaczone jako ulubione, a w drugiej ostatnio wykonane treningi. Pokazuje się także ikona ołówka, po kliknięciu którego możliwa jest edycja zdjęcia profilu. Zakładki te przedstawione zostały na rysunku 2.5.

Rysunek 2.5. Okno *Shortcuts* w aplikacji Breathwrk

Główna zakładka aplikacji to *Now*. Na samej górze widoczny jest streak, a także zakładka *favourites* (pol. ulubione). Środkową część zajmuje karuzela z treningami oddechowymi, które są obecnie trendujące lub zostały wybrane dla użytkownika. W prawym górnym rogu dostępna jest opcja dodania treningu do ulubionych poprzez kliknięcie ikony serca, a także ikona informacyjna, która otwiera okno z opisem treningu. Dostępny jest ogólny opis treningu, propozycje użycia treningu oraz opis biologicznych procesów, które aktywuje trening. U góry okna umiesz-

czone zostały piktogramy symbolizujące w jaki sposób wykonywać wdechy i wydechy oraz czas trwania poszczególnych etapów. Na rysunku 2.6 przedstawiono widok opisu trenigu.

Rysunek 2.6. Podzakładki strony opisu treningu

Po uruchomieniu treningu pokazuje się domyślna animacja - koło. Po kliknięciu na środek ekranu pokazuje się czas do zakończenia treningu, ikona informacji,odnosząca do opisu treningu, przycisk pauzy oraz rezygnacji z treningu. Możliwe jest też dodanie trenigu do ulubionych i jego udostępnienie. Dostępne są także opcje konfiguracji treningu - wybór animacji (koło, małpka lub linia), dodanie nawyku, włączenie instrukcji głosowych, włączenie i wybór dźwięku w tle, a także wibracje telefonu. Widok tej strony przedstawiony został na rysunku 2.7. Po zakończeniu treningu pokazuje się okno ze statystykami oraz gratulacjami. Pokazuje się także przycisk do ponowienia treningu z wybranymi wcześniej ustawieniami.

Rysunek 2.7. Strona treningu w aplikacji Breathwrk

Rysunek 2.8. Przykładowe treningi immersyjne w aplikacji Breathwrk

Na dole zakładki znajduje się podsekcja *Daily Classes* (pol. Codzienne Ćwiczenia) z zestawem 3 ćwiczeń immersyjnych, które zmieniają się co 24 godziny. Ćwiczenia te charakteryzują się nietypowymi wrażeniami wizualnymi - efekt promieniowania czy animacja wody w basenie. Przykładowe ćwiczenie przedstawione zostały na rysunku 2.8.

Explore to zakładka będąca biblioteką treningów, ćwiczeń, wyzwań, nawyków (opcja dla użytkowników *Breathwrk Premium*) oraz testów. Dostępne są także zakładki, po których można szybko filtrować treść: *All*, *Calming*, *Nighttimel*, *Energizing*, *Perform* oraz *Health*. Rysunek 2.9 przedstawia wygląd tej zakładki.

Rysunek 2.9. Zakładka *Discover* w aplikacji Breathwrk)

Rysunek 2.10. Zakładka *Me*

Celem zakładki *Schedule* jest zachęcenie użytkownika do systematycznego używania aplikacji. Możliwe jest ustawienie pojedynczego przypomnienia na trening lub regularnych przypomnień w wybrane dni, tak aby stworzyć nawyk.

Zakładka *Me* składa się z trzech sekcji: statystyk użytkownika, testów udoskanalających oddech oraz tabeli najlepszych użytkowników, ocenianych na podstawie miesięcznego lub tygodniowego czasu spędzonego na aktywnościach oddechowych w aplikacji. Zakładka przedstawiona została na rysunku 2.10

2.4.3. Ograniczenia

Aplikacja jest przedstawiana jako umożliwiająca pełną personalizację ćwiczeń oddechowych, mimo to zawiera ograniczenia.

W aplikacji brakuje przede wszystkim możliwości wyboru długości trwania poszczególnych kroków. Możliwy jest jedynie wybór czasu trwania całego treningu, jednak jest on także ograniczony do trzech predefiniowanych wyborów dla mniejszej ilości interwałów. Dla większej liczby interwałów można je wybrać z dokładnością do jednego interwału, jest to natomiast funkcja dostępna jedynie w płatnym planie *Breathwrk Premium*

Kolejnym ograniczeniem jest brak możliwości tworzenia własnych treningów. Do wyboru pozostają wyłącznie treningi oferowane przez aplikację. Dla użytkowników wersji *Breathwrk Premium* wybór gotowych treningów i zajęć jest szeroki, natomiast dla pozostałych użytkowników jest mocno ograniczony - po jednym lub nawet braku treningu spośród kilku z każdej z kategorii.

2.5. STAmina Apnea Trainer firmy Squarecrowd Apps SIA (Karol Zwierz)

Kolejnym ważnym rozwiązaniem jest aplikacja STAmina Apnea Trainer [13], stworzona przez firmę Squarecrowd Apps SIA. To narzędzie mobilne wspomagające trening wstrzymywania oddechu, szczególnie przydatne dla freediverów, sportowców oraz osób praktykujących techniki oddechowe i relaksacyjne.

2.5.1. Cel aplikacji

STAmina Apnea Trainer to narzędzie przeznaczone przede wszystkim dla osób ćwiczących freediving, łowiectwo podwodne oraz inne dyscypliny wodne, w których kluczowe znaczenie ma umiejętność długiego wstrzymywania oddechu. Głównym założeniem aplikacji jest zapewnienie kompleksowego wsparcia w treningu *static apnea* (pol. bezdech statyczny), wykorzystującego różnorodne tablice treningowe (między innymi ukierunkowane na tolerancję na niedotlenienie - O_2, nadmiar dwutlenku węgla - CO_2, a także kombinacje, techniki relaksacyjne i możliwość tworzenia własnych schematów).

2.5.2. Funkcjonalności

Aplikacja udostępnia pięć gotowych szablonów treningowych, które pozwalają systematycznie rozwijać poszczególne aspekty wstrzymywania oddechu. Pierwszy z nich koncentruje się na niedoborze O_2, czyli stopniowym wydłużaniu czasu wstrzymania przy stałym czasie odpoczynku. Drugi skupia się na tolerancji CO_2, w którym użytkownik zmniejsza przerwy między kolejnymi sesjami wstrzymania oddechu, by przyzwyczaić organizm do wyższego poziomu CO_2. Kolejny schemat, nazwany *Wonka*, wprowadza pauzę po pierwszym odczuwalnym skurczu przepony, co pomaga uczyć się właściwej techniki. Schemat *Mix* (pol. Mieszany) łączy oba podejścia (O_2/CO_2), zapewniając wszechstronne podejście do treningu, natomiast *Pranayama* oferuje techniki oddechowe o charakterze relaksacyjnym, przydatne zarówno przed, jak i po głównej części treningu. Na rysunku 2.11 przedstawiono gotowe schematy dostępne w aplikacji.

Poza gotowymi programami każdy użytkownik może stworzyć trening *Custom* (pol. własny) z pełną konfigurowalnością, dobierając liczbę powtórzeń, czasy wstrzymywania i przerw według indywidualnych potrzeb. Na podstawie dotychczasowego *Personal Best* (pol. rekordu) aplikacja automatycznie generuje sesje w trzech stopniach zaawansowania - *Easy* (pol. łatwy), *Normal* (pol. normalny), *Hard* (pol. trudny), co ułatwia progresję i dostosowanie wysiłku do aktualnych możliwości. Tworzenie treningu przedstawiono na rysunku 2.12.

Rysunek 2.11. Gotowe schematy w aplikacji STAmina Apnea Trainer

Aby wspomóc systematyczne korzystanie, wbudowany system powiadomień przypomina o zaplanowanych treningach. Wszystkie sesje są zapisywane, a użytkownik może obserwować swoje postępy dzięki szczegółowym przeglądom rekordów i trendów w czasie. Dodatkowym wsparciem jest nawigacja głosowa prowadzona przez nagrania profesjonalnych lektorów w wersjach męskiej i żeńskiej, dostępna w kilku językach (m.in. angielskim, francuskim, niemieckim, włoskim i rosyjskim), co pozwala skupić się wyłącznie na ćwiczeniu, bez konieczności patrzenia na ekran. Integracja z Apple Health umożliwia rejestrowanie tętna i poziomu SpO_2 za pomocą Apple Watch lub innych urządzeń Bluetooth, co dostarcza cennych danych biometrycznych podczas treningów. Przedstawiono to na rysunku 2.13.

Synchronizacja z chmurą gwarantuje *backup* (pol. kopia zapasowa) i przywracanie danych pomiędzy urządzeniami, co jest istotne dla osób korzystających z aplikacji na różnych sprzętach. Aplikacja oferuje także możliwość śledzenia momentu wystąpienia skurczów oddechowych, co pomaga w lepszym zrozumieniu własnych granic i rozwoju techniki. Dla lepszego komfortu dostępne są wibracyjne alerty oraz *square breath* (pol. tryb oddechu pudełkowego), wspierające relaksację i stabilizację oddechu.

Rysunek 2.12. Tworzenia treningu w aplikacji STAmina Apnea Trainer

Rysunek 2.13. Wykresy poziomu SpO_2 w aplikacji STAmina Apnea Trainer

Zakładka z ogólnymi pomocami dotyczącymi treningów oddechowych oraz freedivingu dostarcza wiedzy teoretycznej i praktycznych wskazówek, co pomaga użytkownikom zarówno początkującym, jak i bardziej zaawansowanym.

2.5.3. Ograniczenia

Mimo bogatego zestawu funkcji, aplikacja ma też swoje wady. Interfejs jest już nieco przestarzały, co może obniżać komfort użytkowania i sprawiać, że poruszanie się po aplikacji wydaje się mniej intuicyjne w porównaniu ze współczesnymi standardami. Brak wbudowanego przewodnika lub samouczka powoduje, że nowi użytkownicy mogą mieć trudności z rozpoczęciem treningów. Pomocne mogłoby okazać się interaktywne wskazówki dla początkujących czy wprowadzenie krok po kroku.

2.6. Breathe firmy Havabee (Jakub Romanowski)

Kolejną aplikacją wartą uwagi jest Breathe [14] firmy Havabee. Wyróżnia się spośród innych swoim minimalizmem i łatwością obsługi.

2.6.1. Cel aplikacji

Celem aplikacji jest umożliwienie użytkownikom, niezależnie od wieku i zaawansowania technologicznego, wykonywania treningów oddechowych. Została ona zaprojektowana z myślą o osobach niewymagających wiele, co potwierdzają proste treningi dostarczone razem z aplikacją.

2.6.2. Funkcjonalności

Aplikacje w swoim przejrzystym interfejsie zawiera jedynie trzy podstrony: *Domyślnie* (Strona główna), *Własne* (Strona konfiguracji treningów) i *Postęp* (Strona ze statystykami).

Podstrona *Domyślnie* zawiera cztery zdefiniowane przez twórców aplikacji treningi, takie jak: *Oddychanie równoważne*, *Oddychanie pudełkowe*, *Oddychanie sposobem 4-7-8* i *Test wstrzymania oddechu*. Aby dowiedzieć się więcej o danym treningu, wystarczy kliknąć na ikonę informacji.

Możliwa jest również zmiana trwania schematów poprzez kliknięcie w przycisk *Zmień czas trwania*. Gotowe schematy zostały przedstawione na rysunku 2.14.

Rysunek 2.14. Schematy dostępne w aplikacji Breathe

Rysunek 2.15. Trening w aplikacji Breathe

Po wybraniu interesującego nas schematu przenoszeni jesteśmy do strony treningu. Aby przygotować użytkownika do ćwiczeń, w pierwszej kolejności odbywa się krótka sesja pozwalająca skupić się na własnym oddechu i przejść w stan świadomej pracy oddechowej. Po niej przechodzimy do właściwego ćwiczenia, które prowadzone jest zarówno wizualnie, jak i w formie audio. Liczba odbytych cykli wyświetlana jest w postaci zapełniającej się obwódki kółka, jednak brak bezpośredniego wyświetlania wartości liczbowych. Dostępne jest również ustawienie poziomu dźwięku, jednakże brakuje możliwości zatrzymania treningu. Można jedynie powrócić do okna startowego, w dodatku bez ostrzeżenia użytkownika o natychmiastowym zakończeniu sesji. Trening został przedstawiony na rysunku 2.15.

Celem kolejnej zakładki jest umożliwienie użytkownikom tworzenie własnych wzorów oddychania. Po kliknięciu w przycisk plusa ("+"), zostajemy przeniesieni na stronę konfiguracji treningu. Możliwe jest nadanie treningowi nazwy, a także wybranie czasu trwania każdej z faz oddechu (Wdech, Wstrzymanie, Wydech, Wstrzymanie) i liczby cykli w przedziale od jednego do dwustu pięćdziesięciu. Na rysunku 2.16 przedstawiono tę zakładkę.

Na ostatniej stronie możemy podejrzeć czas, jaki spędziliśmy w aplikacji oraz liczbę odbytych sesji w danym dniu, tygodniu, miesiącu i roku. Poniżej ogólnych statystyk znajduje się także osobna sekcja do podglądu odbytych treningów wstrzymywania oddechu, na której można zaobserwować przeciętny i najdłuższy czas trwania takiego oddechu w tych samych ramach czasowych.

Rysunek 2.16. Tworzenie własnych treningów w aplikacji Breathe

2.6.3. Ograniczenia

Wspomniana aplikacja, mimo swojej prostej szaty graficznej oraz oferowanych funkcjonalności, nie jest w stanie sprostać oczekiwaniom wszystkich użytkowników. Każdy z dostępnych wzorców treningowych umożliwia jedynie powtarzanie określonego cyklu oddechowego z góry ustaloną liczbę razy. Stanowi to istotne ograniczenie dla osób, które w swoich treningach wykorzystują zróżnicowane cykle oddechowe lub takie, które składają się z mniejszej liczby faz. Dodatkowym mankamentem jest brak możliwości ustawienia czasu wstrzymania oddechu na zero sekund lub bardzo krótkich wdechów, co skutkuje nieprawidłowym działaniem dźwięku towarzyszącego ćwiczeniu. Koniecznie należy wspomnieć o braku możliwości zatrzymania treningu i mało przejrzystego wyświetlania liczby cykli podczas treningu.

3. ZAŁOŻENIA PROJEKTOWE (Aleksandra Bujny, Karol Zwierz)

3.1. Dostarczane korzyści (Karol Zwierz)

Celem projektu jest stworzenie intuicyjnej i funkcjonalnej aplikacji mobilnej, umożliwiającej użytkownikom skuteczny trening oddechowy, która ma pomagać w poprawie kontroli oddechu, redukcji stresu oraz zwiększeniu świadomości oddechowej. W ramach prac zostanie zaprojektowany i zaimplementowany interfejs użytkownika, który umożliwi łatwe zarządzanie sesjami treningu oddechowego. Kluczowym aspektem aplikacji będzie szeroka personalizacja poprzez wybór parametrów takich jak typ fazy (wdech, retencja, wydech, regeneracja), długość fazy, inkrementacja czy powtórzenia. Służyć ma ona jak najlepszemu dopasowaniu treningów do wizji użytkowników, zapewniając dużą elastyczność i swobodę. Oferując przykładowe treningi, jednocześnie zaspokaja również potrzeby tych osób, które wolą skorzystać z gotowych, prostszych treningów oddechowych. Projekt zakłada również opracowanie mechanizmów interaktywnych, takich jak wizualizacje wspomagające kontrolę oddechu. Ponadto oferować będzie zarówno język polski, jak i angielski, co umożliwi dostępność wśród użytkowników w Polsce i za granicą.

3.2. Główne funkcje aplikacji (Karol Zwierz)

Główne funkcjonalności aplikacji ReSpire zostały zaprojektowane tak, aby zapewnić użytkownikowi pełną kontrolę nad procesem treningowym oraz wysoką jakość doświadczenia audiowizualnego. Do kluczowych punktów realizacji systemu należą:

- **Zaawansowany edytor treningów**: Narzędzie umożliwiające tworzenie złożonych, wieloetapowych i wielofazowych sesji treningowych. System pozwala na definiowanie konfigurowalnych etapów składających się z faz różnego typu. Kluczową funkcjonalnością jest obsługa iteracji sekwencji objętych danym etapem z inkrementacja czasu trwania faz w kolejnych cyklach, co umożliwia realizację treningów progresywnych (np. stopniowe wydłużanie faz).
- **Wielopoziomowy system audio**: Złożony mechanizm konfigurowania dźwięku, pozwalający na przypisywanie tła akustycznego, wskazówek audio, etc. na trzech poziomach hierarchii: globalnym (cały trening), etapu oraz fazy (unikalne dźwięki dla wdechu, wydechu, wstrzymania, regeneracji). System obsługuje komunikaty głosowe zrealizowane za pomocą generowania audio na podstawie tekstu. Umożliwiałoby to wykonywanie treningu bez patrzenia w ekran telefonu.
- **Generator dudnień różnicowych**: Funkcjonalność emitująca w czasie rzeczywistym dźwięki o różnej częstotliwości dla lewego i prawego kanału audio. Wywołuje to odczucie słyszenia jedynie małej różnicy częstotliwości między częstotliwościami kanałów. Generowanie odbywa się dynamicznie na podstawie zadanych parametrów częstotliwości obu kanałów, bez konieczności dostarczania gotowych plików.
- **Silnik wykonywania treningu**: Moduł odpowiedzialny za odtworzenie sesji w czasie rzeczywistym. Precyzyjnie synchronizuje timer, animacje interfejsu oraz warstwę audio, zapewniając płynne działanie nawet przy skomplikowanych strukturach treningowych. Zarządza również specjalnymi fazami przygotowania oraz zakończenia treningu.

- **Persystencja i udostępnianie treningów**: Możliwość zapisu stworzonych treningów i ustawień w lokalnej, wydajnej bazie danych na urządzeniu, co pozwala na pełną funkcjonalność w trybie offline. Dodatkowo system umożliwia import i eksport zrzutów danych treningów w przenośnym formacie, co ułatwia dzielenie się nimi między użytkownikami.

3.3. Przewidziani użytkownicy (Karol Zwierz)

Ze względu na specyfikę aplikacji przewidujemy podział użytkowników na dwie główne grupy:

- Użytkownicy początkujący - osoby o bardzo zróżnicowanym profilu będące nowicjuszami w dziedzinie treningów oddechowych. Wymagają oni jedynie możliwości zaimportowania oraz odtworzenia gotowego treningu. Nie chcą oni nadmiernie ingerować w strukturę swoich treningów.
- Użytkownicy zaawansowani - osoby zainteresowane szczegółową personalizacją ze względu na swoją głęboką znajomość praktyk treningów oddechowych. Wymagają oni wysoce wyszukanych możliwości konfiguracji każdego aspektu sekwencji treningowej.

3.4. Systemy współpracujące (Karol Zwierz)

Rozważana jest implementacja zaawansowanego systemu sprzężenia zwrotnego dla użytkownika wykonującego analizę wykonywania treningu w trakcie jego odtwarzania. System ten nagrywałby krótkie fragmenty dźwięku z mikrofonów telefonu lub podłączonego zestawu słuchawkowego (konfiguracja preferowana dla lepszej izolacji dźwięku zbieranego oraz dźwięku emitowanego przez aplikację). Następnie fragmenty te byłyby przetwarzane równolegle względem głównych procesów aplikacji. Polegałoby to na transformacji fragmentów dźwiękowych na odpowiadające im mel-spektrogramy a następnie przekazanie ich jako wartości wejściowe do wytrenowanego predykcyjnego modelu uczenia maszynowego w celu uzyskania danych zwrotnych oceniających precyzję wykonania przez użytkownika zadanego treningu oddechowego.

4. ANALIZA WYMAGAŃ (Hanna Banasiak, Aleksandra Bujny)

W niniejszym rozdziale przedstawione zostały wymagania projektowanej aplikacji. Użyto w tym celu trzech modeli - przypadków użycia, klas oraz sterowania dźwiękami.

4.1. Model przypadków użycia (Aleksandra Bujny)

Celem modelu jest określenie głównych funkcji aplikacji, aktorów oraz relacji między nimi. W jego zakres wchodzą wszystkie funkcje dostępne dla użytkownika końcowego. Wyróżnionych zostało czterech aktorów, w tym jeden ożwiony: osoba korzystająca z aplikacji oraz trzech nieożywionych: lokalna baza danych, syntezator mowy oraz odtwarzacz dźwięków. Ze względu na czytelność, diagram przypadków użycia został podzielony na trzy mniejsze części, gdzie zgrupowane są przypadki ze sobą powiązane.

Pierwszy z diagramów został przedstawiony na rysunku 4.1. Zawiera podstawowe przypadki użycia, takie jak: wyświetlanie strony treningu (opis przypadku w tabeli 4.1) lub ustawień, usuwanie (tabela 4.2), eksport (tabela 4.4), import (tabela 4.3), edycję (tabela 4.5), dodawanie (tabela 4.6), odtwarzanie (tabela 4.7) lub wyświetlanie listy albo szczegółów treningów oddechowych. Relacje include (pol. zawierania) i extend (pol. rozszerzenia) powiązane z przypadkami dodawania, edycji oraz odtwarzania zostały przedstawione na kolejnych, odrębnych diagramach.

Rysunek 4.1. Główne przypadki użycia dla aplikacji ReSpire

Warunki początkowe	Istnieje co najmniej jeden trening, wykonano przypadek użycia *Wyświetlenie listy treningów* (opisany w tabeli4.1).
Przebieg	1. Użytkownik zgłasza żądanie wyświetlenia strony wybranego treningu. 2. System wyświetla stronę zgodnie z żądaniem. 3. Użytkownik może zgłosić żądanie edycji (opisanej w tabeli 4.5), usunięcia (tabela 4.2), eksportu (tabela 4.4) lub odtworzenia treningu (tabela 4.7).
Przebieg alternatywny	brak
Warunki końcowe	brak

Tabela 4.1. Wyświetlenie strony treningu

Warunki początkowe	Istnieje co najmniej jeden trening, wykonano przypadek użycia *Wyświetlenie strony treningu* (opisany w tabeli 4.1).
Przebieg	1. Użytkownik zgłasza żądanie usunięcia treningu. 2. System wyświetla okno z prośbą o potwierdzenie akcji. 3. Użytkownik potwierdza żądanie. 4. Użytkownik zgłasza żądanie opuszczenia strony edycji. 5. System usuwa trening z bazy.
Przebieg alternatywny	3a. Użytkownik nie zgłasza żadnego żądania modyfikacji treningu. 3a1. Użytkownik zgłasza żądanie opuszczenia strony edycji. 3a2. Przypadek użycia zostaje przerwany.
Warunki końcowe	Trening jest poprawnie usunięty i nie wyświetla się na liście.

Tabela 4.2. Usuwanie treningu

Warunki początkowe	Użytkownik znajduje się na stronie głównej.
Przebieg	1. Użytkownik zgłasza żądanie importu treningu. 2. System wyświetla okno z eksploratorem plików. 3. Użytkownik wybiera plik w formacie JSON do importu.
Przebieg alternatywny	3a. Użytkownik wybiera plik o błędnym rozszerzeniu, złej strukturze lub przerywa akcję. 3a1. Przypadek użycia zostaje przerwany, system wyświetla odpowiedni komunikat.
Warunki końcowe	Trening lub treningi zostają poprawnie zaimportowane i pojawiają się na liście.

Tabela 4.3. Import treningu

Warunki początkowe	Istnieje co najmniej jeden trening, wykonano przypadek użycia *Wyświetlenie strony treningu* (opisany w tabeli 4.1) lub otwarta jest strona główna.
Przebieg	1. Użytkownik zgłasza żądanie eksportu treningu (lub treningów). 2. System wyświetla okno eksploratora plików. 3. Użytkownik wybiera tytuł pliku oraz jego lokalizację i potwierdza żądanie zapisu. 4. System wyświetla potwierdzenie zapisu.
Przebieg alternatywny	3a. Użytkownik przerywa akcję zapisu. 3a1. Przypadek użycia zostaje przerwany, system wyświetla odpowiedni komunikat.
Warunki końcowe	Trening w formacie JSON zostaje zapisany na urządzeniu użytkownika.

Tabela 4.4. Eksport treningu

Drugi diagram został przedstawiony na rysunku 4.2. Przedstawia wspomniane wcześniej: edycję treningu (opisaną w tabeli 4.5) oraz dodawnie treningu (tabela 4.6) wraz z powiązanymi z nim przypadkami.

Rysunek 4.2. Przypadki użycia dla aplikacji ReSpire - edycja treningu

Warunki początkowe	Istnieje co najmniej jeden trening, widok ze szczegółami treningu jest otwarty.
Przebieg	1. Użytkownik zgłasza żądanie edycji treningu. 2. System przenosi użytkownika na stronę edytora. 3. Użytkownik może zgłosić żądania: edycji opisu, tytułu, dźwięków w tle lub zmiany, dodania czy usunięcia etapu (wraz z fazami). 4. Użytkownik zgłasza żądanie opuszczenia strony edytora. 5. System dokonuje aktualizacji treningu o wprowadzone zmiany
Przebieg alternatywny	3a. Użytkownik nie zgłasza żadnego żądania modyfikacji treningu. 3a1. Użytkownik zgłasza żądanie opuszczenia strony edytora. 3a2. Przypadek użycia zostaje przerwany.
Warunki końcowe	Trening jest poprawnie zaktualizowany o wprowadzone zmiany.

Tabela 4.5. Edycja treningu

Warunki początkowe	Istnieje co najmniej jeden trening,wykonano przypadek użycia *Wyświetlenie strony treningu* (opisany w tabeli 4.1).
Przebieg	1. Użytkownik zgłasza żądanie dodania treningu. 2. System przenosi użytkownika na stronę edytora. 3. Użytkownik zgłasza żądania dodania etapów wraz z fazami. 4. Użytkownik może także zgłosić żądania: edycji opisu, tytułu, dźwięków w tle lub usunięcia etapów/faz. 5. Użytkownik zgłasza żądanie opuszczenia strony edytora. 6. System dokonuje aktualizacji listy treningów.
Przebieg alternatywny	3a. Użytkownik nie zgłasza żądania dodania żadnego elementu treningu. 3a1. Użytkownik zgłasza żądanie opuszczenia strony edytora. 3a2. System wyświetla komunikat o pustym treningu. 3a3. Użytkownik potwierdza chęć przerwania dodawania treningu i przypadek użycia zostaje przerwany lub wraca do punktu 3. 3b. Użytkownik zgłasza żądanie dodania etapów bez dodania faz. 3b1. System wyświetla komunikat o niepełnym treningu. 3b2. Użytkownik uzupełnia trening o brakujące elementy, a następnie wraca do punktu 4 lub rezygnuje z dodania treningu i przypadek użycia zostaje przerwany.
Warunki końcowe	Trening został dodany do listy.

Tabela 4.6. Dodawanie treningu

Ostatni, trzeci diagram został przedstawiony na rysunku 4.3.

Rysunek 4.3. Przypadki użycia dla aplikacji ReSpire - odtwarzanie treningu

Warunki początkowe	Istnieje co najmniej jeden trening, wykonano przypadek użycia *Wyświetlenie strony treningu* (opisany w tabeli 4.1).
Przebieg	1. Użytkownik zgłasza żądanie odtworzenia treningu. 2. System odtwarza trening z wizualizacją. 3. Jeśli zostały ustwione odpowiednie parametry w edytorze syntezator mowy może zażądać odtworzenia instrukcji głosowych, a odtwarzacz dźwięków - odtwarzania dźwięków lub wybranych częstotliwości. 4. Użytkownik może zażądać wstrzymania treningu. 5. System zakańcza odtwarzanie treningu po upływie określonego czasu.
Przebieg alternatywny	2a. Użytkownik zgłasza żądanie opuszczenia treningu. 3a1. System wyświetla okno z zapytaniem o potwierdzenie akcji. 3a2. Użytkownik potwierdza żądanie opuszczenia treningu i przypadek użycia zostaje przerwany lub analuje akcje i wraca do punktu 2.
Warunki końcowe	Następuje powrót na stronę szczegółów treningu.

Tabela 4.7. Otwarzanie treningu

4.2. Model sterowania dźwiękami (Hanna Banasiak)

W celu zobrazowania logiki sterowania warstwą dźwiękową i dostępnych strategii organizacji dźwięków w opracowanej aplikacji treningu oddechowego przygotowano dwa diagramy — sterowania muzyką w tle i dźwiękami w etapie. Przyjęto jednolitą konwencję graficzną, w której prostokąty z przerywaną ramką oznaczają ścieżki dźwiękowe, natomiast prostokąty z ciągłą ramką reprezentują struktury logiczne aplikacji, takie jak etapy, cykle i fazy oddechowe. Identyczny kolor wypełnienia wskazuje na tę samą ścieżkę dźwiękową, a strzałka pozioma symbolizuje oś czasu. Warto wspomnieć, iż każdy dźwięk przedstawiony w poniższych diagramach może również zostać wyłączony, jeśli użytkownik preferuje wykonywać ćwiczenia w ciszy.

Diagram sterowania muzyką w tle przedstawiony został na rysunku 4.4. Prezentuje on cztery dostępne tryby konfiguracji warstwy muzycznej całej sesji treningowej. We wszystkich trybach występują muzyka rozpoczęcia i muzyka zakończenia. Pierwszy tryb przewiduje jedną, wspólną ścieżkę muzyczną odtwarzaną przez cały czas trwania głównej części treningu. Drugi tryb umożliwia przypisanie osobnej muzyki do każdego etapu treningu, dzięki czemu zmiana utworu wyraźnie sygnalizuje przejście do kolejnej sekwencji faz. Trzeci tryb uzależnia wybór muzyki wyłącznie od rodzaju fazy oddechowej (wdech, retencja, wydech, regeneracja), co oznacza, że ta sama faza w różnych etapach korzysta z identycznej ścieżki. Najbardziej elastyczny, czwarty tryb łączy zależności opcji drugiej i trzeciej — muzyka zależy jednocześnie od etapu i rodzaju fazy, pozwalając na przykład na zróżnicowanie charakteru wszystkich wdechów w etapie pierwszym od wdechów w etapie drugim.

Rysunek 4.4. Diagram sterowania muzyką w tle

Diagram sterowania krótkimi dźwiękami w etapie przedstawiony został na rysunku 4.5. Ilustruje on sposób wyzwalania dźwięków sygnalizujących zmiany oraz odliczanie czasu w fazach. Zaprojektowano dwa wzajemnie wykluczające się warianty — dźwięk zmiany etapu, odtwarzany wyłącznie na początku pierwszego cyklu nowego etapu, oraz dźwięk zmiany cyklu, pojawiający

się na starcie każdego kolejnego cyklu w obrębie tego samego etapu. Po sygnale zmiany następuje właściwy cykl oddechowy, w którym na początku każdej fazy odtwarzany jest krótki sygnał informujący o jej rozpoczęciu. Następnie w fazach odtwarzana jest sekwencja odliczająca, reprezentowana przez bloki "3", "2"i "1". Rozwiązanie to zapewnia precyzyjne prowadzenie użytkownika w czasie rzeczywistym.

Rysunek 4.5. Diagram sterowania dźwiękami w etapie

5. PROJEKT ROZWIĄZANIA (Hanna Banasiak, Aleksandra Bujny, Jakub Romanowski)

5.1. Projekt architektury systemu (Aleksandra Bujny)

Przy projektowaniu architektury aplikacji wzięte pod uwagę zostały dobre praktyki i koncepty proponowane w dokumentacji Fluttera [15], a także wzorce projektowe. Celem było stworzenie struktury, która będzie łatwa w utrzymaniu i rozwoju, ale równocześnie stabilna i wydajna.

5.1.1. Architektura warstwowa

Aplikacja została zaprojektowana zgodnie z architekturą warstwową, zgodnie z wytycznymi Fluttera. Składa się z trzech głównych warstw: prezentacji, logiki oraz danych. Zostały one przedstawione na rysunku 5.1.

Rysunek 5.1. Architektura warstwowa aplikacji ReSpire

Warstwy komunikują się jedynie z warstwami bezpośrednio nad lub pod nimi - oznacza to, że warstwa prezentacji nie powinna wiedzieć o istnieniu warstwy danych i na odwrót.

Warstwy aplikacji i ich rola:

- *Warstwa prezentacji* - wyświetla dane otrzymane poprzez warstwę logiki użytkownikowi oraz obsługuje żądania użytkownika. W ReSpire jest reprezentowana poprzez komponenty oraz składające się z nich strony.
- *Warstwa logiki* - implementuje logikę biznesową i jest pośrednikiem między dwoma pozostałymi warstwami. Klasy stanowiące tę warstwę znajdują się w folderze *service/* (pol. serwis) i obsługują różne funkcjonalności w aplikacji, takie jak na przykład tłumaczenie aplikacji, zarządzanie i odtwarzanie dźwięków, kontrola przebiegu treningu czy zamiana tekstu na mowę.
- *Warstwa danych* - odpowiada za interakcje z bazą danych i przekazywanie danych do warstwy logiki. W aplikacji dane przechowywane są w lekkiej bazie *Hive* i z niej są pobierane.

5.1.2. Architektura monolityczna

ReSpire to aplikacja monolityczna - nie posiada podziału na mikroserwisy ani inne niezależne usługi. Wszystkie funkcje i komponenty są zintegrowane w jednym kodzie źródłowym. Ułatwia to zarządzanie projektem i redukuje niedgogodności związane z komunikacją między usługami, z którymi muszą się mierzyć systemy rozproszone. Aplikacja działa lokalnie, nie komunikuje się z zewnętrznym serwerem - posiada lekką, wbudowaną bazę danych. Nie jest przeznaczona dla bardzo dużego grona odbiorców, więc nie musi cechować się wyjątkową wydajnością. Cała

logika biznesowa, interfejs użytkownika i zarządzanie danymi są zawarte w jednej aplikacji. Do zalet monolitu należą szybkość implementacji, łatwość zaprojektowania oraz zrozumienia. Architektura ta może natomiast powodować potencjalne problemy ze skalowalnością i elastycznością aplikacji w przyszłości. Obecne wymagania sprawiaja jednak, że architektura monolityczna jest odpowiednia dla tego projektu i jej zalety przewyższają wady.

5.1.3. Modularny podział aplikacji

System oparty jest na modularnym podziale, czyli składa się z mniejszych komponentów. Dzięki temu możliwe jest niezależne rozwijanie, testowanie, zmienianie i utrzymywanie poszczególnych modułów, a także ich wielokrotne użycie. Pozwala to również uporządkować kod i architekturę systemu. Poszczególne elementy warstwy prezentacji zostały wydzielone z widoków - na przykład strona treningu oddechowego posiada karuzelę z instrukcjami oraz animowane koło jako osobne komponenty. Łączy się to z zasadą *Separation of concerns* (pol. podział zagadnień), gdzie każdy moduł skupia się na innym, określonym obszarze funkcjonalnym. Zmiany w jedym komponencie nie wpływają na pozostałe, co ułatwia zarządzanie nimi i wprowadzanie zmian.

5.1.4. Model obiektów stron

Widoki aplikacji zostały zamodelowane zgodnie z *Page Object Model* (pol. model obiektów stron), który określa, że strony powinny być osobnymi klasami. Dostępne jest pięć widoków: strona główna, szczegółów treningu, treningu oddechowego, ustawień oraz edytora. Każdy z nich jest oddzielną klasą umieszczoną w osobnym pliku (w folderze *pages/*). Skutkuje to czytelnością i łatwiejszym zarządzaniem kodem.

5.1.5. Stan aplikacji i warstwa prezentacji

Zgodnie z zasadą Fluttera "UI is a function of state"(pol. Warstwa widoku jest funkcją stanu), warstwa prezentacji jest bezpośrednio powiązana ze stanem aplikacji. Dzięki deklaratywnemu podejściu zmiany w danych automatycznie odzwierciedlane są w interfejsie użytkownika, co zapewnia spójność i aktualność prezentowanych informacji. Dane są trwałe i niemodyfikowalne - po ich utworzeniu nie można ich zmienić, a jedynie stworzyć nowe na ich podstawie. Ułatwia to zarządzanie stanem aplikacji i minimalizuje ryzyko błędów wynikających z nieoczekiwanych zmian danych. Widoki aplikacji są sterowane danymi i zawierają minimalną logikę biznesową, skupiając się głównie na prezentacji informacji i interakcji z użytkownikiem. Jest to osiągane między innymi poprzez stosowanie klasy `ChangeNotifier` i jej pochodnych, która powiadamia widoki o zmianach stanu. Flutter udostępnia także dwa typy widgetów: `StatelessWidget` (niemodyfikowalny) i `StatefulWidget` (modyfikowalny), które zapewniają efektywne i wydajne działanie. W aplikacji ReSpire wykorzystywane są oba typy, w zależności od potrzeb danego widoku.

5.1.6. Model serwisów

Warstwa logiczna aplikacji została zaprojektowana na podstawie architektury modułowej, obejmującej łącznie 12 wyspecjalizowanych serwisów. Każdy z komponentów realizuje odrębną domenę funkcjonalną, co pozwala na skuteczną separację logiki biznesowej od warstwy prezentacji. Zależności oraz relacje między kluczowymi modułami przedstawiono na rysunku 5.2.

Rysunek 5.2. Uproszczony model architektury serwisów

Poniżej przedstawiono krótki opis głównych odpowiedzialności poszczególnych komponentów:

- `TrainingController` — centralny moduł aplikacji, koordynujący przebieg sesji treningowej. Odpowiada za zarządzanie stanem ćwiczenia, odliczanie czasu oraz synchronizację warstwy audio z wizualizacją. Jego szczegółowa budowa została omówiona w dedykowanym rozdziale 6.4.2.
- `SoundManager` — globalny zarządca zasobów dźwiękowych. Odpowiada za ładowanie plików do pamięci, zarządzanie pulami odtwarzaczy oraz emisję sygnałów audio. Został dokładnie opisany wraz z innymi serwisami audio w rozdziale 6.5.
- `SingleSoundManager` — zarządca odtwarzania pojedynczych ścieżek, zapewniający obsługę zdarzeń zakończenia utworu.
- `PlaylistManager` — moduł sterujący sekwencyjnym odtwarzaniem utworów, delegujący zadania wykonawcze do niższych warstw systemu audio.
- `BinauralBeatGenerator` — serwis odpowiedzialny za syntezę dudnień różnicowych w czasie rzeczywistym.
- `TextToSpeechService` — moduł wykorzystujący bibliotekę `flutter_tts` do odczytywania liczb i nazw faz oddechowych przez syntezator mowy.
- `SettingsProvider` — komponent zarządzający trwałym zapisem konfiguracji użytkownika (np. wybór języka, lektora).
- `TranslationProvider` — serwis dostarczający tłumaczenia interfejsu w zależności od wybranego języka. Logika komponentu została szczegółowo opisana w rozdziale 6.6.
- `PresetDataBase` — klasa odpowiedzialna za zarządzanie bazą danych treningów zapisanych na urządzeniu użytkownika.
- `TrainingImportExportService` — moduł odpowiedzialny za operacje wejścia/wyjścia oraz wymianę danych z zewnętrznym systemem plików. Umożliwia trwały zapis treningów oraz ich udostępnianie pomiędzy różnymi urządzeniami. Proces importu i eksportu danych omówiono szerzej w sekcji 6.3.

- `TrainingJsonConverter` — warstwa transformacji danych, realizująca proces serializacji oraz deserializacji. Zapewnia standaryzację struktury plików treningowych, gwarantując ich poprawną interpretację podczas importu.

5.1.7. Model treningu

W celu zapewnienia elastyczności konfiguracji, model danych treningu został poddany dekompozycji i zgodnie z założeniami projektowymi, architektura opiera się na trzech głównych filarach. Uzupełnieniem tego modelu jest dziewięć klas pomocniczych, realizujących specyficzne, mniejsze zadania lub pełniących funkcję kontenerów danych. Taki podział zapewnia czytelność kodu oraz znacząco usprawnił implementację kolejnych funkcjonalności. Końcową architekturę przedstawia rysunek 5.3.

Rysunek 5.3. Diagram klas treningowych

Rdzeń modelu danych stanowią trzy kluczowe klasy tworzące strukturę hierarchiczną: `Trening`, `TrainingStage` oraz `BreathingPhase`. Relacja między nimi opiera się na kompozycji: obiekt nad-

rzędny (`Trening`) agreguje kolekcję etapów (`TrainingStage`), które z kolei zawierają listy faz oddechowych (`BreathingPhase`). Uzupełnione są one o następujące klasy pomocnicze:

- `Sounds` — klasa kontenerowa agregująca konfigurację dźwięków zdefiniowaną przez użytkownika, uwzględniająca ich zakresy obowiązywania (zdefiniowane przez `SoundScope`).
- `SoundScope` — typ wyliczeniowy określający zasięg odtwarzania przypisanego dźwięku. Dostępne warianty to: brak (`None`), globalny (`global`), dla etapu (`perStage`), dla fazy (`perPhase`) oraz cykliczny dla każdej fazy w etapie (`perEveryPhaseInEveryStage`).
- `SoundAsset` — struktura danych reprezentująca pojedynczy zasób audio. Przechowuje metadane pliku, takie jak nazwa wyświetlana oraz ścieżka dostępu do zasobu w systemie plików.
- `SoundType` — typ wyliczeniowy kategoryzujący rodzaje sygnałów dźwiękowych. Zbiór wartości obejmuje: brak (`None`), głos lektora (`Voice`), melodię tła (`Melody`), sygnał dźwiękowy (`Cue`) oraz odliczanie (`Counting`).
- `BreathingPhaseSounds` — klasa konfigurująca oprawę audio dla pojedynczej fazy oddechowej. Umożliwia zdefiniowanie ścieżki tła oraz sygnału poprzedzającego rozpoczęcie danej fazy.
- `BreathingPhaseType` — typ wyliczeniowy definiujący rodzaj czynności oddechowej. Wartości obejmują: wdech (`inhale`), wydech (`exhale`), zatrzymanie powietrza (`retention`) oraz regenerację (`recovery`).
- `BreathingPhaseIncrement` — struktura definiująca parametr progresji czasu trwania fazy. Określa wartość (wyrażoną w sekundach), o którą wydłużana jest faza w każdym kolejnym cyklu treningowym.
- `BreathingPhaseIncrementType` — typ wyliczeniowy określający jednostkę przyrostu czasu (obecnie obsługiwane są sekundy). Struktura ta została zachowana w celu zapewnienia elastyczności modelu pod kątem przyszłej implementacji inkrementacji procentowej.
- `Settings` — klasa agregująca globalne parametry sesji treningowej. Przechowuje konfigurację czasów przygotowania (`preparationDuration`) i zakończenia (`endingDuration`), a także ustawienia generatora dudnień różnicowych (status aktywacji oraz częstotliwość fali).

5.2. *Projekt interfejsu użytkownika (Hanna Banasiak, Aleksandra Bujny, Karol Zwierz)*

Podrozdział ten poświęcony jest projektowaniu interfejsu użytkownika opracowanej aplikacji mobilnej. Głównym celem było stworzenie intuicyjnej, wizualnie estetycznej oraz w pełni responsywnej warstwy użytkownika, zapewniającego pozytywne doświadczenia przy jednoczesnym spełnieniu wcześniej zdefiniowanych wymagań funkcjonalnych i niefunkcjonalnych. Do kluczowych wymagań funkcjonalnych interfejsu należały:

- przeglądanie i zarządzanie listą treningów,
- przebieg treningu oddechowego z licznikiem czasu i wyświetlanymi instrukcjami,
- zaawansowane tworzenie oraz edycja treningów oddechowych.

Spośród wymagań niefunkcjonalnych szczególnie uwzględniono:

- pełną responsywność projektu oraz podejście *mobile-first*,
- wsparcie dla dwóch wersji językowych (polski i angielski),
- wysoką czytelność i estetykę wizualną,
- częściową zgodność z wytycznymi dostępności WCAG 2.1 na poziomie AA.

Przy projektowaniu świadomie kierowano się wytycznymi dostępności WCAG 2.1 jako punktem odniesienia. Większość kluczowych zaleceń tego standardu - w tym odpowiedni kontrast tekstu (minimum 4.5:1), widoczny wskaźnik aktywnej klawiatury czy tekstowe alternatywy dla elementów nietekstowych - została zastosowana. Ze względu na prototypowy charakter projektu wykonanego w narzędziu Figma, pełna zgodność z WCAG 2.1 AA nie była celem, a jedynie wskazówką projektową.

Proces projektowania oparto na następujących metodykach i podejściach:

- *Mobile-first* - projekt został stworzony skupiając się w pierwszej kolejności na użytkownikach urządzeń mobilnych,
- *Design System* - zbudowano spójny system projektowy obejmujący paletę kolorów, typografię i komponenty, gwarantując jednolity wygląd i zachowanie aplikacji,
- *Atomic Design* - interfejs skonstruowano w oparcie o hierarchię komponentów.

5.2.1. Projekt w narzędziu Figma

Do zaprojektowania interfejsu użytkownika wybrano narzędzie Figma, które oferuje zaawansowane prototypowanie, w tym reakcję zmiany widoków po kliknięciu w dany obiekt. Dodatkowo narzędzie zostało wybrane również ze względu na jego znajomość przez członków zespołu, możliwość współpracy i łatwego udostępniania gotowego projektu.

W pierwotnej wersji ekran główny aplikacji stanowiła lista treningów oddechowych prezentowana w formie kafelków. Dodawanie nowego treningu realizowane było za pomocą przycisku pływającego, umieszczonego pod kafelkami. Dostępne były również opcje w prawym górnym rogu. Każdy kafelek treningu obsługiwał gesty przesunięcia w lewo, dając możliwość jego usunięcia, i w prawo - dając możliwość jego edycji. Projekt można zobaczyć na rysunku 5.4. Mechanizm przesuwalnych kafelków okazał się jednak nieintuicyjny dla użytkowników i postanowiono z niego zrezygnować. W finalnej wersji aplikacji każdy kafelek treningu po kliknięciu przenosi użytkownika na dedykowany ekran szczegółów treningu, na którym w wyraźny sposób udostępniono przyciski do edycji, usunięcia i włączenia treningu.

Rysunek 5.4. Pierwszy projekt aplikacji ReSpire - Strona Główna

Przebieg treningu oddechowego w prototypie aplikacji ReSpire zaprojektowano w sposób maksymalnie intuicyjny i czytelny dla użytkownika. W centralnej części ekranu znajduje się element animowany w postaci koła, które płynnie zmienia swój promień i wyświetla pozostały czas do końca obecnej fazy oddechowej. Bezpośrednio nad nim znajduje się komponent wyświetlający instrukcje w postaci trzech kafelków, przedstawiających kolejno od lewej fazę poprzednią, obecną i następną. Dodatkowo w tym samym obszarze prezentowany był globalny postęp treningu - liczba ukończonych cykli w stosunku do zaplanowanej liczby wszystkich cykli. Kliknięcie w centralne koło miało powodować natychmiastowe wstrzymanie treningu, a ponowne jego kliknięcie wznawiało sesję od miejsca przerwania. Przy próbie opuszczenia ekranu treningu aplikacja miała wyświetlać potwierdzenie z informacją, iż bieżąca sesja zostanie przerwana i jej postęp nie zostanie zapisany. Projekt przebiegu treningu przedstawiono na rysunku 5.5.

Rysunek 5.5. Pierwszy projekt aplikacji ReSpire - przebieg treningu

Ekran edycji treningu w pierwotym projekcie zawiera w górnej części tytuł, a w dolnej przycisk zapisu *Save*, z którego w wersji ostatecznej zrezygnowano, ze względu na wprowadzenie mechanizmu automatycznego zapisu zmian. Główną część ekranu stanowi panel z trzema zakładkami:

- *Training* - sekcja służąca do tworzenia i edycji własnych schematów oddechowych,
- *Sound* - ustawienia muzyki i sygnałów dźwiękowych,
- *Other* - pozostałe preferencje dotyczące treningu.

Wstępny projekt nie zakładał opisu treningu, który został dodany dopiero na późniejszych etapach rozwoju aplikacji. Prototyp przedstawiono na rysunku 5.6.

Rysunek 5.6. Pierwszy projekt aplikacji ReSpire - edycja treningu

5.2.2. Zrealizowany wygląd interfejsu użytkownika aplikacji

Finalnie zrealizowany projekt graficznego interfejsu użytkownika składa się z 8 głównych, pełnoekranowych widoków oraz *splash screen'a* (pol. ekranu powitalnego) z dodatkowymi wyskakującymi oknami dialogowymi. Na rysunku 5.7 przedstawiono diagram przejść między widokami. Strzałki obrazują nawigację, która odbywa się poprzez interakcję z elementami interfejsu (głównie przyciskami).

Rysunek 5.7. Możliwe przejścia między widokami w trakcie użytkowania aplikacji.

Wejściowym punktem aplikacji jest widok `HomePage`, do którego użytkownik trafia po wyświetleniu *SplashScreen* w momencie uruchomienia aplikacji. Z poziomu ekranu głównego użytkownik ma dostęp do przejścia do widoku wybranego treningu (`TrainingPage`) poprzez kliknięcie w odpowiadający mu kafelek (`PresetTile`) lub przejście do strony ustawień aplikacji (`SettingsPage`) za pomocą przycisku koła zębatego w prawym górnym rogu ekranu. Z poziomu widoku danego treningu użytkownik ma możliwość uruchomienia treningu poprzez kliknięcie przycisku "Rozpocznij trening". Użytkownik przenoszony jest wówczas na chwilę na ekran ładowania (`PreloadingScreen`), by następnie zostać przekierowanym przez aplikację do odtwarzacza zadanego treningu (`BreathingPage`). Po naciśnięciu przycisku edycji na ekranie treningu (`TrainingPage`) użytkownik przenosi się do edytora (`TrainingEditorPage`), a konkretniej do panelu edycji struktury treningu. Za pomocą górnej kontrolki `CustomSlidingSegmentedControl` (zachowującej się jak *radio button* (pol. przycisk opcji)) użytkownik może przełączać między aktualnymi aspektami edycji treningu (między strukturą, dźwiękami a zakładką "inne").

Powrót do poprzedniego widoku realizowany jest standardowo poprzez przycisk wstecz w lewym górnym rogu ekranu lub gest systemowy.

5.2.3. Identyfikacja wizualna aplikacji

W celu zapewnienia spójności wyglądu oraz estetyki aplikacji ReSpire przygotowana została identyfikacja wizualna obejmująca logo, kolory oraz czcionkę.

Logo aplikacji zostało zaprojektowane w narzędziu *Canva*, a część jego elementów była narysowana w edytorze graficznym *AdobeFresco*. Połączenie dwóch różnych czcionek o odmiennych grubościach oraz dodatek grafiki podmuchu i detalu symbolizującego cząsteczkę powietrza nad literą *i* oddaje ducha ReSpire. Logo używane w ekranie ładowania aplikacji oraz ikonie posiada napis umieszczony w dwóch liniach, jak przedstawiono na rysunku 5.8. Pomysł umieszczenia go na pasku na ekranie głównym spowodował, że powstała także druga wersja, w której całość mieści się w jednej linii, co pokazano na rysunku 5.9.

Rysunek 5.8. Logo ReSpire w oryginalnej wersji

Rysunek 5.9. Logo przystosowane do użycia na ekranie głównym aplikacji

W oparciu o kolorystykę elementów w logo zostały wybrane 3 główne kolory aplikacji. Są to odcienie niebieskiego. Kolor ten jest kojarzony z poczuciem spokoju i bezpieczeństwa [16]. Dzięki temu aplikacja jest spójna kolorystycznie, a także wywołuje przyjemne odczucia w użytkowniku, zapewniając wyciszenie i harmonię. Poniżej przedstawiono listę kolorów zapisanych w dwóch powszechnie stosowanych systemach zapisu barw oraz ich wizualizację:

HEX: #1A93A8 **RGB:** (26, 147, 168)

HEX: #32B7CF **RGB:** (50, 183, 207)

HEX: #7BDEF0 **RGB:** (123, 222, 240)

Rysunek 5.10. Paleta kolorów głównych aplikacji

Jedna z dwóch czcionek użytych w logo - *Glacial Indifference* [17] - została również wykorzystana w aplikacji, zapewniając jej unikatowy charakter. Jest to darmowy, otwartoźródłowy krój pisma zaprojektowany przez *Hanken Design Co.* z licencją *SIL OPEN FONT LICENSE Version 1.1* [18] pozwalającą na użytek zarówno prywatny, jak i komercyjny. Przykładowy tekst oraz znaki zapisane użytą czcionką przedstawione zostały na rysunku 5.11. Wykorzystana została także domyślna czcionka *Fluttera* dla aplikacji na systemie *Android* - *Roboto*.

Rysunek 5.11. Czcionka *Glacial Indifference* użyta w logo oraz aplikacji - po lewej w wersji pogrubionej, a po prawej w standardowej

Elementy charakterystyczne dla wyglądu ReSpire przedstawione zostały na rysunku 5.12. Białe kafelki z nazwami treningów na stronie głównej z asymetrycznymi zaokrągleniami oraz symbolami podmuchu stanowią prosty, ale modernistyczny element aplikacji. Przycisk dodawania własnego treningu nawiązuje do detalu cząstki powietrza w logo, czyniąc aplikację wyjątkową.

Rysunek 5.12. Kafelek z nazwą treningu oraz przycisk dodawania nowego poniżej

Wszytskie ikony używane w aplikacji pochodzą z otwartoźródłowej biblioteki *Material Icons* [19] należącej do *Google*. Są intuicyjne i minimalistyczne, dzięki czemu wygląd aplikacji jest spójny oraz przejrzysty dla użytkownika. Można je wykorzystywać na podstawie licencji *Apache License, Version 2.0* [18]. Przykładowe ikony przedstawiono na rysunku 5.13.

Rysunek 5.13. Przykładowe ikony z biblioteki *Material Icons*

W aplikacji zastosowano lekkie, łatwo skalowalne animacje w formacie Lottie [20]. Umożliwia on zapis w formie plików JSON, są bazowane na grafice wektorowej i idealnie nadaje się do aplikacji mobilnych, zapewniając wysoką płynność i niskie wykorzystanie zasobów. Wykorzystane zostały dwie darmowe animacje - fala [21] wykonana przez *CosmoYo* oraz łódka [22], której autorem jest *Sergey Riznyk* na podstawie licencji *Lottie Simple License (FL 9.13.21)* [23]. Zostały one dostosowane do potrzeb aplikacji poprzez między innymi: zmianę kolorów, usunięcie niektórych elementów, spowolnienie tempa odtwarzania czy zmianę orientacji. Animacje ożywiają warstwę prezentacji, dopełniając przestrzenie ekranu głównego i strony ze szczegółami treningu. Dzięki powolnemu tempu wprowadzają spokój.

5.3. Projekt struktury danych (Jakub Romanowski)

Model danych oparto na strukturze hierarchicznej, której fundament stanowią trzy kluczowe klasy: `Trening`, `EtapTreningu` oraz `FazaOddechowa`. Główna encja, `Trening`, składa się z co najmniej jednego `EtapuTreningu` (relacja jeden do wielu). Każdy etap charakteryzuje się zdefiniowaną liczbą cykli (powtórzeń) oraz parametrem przyrostu czasu. Na najniższym poziomie hierarchii znajduje się `FazaOddechowa`. Etapy składają się z sekwencji faz o określonym czasie trwania oraz typie, wybieranym ze zbioru zdefiniowanych wartości: wdech, wydech, regeneracja lub wstrzymanie. Opisywany projekt przedstawia rysunek 5.14, natomiast rozwinięcie i implementację tej struktury danych opisano w sekcji 5.1.7.

Rysunek 5.14. Struktura treningu

6. IMPLEMENTACJA APLIKACJI MOBILNEJ RESPIRE (Hanna Banasiak, Aleksandra Bujny, Jakub Romanowski, Karol Zwierz)

Projekt został wykonany w środowisku Flutter za pomocą zestawu narzędzi dla programistów Flutter w wersji 3.27.2 wspierającego język Dart w wersji 3.6.1. Ten zestaw narzędzi dla programistów umożliwia tworzenie aplikacji na platformy Android, iOS, Linux, macOS oraz Windows z wykorzystaniem jednej bazy kodu źródłowego. Wybór Fluttera był podyktowany jego wydajnością, bogatym zestawem narzędzi do tworzenia interfejsów użytkownika oraz dużą dostępnością wsparcia. Zapewnie trwałości danych między uruchomieniami zostało zrealizowane przy użyciu lokalnej, nierelacyjnej bazy danych Hive w wersji 2.2.3, przygotowanej pod środowisko Flutter oraz obiektowy język programowania Dart.

6.1. Schemat plików/klas/modułów (Aleksandra Bujny, Jakub Romanowski)

Aplikacja ReSpire charakteryzuje się modułową budową, na którą składają się łącznie 63 pliki źródłowe w języku Dart. Projekt obejmuje 80 klas oraz ponad 400 metod realizujących logikę biznesową i interfejs użytkownika. Całkowity rozmiar projektu wynosi nieco ponad 10 KLOC (*Thousands (kilo) Lines of Code* – pol. tysięcy linii kodu), z czego blisko 8900 stanowi tzw. czysty kod, określany jako SLOC (*Source Lines of Code* – pol. źródłowe linie kodu). Ze względu na taką złożoność struktury, diagramy klas i architektury zostały w niniejszej dokumentacji podzielone na logiczne fragmenty, aby zachować ich czytelność.

Kod aplikacji składa się z wielu folderów, gdzie struktura i zawartość większości z nich została automatycznie wygnerowana przy tworzeniu projektu za pomocą zestawu narzędzi *Flutter*, zapewniając elementy niezbędne do uruchomienia aplikacji na różnych platformach. Zostały one pokazane na rysunku 6.1. Foldery *android/*, *ios/*, *macos/*, *windows/* oraz *linux/* zawierają natywne projekty dla platform mobilnych oraz systemów operacyjnych. W katalogu *.idea/* umieszczone są pliki konfiguracji dla *Android Studio*, natomiast zawartość folderu *build/* jest automatycznie generowana przy budowie aplikacji. Główny kod źródłowy, który został napisany podczas tworzenia aplikacji, znajduje się w folderze *lib/*.

```
ReSpire/
├── dart_tool/...
├── idea/...
├── android/...
├── assets/...
├── build/...
├── ios/...
├── lib/...
├── linux/...
├── macos/...
├── scripts/...
├── web/...
├── windows/...
└── ...
```

Rysunek 6.1. Struktura plików projektu - główne foldery

Konstrukcja folderu *lib/* została przedstawiona na rysunku 6.2. Podział na podfoldery zgodnie ze ścisłą strukturą ułatwił organizację projektu oraz umożliwił zachowanie porządku w obszernym kodzie projektu. W folderze *components/* zostały umieszczone pliki zawierające bazę danych oraz elementy używane do budowania widoków aplikacji, które są natomiast zawarte w katalogu *pages/*. Na tym samym poziomie zagłębienia utworzony został folder *services/*, który przechowuje wszystkie serwisy i kontrolery, w tym dwa podfoldery *SoundManagers* oraz *TranslationProvider* dedykowane odpowiednio klasom zarządzającym dźwiękami w treningu oraz zmianą języka aplikacji. Pozostałe foldery - *theme/* oraz *utils/* zawierają natomiast pliki dotyczące kolorów używanych w aplikacji oraz funkcje pomocne w formatowaniu tekstu czy jednostek czasowych. Jedynym plikiem umieszczonym poza podkatalogami jest *main.dart*, który stanowi punkt wejścia do aplikacji.

```
lib/
├─ components/
│  ├─ BreathingPage/...
│  ├─ Global/...
│  ├─ HomePage/...
│  ├─ Settings/...
│  └─ TrainingEditorPage/...
├─ pages/...
├─ services/
│  ├─ SoundManagers/...
│  ├─ TranslationProvider/...
│  └─ ...
├─ theme/
│  └─ Colors.dart
├─ utils/
│  ├─ DurationFormatter.dart
│  └─ TextUtils.dart
└─ main.dart
```

Rysunek 6.2. Struktura plików projektu - kod źródłowy

Zasoby dostarczane do aplikacji umieszczone zostały w folderze *assets/* znajdującym się na tym samym poziomie zagłębienia, co *lib/*. Schemat przedstawiony został na rysunku 6.3. Zasoby złożone są z: animacji przechowywanych w katalogu *animations/*, plików JSON będących słownikami używanymi do tłumaczenia aplikacji w folderze*languages/* oraz dźwięki w katalogu *sounds/*, który został podzielony na trzy podfoldery dla różnych typów plików dźwiękowych. Do zasobów należą także ilustracje logo umieszczone bezpośrednio w opisywanym folderze.

```
assets/
├─ animations/...
├─ languages/...
├─ sounds/
│  ├─ counting/...
│  ├─ long/...
│  ├─ short/...
│  └─ generator.dart
└─ ...
```

Rysunek 6.3. Struktura plików projektu - pliki zasobów

6.2. Edytor treningów (Karol Zwierz)

Edytor treningów został wykonany jako dedykowany widok `TrainingEditorPage`, który operuje bezpośrednio na przekazanym mu przez referencję konkretnym obiekcie `Training`. Struktura treningu przechowywana jest w odpowiednich obiektach klas (`TrainingStage`, `BreathingPhase`, `Sounds`, `Settings`), dzięki czemu każda modyfikacja wprowadzona w interfejsie natychmiast zostaje zapisana w pamięci a następnie po przejściu do poprzedniego etapu zostaje przekazana do zapisu stałego w bazie danych.

Obsługa edytora została podzielona na trzy logiczne panele. Aktualnie wyświetlaną zakładkę można wybrać poprzez komponent `CustomSlidingSegmentedControl` inspirowany obecnym w komponencie `segmented` z systemu Apple iOS. Pozwoliło to oddzielić edycję przebiegu treningu, konfigurację dźwięków oraz ustawienia uzupełniające, utrzymując jednocześnie wspólną nawigację i kontrolę zapisów.

6.2.1. Menu - trening

Pierwsza zakładka `Trening` odpowiada za logiczny układ hierarchicznej sesji oddechowej. Reprezentowana jest przez listę o zmiennym porządku - `ReorderableListView`, w której wyświetlane są kolejne etapy treningu (`TrainingStageTile`). Każdy kafelek etapu treningu umożliwia zmianę jego nazwy, liczby powtórzeń i iteracyjnego przyrostu czasów, a także usunięcie całego etapu treningu za potwierdzeniem w wyskakującym oknie dialogowym zaimplementowanym za pomocą `AlertDialog`. Wewnątrz każdego kafelka etapu treningu umieszczona jest lista faz oddechu reprezentowana przez sekwencję komponentów `BreathingPhaseTile` zagnieżdżonych w kolejnej `ReorderableListView`. Dzięki temu zarówno etapy treningu w obrębie całego treningu jak i fazy oddechu w obrębie etapu można uporządkować w kolejności zgodnej z upodobaniem użytkownika w trybie „przeciągnij za kontrolkę i upuść". Pola numeryczne panelu edycji etapu treningu (liczba powtórzeń, przyrost) oraz fazy oddechu (czas trwania) wykorzystują `TextEditingController` oraz `FocusNode`, aby zatwierdzać zmienione wartości dopiero po utracie skupienia na danym elemencie wprowadzania danych, a dodatkowe przyciski +/- umożliwiają inkrementalne korekty w krokach od 0,1 do 0,5 s. Dodawanie nowych faz i etapów wywołuje przewinięcie listy oraz zdejmuje skupienie z aktywnych pól, co zostało wprowadzone w celu zapobiegania konfliktom z klawiaturą ekranową. Przy próbie opuszczenia ekranu edycji treningu komponent opakowujący `WillPopScope` sprawdza, czy wszystkie etapy treningu zawierają przynajmniej jedną fazę, i w razie potrzeby wyświetla komunikat z możliwością wyboru automatycznego usunięcia pustych etapów treningu lub powrotu do edycji treningu oddechowego.

6.2.2. Menu - dźwięki

Druga zakładka służy do zarządzania warstwą dźwiękową. Korzysta w tym celu ze współdzielonego obiektu klasy `Sounds`. Sprowadza się ona do przypisywania plików dźwiękowych do konkretnych sytuacji w treningu. Zakładka ta dzieli się na 3 grupy.

W `Dźwiękach treningu` użytkownik może ustawić sygnał odliczania, sposób odtwarzania komunikatów o nadejściu kolejnej fazy (brak, per faza, globalnie - wybierane z listy rozwijalnej) i wybrać wskazówki audio zmiany etapu oraz cyklu.

W grupie `Muzyka treningu` może zdefiniować tło muzyczne w jednym z pięciu zakresów wybieranych z listy rozwijalnej: brak, globalnie, per etap, per faza, per faza w danym etapie. W trybie globalnym oraz trybie per etap wykorzystywane są komponenty edycji list odtwarzania. W za-

kresie globalnym jest to `PlaylistEditor`, który obsługują listy odtwarzania poprzez przeciąganie pozycji, usuwanie oraz dodawanie nowych ścieżek z okna `AudioSelectionPopup`. Dla wariantu per etap zastosowano `StagePlaylistsEditor` (korzystający z `PlaylistEditor`), przyporządkowujący listy odtwarzania na identyfikatory etapów (`TrainingStage.id`). We wszystkich miejscach obsługiwanych przez `AudioSelectionPopup` użytkownik może także zaimportować własne pliki audio poprzez `file_picker`; pliki są zapisywane w lokalnym systemie plików aplikacji i natychmiast dostępne. Ponadto wszystkie dźwięki w `AudioSelectionPopup` można odsłuchiwać dzięki `SingleSoundManager`. Dodatkowo przewidziano osobne pola wyboru dla muzyki odtwarzanej podczas specjalnych faz przygotowawczej i końcowej.

W trzeciej grupie zatytułowanej `Dźwięki Binauralne` aplikacja umożliwia wyłączenie muzyki w tle i zamiast niej włączenie odtwarzania tzw. dudnień binauralnych z regulacją częstotliwościami składowymi tychże dźwięków.

Istotnym aspektem edytora jest również zaawansowana konfiguracja listy odtwarzania dla każdego etapu. Widget `PlaylistEditor` (listing kodu 6.1) umożliwia użytkownikom nie tylko wybór plików dźwiękowych, ale także ich intuicyjne porządkowanie. Wykorzystano tu komponent `ReorderableListView`, który obsługuje gesty przeciągania elementów. Dodatkowo, w celu poprawy doświadczenia użytkowników, czasy trwania utworów są ładowane asynchronicznie, co zapobiega blokowaniu interfejsu podczas odczytu metadanych plików audio.

```
final Map<String, Duration?> _durationCache = {};

Future<void> _loadDurations() async {
  for (var sound in widget.playlist) {
    if (!_durationCache.containsKey(sound.name)) {
      final duration = await
          _soundManager.getSoundDuration(sound.name);
      if (mounted) {
        setState(() {
          _durationCache[sound.name] = duration;
        });
      }
    }
  }
}

void _reorderSounds(int oldIndex, int newIndex) {
  if (oldIndex < newIndex) {
    newIndex -= 1;
  }
  final newPlaylist = List<SoundAsset>.from(widget.playlist);
  final item = newPlaylist.removeAt(oldIndex);
  newPlaylist.insert(newIndex, item);
  widget.onChanged(newPlaylist);
}

@override
Widget build(BuildContext context) {
  return Column(
    children: [
      ReorderableListView.builder(
        shrinkWrap: true,
        physics: const NeverScrollableScrollPhysics(),
        itemCount: widget.playlist.length,
        onReorder: _reorderSounds,
        itemBuilder: (context, index) {
          final sound = widget.playlist[index];
          return _buildSoundTile(sound, index);
        },
      )
    ],
  );
}
```

Listing 6.1. Logika zarządzania listą odtwarzania w uproszczonym kodzie edytora (PlaylistEditor)

6.2.3. Menu - inne

Zakładka „Inne” obsługuje edycję obiektu klasy `Settings`. Zawiera pole `TextField` do wprowadzenia opisu treningu, a także licznik czasu przygotowawczego z własnym formatowaniem cyfr oraz walidacją wartości minimalnej, co przedstawiono w listingu kodu 6.2.

```
TextField(
  key: ValueKey('preparation_${widget.training.hashCode}'),
  controller: preparationController,
  focusNode: preparationFocusNode,
  keyboardType: TextInputType.number,
  textAlign: TextAlign.center,
  inputFormatters: [
    FilteringTextInputFormatter.digitsOnly,
  ],
  onChanged: (value) {
    int newValue = int.tryParse(value) ?? 0;
    setState(() {
      widget.training.settings.preparationDuration = newValue;
    });
  }
)
```

Listing 6.2. Obsługa pola tekstowego z walidacją dla czasu przygotowania

Tak jak każdy element graficznego interfejsu użytkownika w aplikacji, edytor jest przetłumaczony z użyciem `TranslationProvider`, co umożliwia dynamiczne zmiany języka i konsekwentne stosowanie lokalnych etykiet w dialogach, walidatorach czy komunikatach ostrzegawczych. Dzięki temu moduł edytora zachowuje spójność wizualną i logiczną z resztą aplikacji oraz zapewnia użytkownikowi poczucie kontroli nad każdym aspektem treningu bez konieczności przełączania kontekstu.

6.3. Importowanie oraz eksportowanie treningów (Karol Zwierz)

Funkcje importu oraz eksportu treningów w aplikacji zostały zaimplementowane w celu zapewnienia użytkownikom łatwej wymiany konfiguracji treningowych pomiędzy różnymi urządzeniami, oraz tworzenia kopii zapasowych swoich ustawień pod nieobecność systemu scentralizowanego serwera do trwałego zapisu i synchronizacji danych.

Rozwiązanie jest realizowane przez dedykowany serwis TrainingImportExportService. Odgrywa on rolę fasady dla operacji wejścia–wyjścia związanych z plikami treningowymi, lecz sam nie zajmuje się logiką biznesową. Logikę biznesową wykonuje na żądanie wcześniej wymienionego serwisu klasa pomocnicza `TrainingJsonConverter`, która zajmuje się operacjami serializacji obiektów treningów i deserializacji zrzutów danych. W celu wywołania natywnych, systemowych okien dialogowych wyboru i zapisu plików została wykorzystana biblioteka `file_picker` zapewniająca spójne doświadczenie użytkowników niezależnie od platformy (Android/iOS).

Eksport danych możemy wykonać z dwóch miejsc w aplikacji. Pierwszą możliwością jest naciśnięcie przycisku udostępniania dostępnego z poziomu widoku szczegółów treningu (`TrainingPage`). Można w ten sposób wyeksportować pojedynczy trening okraszony dynamicznie wygenerowaną na podstawie tytułu treningu nazwą pliku. Drugim punktem wywołania eksportu jest zaimplementowany na ekranie głównym (`HomePage`), przycisk umożliwiający masowy eksport wielu treningów jednocześnie. Niezależnie od wybranego punktu dostępu, za realizację operacji odpowiada ta sama metoda serwisowa, której implementację przedstawiono na listingu 6.3. Użytkownik, korzystając z trybu wyboru, zaznacza poprzez dotknięcie interesujące go pozycje, które następnie są pakowane w zbiorczą strukturę JSON. Proces wybierania został celowo zaprojektowany w sposób zbliżony do zaznaczania plików w managerach plików systemów mobilnych. Plik wynikowy otrzymuje nazwę zawierającą znacznik czasu, co ułatwia katalogowanie kopii zapasowych.

Struktura wyeksportowanego pliku JSON jest kompletnym odzwierciedleniem modelu danych aplikacji. Zawiera ona metadane (tytuł, opis), pełną hierarchię etapów (`TrainingStage`) wraz z fazami oddechowymi (`BreathingPhase`) i ich parametrami czasowymi, a także szczegółową konfigurację ustawień (`Settings`) oraz zależności dźwięków (`Sounds`). Dzięki temu wyeksportowany plik jest samowystarczalną jednostką informacji, możliwą do osadzenia w dowolnej innej instancji aplikacji.

Metody parsujące w klasie pomocniczej `TrainingJsonConverter` zostały zaprojektowane tak, aby rozpoznawać i poprawnie przetwarzać zarówno pojedyncze obiekty treningów, jak i listy obiektów lub struktury opakowane (używane przy eksporcie masowym). Po wczytaniu danych następuje proces walidacji oraz integracji, w ramach którego odtwarzane są powiązania do zasobów dźwiękowych metodą `updateSounds()`. Poprawnie zweryfikowane treningi są następnie dodawane do lokalnej bazy danych Hive, a interfejs użytkownika jest natychmiastowo odświeżany. Implementację metody realizującej ten proces przedstawiono na listingu 6.4.

```
static Future<bool> exportTraining(Training training, {String?
    fileName}) async {
  try {
    final String defaultFileName = fileName ??
    '${TextUtils.sanitizeFileName(training.title)}_training.json';

    final String jsonString =
     TrainingJsonConverter.toJson(training);
    final Uint8List bytes =
      Uint8List.fromList(utf8.encode(jsonString));

    String? outputPath = await FilePicker.platform.saveFile(
      dialogTitle: TranslationProvider().getTranslation(
        "FilePicker.save_training"),
      fileName: defaultFileName,
      type: FileType.custom,
      allowedExtensions: ['json'],
      bytes: bytes,
    );

    return outputPath != null;
  } catch (e) {
    debugPrint('Error during training export: $e');
    return false;
  }
}
```

Listing 6.3. Metoda eksportu treningu

```
static Future<List<Training>?> importTrainings() async {
  try {
    FilePickerResult? result = await
        FilePicker.platform.pickFiles(
      type: FileType.custom,
      allowedExtensions: ['json'],
      allowMultiple: false,
    );

    if (result == null || result.files.isEmpty) {
      return null;
    }

    final PlatformFile file = result.files.single;
    String? jsonString;

    if (file.bytes != null) {
      jsonString = utf8.decode(file.bytes!);
    } else if (file.path != null) {
      jsonString = await File(file.path!).readAsString();
    }

    if (jsonString == null) return null;

    return TrainingJsonConverter.fromJsonMultiple(jsonString);
  } catch (e) {
    debugPrint('Error during training import: $e');
    return null;
  }
}
```

Listing 6.4. Metoda importu treningów

6.4. Przebieg treningu (Hanna Banasiak)

6.4.1. Klasa TrainingParser

Klasa `TrainingParser` powstała w celu przekształcenia hierarchicznych danych treningowych pobranych z lokalnej bazy danych *Hive* w ciąg występujących po sobie faz, oparty na skonfigurowanym uprzednio przez użytkownika wzorcu oddychania. Jej zadaniem jest zwracanie kolejnych faz do obiektu `TrainingController`. Dzięki temu logika przełączania faz i powtórzeń jest odseparowana od interfejsu.

Konstruktor jako parametr przyjmuje obiekt klasy `Training` i zapisuje do zmiennej *currentTrainingStage* pierwszy etap treningu. Fragment realizujący tę funkcjonalność przedstawiono poniżej w listingu kodu 6.5.

```
Training training;
TrainingStage currentTrainingStage;
late breathing_phase.BreathingPhase currentBreathingPhase;

TrainingParser({required this.training})
    : currentTrainingStage = training.trainingStages[0];
```

Listing 6.5. TrainingParser - konstruktor

Zadaniem funkcji `nextInstruction`, przedstawionej na listingu 6.6, jest zwrócenie danych dotyczącej kolejnej fazy oddechowej w postaci mapy `Map<String, dynamic>` lub wartości `null` w przypadku zakończenia całego treningu.

W pierwszej kolejności analizowany jest aktualny indeks fazy oddechowej (*breathingPhaseID*). Jeżeli wskazuje on ostatnią fazę w bieżącym etapie, oznacza to zakończenie jednego pełnego cyklu etapu. W takim przypadku indeks fazy jest zerowany (*breathingPhaseID = 0*), a licznik wykonanych powtórzeń w etapie (*doneReps*) zwiększany jest o jeden. Następnie sprawdzana jest liczba wykonanych powtórzeń w odniesieniu do wartości zdefiniowanej w obiekcie etapu (*currentTrainingStage.reps*). W przypadku jej osiągnięcia następuje przejście do kolejnego etapu treningu poprzez inkrementację indeksu *trainingStageID*. Jeżeli po tej operacji indeks ten osiągnie wartość równą liczbie wszystkich etapów w strukturze treningu, funkcja zwraca *null*, sygnalizując zakończenie sesji. W przeciwnym razie wczytywany jest nowy etap (*currentTrainingStage = training.trainingStages[trainingStageID]*), a licznik *doneReps* zostaje zresetowany do zera. Gdy aktualna faza nie była ostatnią w cyklu, indeks *breathingPhaseID* jest jedynie zwiększany o jeden. Po ustaleniu poprawnego indeksu do zmiennej *currentBreathingPhase* przypisywana jest odpowiadająca mu faza oddechowa. Kolejnym etapem jest obliczenie rzeczywistego czasu trwania fazy z uwzględnieniem mechanizmu progresji.

Na podstawie obliczonego czasu tworzona jest zmienna *progressedBreathingPhase*, w której pole *duration* przyjmuje wartość *durationSeconds*, natomiast pozostałe atrybuty (*breathingPhaseType*, *breathType*, *breathDepth*, *sounds*) są kopiowane z obecnej fazy oddechowej.

Funkcja zwraca mapę zawierającą następujące klucze:

- *breathingPhase* — pełny obiekt fazy,
- *remainingTime* — czas trwania fazy wyrażony w milisekundach,
- *trainingStageName* — nazwę aktualnego etapu treningu.

W ten sposób *nextInstruction()* pełni rolę centralnego mechanizmu sterującego przebiegiem treningu oddechowego, zapewniając poprawne przechodzenie pomiędzy fazami i etapami oraz automatyczne zwiększanie trudności zgodnie z zaimplementowanym modelem progresji liniowej.

```
Map<String, dynamic>? nextInstruction() {
  if (breathingPhaseID ==
      currentTrainingStage.breathingPhases.length - 1) {
    breathingPhaseID = 0;
    doneReps++;

    if (doneReps == currentTrainingStage.reps) {
      trainingStageID++;
      if (trainingStageID == training.trainingStages.length) {
        return null;
      } else {
        currentTrainingStage =
            training.trainingStages[trainingStageID];
        doneReps = 0;
      }
    }
  } else {
    breathingPhaseID++;
  }

  currentBreathingPhase =
      currentTrainingStage.breathingPhases[breathingPhaseID];

  double durationSeconds = currentBreathingPhase.duration +
      (currentTrainingStage.increment * doneReps);

  final progressedBreathingPhase = breathing_phase.BreathingPhase(
    duration: durationSeconds,
    breathingPhaseType: currentBreathingPhase.breathingPhaseType,
    breathType: currentBreathingPhase.breathType,
    breathDepth: currentBreathingPhase.breathDepth,
    sounds: currentBreathingPhase.sounds,
  );

  return {
    "breathingPhase": progressedBreathingPhase,
    "remainingTime": (durationSeconds * 1000).truncate(),
    "trainingStageName": currentTrainingStage.name,
  };
}
```

Listing 6.6. TrainingParser - pobranie instrukcji

6.4.2. Klasa TrainingController

`TrainingController` jest centralnym kontrolerem sesji treningu w aplikacji. Odpowiada za precyzyjne, milisekundowe sterowanie czasem, płynne przechodzenie między fazami oddechowymi i inteligentne zarządzanie wszystkimi warstwami dźwiękowymi — muzyka w tle, playlisty, odliczanie, zapowiedzi i dudnienia różnicowe. Zapewnia również poprawne przeprowadzenie użytkownika przez trzy główne etapy — przygotowanie, właściwy trening i fazę końcową. Komponent zarządza wieloma zmiennymi, które zostały przedstawione w tabeli 6.4.2.

Pole	**Znaczenie**
`_trainingStageIdQueue`	Równoległa kolejka ID etapów treningu. Pozwala wykryć zmianę etapu przed rozpoczęciem pierwszej fazy nowego etapu, co jest kluczowe dla natychmiastowego przełączenia playlisty i dźwięku zmiany etapu.
`breathingPhasesQueue`	Nasłuchiwana przez obiekt `InstructionSlider` kolejka faz oddechowych. Zawsze zawiera trzy elementy, które mogą również przyjąć wartość `null`. Pod indeksem 0 znajduje się obecnie trwająca faza, a na kolejnych dwie przyszłe. Mechanizm umożliwia wcześniejsze odtwarzanie zapowiedzi i płynną zmianę dźwięku tła i animacji.
`_remainingTime`	Pozostały czas fazy bieżącej w milisekundach.
`_nextRemainingTime`	Czas następnej fazy wczytany z wyprzedzeniem.
`second`	Zmienna typu `ValueNotifier<int>`, nasłuchiwana przez elementy interfejsu użytkownika do dynamicznego wyświetlania pozostałego czasu do końca fazy.
`isPaused`	Nasłuchiwana przez kontroler flaga stanu pauzy.
`end`	Flaga całkowitego zakończenia sesji.
`currentStageIndex`	Nasłuchiwany licznik numeru aktualnego etapu wyświetlany w interfejsie użytkownika.
`currentCycleIndex`	Nasłuchiwany licznik numeru aktualnego cyklu w etapie wyświetlany w interfejsie użytkownika.
`totalStages`	Łączna liczba etapów
`totalCycles`	Łączna liczba powtórzeń w bieżącym etapie.
`_preparationPhaseCompleted`	Flaga zakończenia fazy przygotowania
`_endingInitiated`	Flaga rozpoczęcia fazy zakończenia

Tabela 6.5.2. Główne zmienne w klasie `TrainingController`.

Konstruktor przyjmuje jako parametr obiekt `TrainingParser` i zapisuje go do zmiennej *parser*. Dodatkowo przy tworzeniu komponentu tworzone są obiekty `SoundManager`, `PlaylistManager` i `BinauralBeatGenerator` zajmujące się obsługą dźwięków. Wyłączane jest także wyświetlanie etykiet progresu treningu, dopóki nie zostanie on rozpoczęty. Następnie ustawiane są wartości początkowe, przygotowywana jest faza rozpoczęcia i pobierane są pierwsze fazy treningu. Konstruktor `TrainingController` przedstawiono w listingu 6.7.

```
TrainingController(this.parser) {
  soundManager = SoundManager();
  soundManager.stopAllSounds();
  playlistManager = PlaylistManager();
  binauralGenerator = BinauralBeatGenerator();
  _sounds = parser.training.sounds;
  _settings = parser.training.settings;
  showLabels.value = false;

  if (parser.training.trainingStages.isNotEmpty) {
    _currentTrainingStageId =
       parser.training.trainingStages[0].id;
    totalStages.value = parser.training.trainingStages.length;
    currentStageIndex.value = 1;
    currentCycleIndex.value = 1;
    totalCycles.value = parser.training.trainingStages[0].reps;
  }
  _preloadBreathingPhases();
  _initializePreparationSound();
  _loadEndingSoundDuration();
}
```

Listing 6.7. TrainingController - konstruktor

Funkcja `_preloadBreathingPhases` służy do inicjalizacji kolejki faz oddechowych w treningu. Na początku tworzona jest pusta faza z wartościami *null*. Spowodowane jest to potrzebą symulacji fazy przygotowania, która nie posiada żadnych wspólnych cech z pozostałymi fazami poza czasem trwania. Przypisanie wartości *null* pozwala na rozróżnienie jej od właściwego treningu w wizualizacji instrukcji użytkownika w klasie `InstructionSlider`. Następnie dodawane są dwie pierwsze fazy oddechowe treningu. Funkcję przedstawiono w listingu kodu 6.8.

```
void _preloadBreathingPhases() {
  breathingPhasesQueue.value.add(null);
  _trainingStageNameQueue.add(null);
  _trainingStageIdQueue.add(null);
  _updateCurrentTrainingStageLabel();
  _fetchNextBreathingPhase();
  _nextRemainingTime = _newBreathingPhaseRemainingTime;
  _fetchNextBreathingPhase();
}
```

Listing 6.8. TrainingController - przygotowanie kolejki

Pobieranie kolejnych faz realizowane jest za pomocą metody *_fetchNextBreathingPhase*. Dane instrukcji w postaci mapy uzyskiwane są z obiektu *parser*. W przypadku zwrócenia przez parser wartości *null*, oznaczającej zakończenie pobierania danych treningowych, następuje oznaczenie zakończenia ładowania w zmiennej *_finishedLoadingBreathingPhases* oraz dodanie końcowej fazy *null*, reprezentującej fazę zakończenia. W przeciwnym razie wartości fazy, nazwa etapu treningowego, identyfikator etapu oraz czas pozostały są zapisywane do odpowiednich kolejek, a czas trwania nowej fazy zapamiętywany jest w polu *_newBreathingPhaseRemainingTime*. Metodę przedstawiono w listingu 6.9.

```
void _fetchNextBreathingPhase() {
  var instructionData = parser.nextInstruction();
  if (instructionData == null) {
    _finishedLoadingBreathingPhases = true;
    breathingPhasesQueue.value.add(null);
    _trainingStageNameQueue.add(null);
    _trainingStageIdQueue.add(null);
    return;
  }

  breathingPhasesQueue.value.add(
    instructionData["breathingPhase"]);
  _trainingStageNameQueue.add(_resolveTrainingStageName(
      instructionData["trainingStageName"] as String?,
      parser.trainingStageID));

  String? stageId;
  if (parser.trainingStageID <
      parser.training.trainingStages.length) {
    stageId =
    parser.training.trainingStages[parser.trainingStageID].id;
  }
  _trainingStageIdQueue.add(stageId);

  _newBreathingPhaseRemainingTime =
      instructionData["remainingTime"];
}
```

Listing 6.9. TrainingController - pobieranie fazy

Następnie wywoływana jest metoda *start*, która odpowiada za cały przebieg treningu. Bazuje ona na obiekcie `Timer` z pakietu `dart`, który cyklicznie wykonuje zdefiniowane zachowanie co zadany odcinek czasu zapisany w zmiennej *_updateInterval*. Ponieważ standardowy `Timer` okresowy nie gwarantuje idealnej precyzji i mogą występować opóźnienia, zastosowano mechanizm korekcji oparty na pomiarze rzeczywistego upływającego czasu. W tym celu wykorzystano zmienne *lastTick* i *now*, które pozwalają na bieżąco wyliczać faktycznie upłynięty czas pomiędzy kolejnymi wywołaniami i korygować ewentualne niedokładności zegara. Różnica ta zapisywana jest w zmiennej *elapsed*. Fragment metody przedstawiono w listingu 6.10.

```
void _start() {
  int previousSecond = _remainingTime ~/ 1000;
  DateTime lastTick = DateTime.now();
  soundManager.playSound(_currentSound);
  bool skipFirstCounting = false;

  _timer = Timer.periodic(Duration(
    milliseconds: _updateInterval), (Timer timer) {
    final now = DateTime.now();
    final int elapsed = now.difference(lastTick).inMilliseconds;
    lastTick = now;
```

Listing 6.10. TrainingController - rozpoczęcie treningu

W kolejnym etapie pętli sterującej realizowana jest precyzyjna logika zarządzania czasem pozostałym do zakończenia bieżącej fazy oddechowej oraz synchronizacja wydarzeń dźwiękowych z rzeczywistym przebiegiem treningu. Zmienna *previousSecond* przechowuje wartość ostatniej wyświetlonej sekundy, natomiast *second* służy do natychmiastowej aktualizacji interfejsu użytkownika. Również wtedy, w zależności od aktualnego etapu treningu i ustawień dźwiękowych wywoływana jest metoda *_playCountingSound*, odpowiedzialna za odtwarzanie sygnału odliczającego. Mechanizm ten jest aktywny wyłącznie wtedy, gdy nie zachodzi celowe pominięcie pierwszego odliczania (zmienna *skipFirstCounting*) oraz gdy w danej fazie użytkownik nie wybrał braku ścieżki dźwiękowej. Fragment przedstawiono w listingu kodu 6.11.

```
if (previousSecond > _remainingTime ~/ 1000 && !end) {
  previousSecond = _remainingTime ~/ 1000;
  second.value = previousSecond;
  if(!skipFirstCounting &&
  ((!_preparationPhaseCompleted &&
  _sounds.preparationTrack.type == SoundType.none) ||
    (_preparationPhaseCompleted && !_endingInitiated) ||
    (_endingInitiated && _sounds.endingTrack.type ==
       SoundType.none))){
    _playCountingSound(previousSecond);
  }else{
    skipFirstCounting = false;
  }
}
```

Listing 6.11. TrainingController - odliczanie

Dalsza część algorytmu koncentruje się na zapewnieniu płynnego przejścia pomiędzy kolejnymi fazami oddechowymi z odpowiednim wyprzedzeniem dźwiękowym. Jeżeli w kolejce faz znajduje się co najmniej jedna kolejna faza, system monitoruje pozostały czas bieżącej fazy. Gdy osiągnie on wartość 100 ms lub mniej i sygnał zapowiadający zmianę fazy nie został jeszcze odtworzony, wywoływana jest metoda *_playPreBreathingPhaseSound* i jest on odtwarzany. Co więcej, przed końcem fazy inicjowana jest łagodna zmiana muzyki tła poprzez wywołanie metody *_handleBackgroundSoundChange*. Dzięki temu przejście między ścieżkami dźwiękowymi

jest płynne i przyjemne dla użytkownika. Na końcu każdej iteracji pętli dokonywana jest aktualizacja licznika czasu pozostałego. Jeżeli jego wartość jest równa lub mniejsza od zmierzonej ilość upływającego czasu (*elapsed*), ale nadal dodatnia, czas pozostały jest jawnie zerowany. Ten mechanizm korekcji jest kluczowy z punktu widzenia stabilności numerycznej algorytmu. Eliminuje on ryzyko uzyskania ujemnych wartości licznika na skutek drobnych niedokładności zegara systemowego lub opóźnień, zapobiegając tym samym pomijaniu faz lub ich nieprawidłowemu kolejkowaniu. Fragment przedstawiono w listingu 6.12.

```
if (previousSecond > _remainingTime ~/ 1000 && !end) {
  previousSecond = _remainingTime ~/ 1000;
  second.value = previousSecond;
  if(!skipFirstCounting &&
  ((!_preparationPhaseCompleted &&
  _sounds.preparationTrack.type == SoundType.none) ||
    (_preparationPhaseCompleted && !_endingInitiated) ||
    (_endingInitiated && _sounds.endingTrack.type ==
       SoundType.none))){
    _playCountingSound(previousSecond);
  }else{
    skipFirstCounting = false;
  }
}

if (_remainingTime > elapsed) {
  _remainingTime -= elapsed;

  if (breathingPhasesQueue.value.length > 1 &&
    breathingPhasesQueue.value.elementAt(1) != null) {
    final nextPhase = breathingPhasesQueue.value.elementAt(1)!;
    if(!_nextPhaseSoundPlayed && _remainingTime <= 100) {
      _playPreBreathingPhaseSound(nextPhase);
      _nextPhaseSoundPlayed = true;
    }

    if (_remainingTime <= 300 && _remainingTime > 200) {
      breathing_phase.BreathingPhase breathingPhase =
          breathingPhasesQueue.value.elementAt(1)!;
      _handleBackgroundSoundChange(
          breathingPhase.sounds.background.name, 500);
    }
  }
} else if (_remainingTime > 0) {
  _remainingTime = 0;
}
```

Listing 6.12. TrainingController - dźwięki treningu

Po osiągnięciu zera przez licznik czasu bieżącej fazy następuje przejście do kolejnej fazy oddechowej. W przypadku pierwszej fazy treningu właściwego, jeżeli użytkownik włączył dudnienia różnicowe, obliczana jest całkowita długość części głównej treningu, a następnie uruchamiany jest generator dudnienień różnicowych z zadanymi częstotliwościami lewego i prawego kanału. Równolegle, przy pierwszym przejściu z fazy przygotowawczej do treningu właściwego, zatrzymywany jest aktualnie odtwarzany dźwięk przygotowawczy i faza przygotowawcza oznaczana jest jako zakończona. Aktywowane jest również wyświetlanie etykiet w interfejsie (*showLabels*). Następnie, w zależności od ustawiono globalny zakres muzyki tła, aktywowany zostaje tryb odtwarzania playlisty. Natomiast, jeśli ustawiony jest zakres dla etapu, wywoływana jest metoda *_switchToStagePlaylist*, która inicjuje odtwarzanie playlisty przypisanej do danego etapu. Fragment przedstawiono w listingu 6.13.

```
if (_remainingTime == 0) {
  breathingPhasesCount.value++;

  if (breathingPhasesCount.value == 1 &&
      _settings.binauralBeatsEnabled) {
    final trainingDuration =
        parser.calculateTrainingDurationWithoutPreparation();

    binauralGenerator.start(
      _settings.binauralLeftFrequency,
      _settings.binauralRightFrequency,
      durationSeconds: trainingDuration,
    );
  }

  if (breathingPhasesCount.value == 1 &&
      !_preparationPhaseCompleted) {
    soundManager.stopSound(_currentSound);
    _currentSound = null;
    _preparationPhaseCompleted = true;
    showLabels.value = true;
    if (_sounds.backgroundSoundScope == SoundScope.global &&
        _sounds.trainingBackgroundPlaylist.isNotEmpty) {
      _isUsingPlaylist = true;
      playlistManager.playPlaylist(
          _sounds.trainingBackgroundPlaylist.map((s) =>
              s.name).toList());
    } else if (_sounds.backgroundSoundScope ==
        SoundScope.perStage &&
        _currentTrainingStageId != null) {
      _switchToStagePlaylist(_currentTrainingStageId!);
    }
  }
```

Listing 6.13. TrainingController - inne dźwięki

Po osiągnięciu zera przez licznik bieżącej fazy realizowana jest logika zakończenia danej fazy oraz ewentualnego przejścia do fazy końcowej lub całkowitego zakończenia treningu. Gdy wszystkie fazy zostały wczytane, z kolejki faz usuwana jest właśnie zakończona faza, a na jej miejsce dodawana jest wartość *null*, reprezentująca fazę końcową. Licznik *_stopTimer* określa, ile faz pozostało do momentu uruchomienia sekwencji zakończenia treningu. Jeżeli *_stopTimer*

osiągnie zero, inicjowana jest faza końcowa. Jednocześnie aktywowana flaga *skipFirstCounting*, aby uniknąć niepożądanego odtworzenia dźwięku odliczającego w fazie końcowej. W przypadku aktywnej playlisty treningowej wywoływana jest metoda *completePlaylist()*, która zapewnia kontrolowane zakończenie odtwarzania i zwolnienie zasobów. Etykiety w interfejsie użytkownika są ukrywane, a metoda *_playEndingSound* uruchamia właściwą ścieżkę dźwiękową. W przeciwnym razie, gdy *_stopTimer* nie osiągnął zera, następuje przejście do kolejnej fazy treningu właściwego. Jeśli faza końcowa jest już aktywna (*_endingInitiated*) i właśnie dobiegła końca, ustawiana jest flaga *end*, sygnalizująca zakończenie treningu, a ekran zostaje zamknięty, kończąc całą sesję. Fragment przedstawiono w listingu kodu 6.14.

```
if (_remainingTime == 0) {
  if (_endingInitiated){
    second.value = 0;
    end = true;
    Navigator.pop(_context);
  }
  if (_finishedLoadingBreathingPhases) {
    tryUpdateStageCounter();
    final removedPhase = breathingPhasesQueue.value.removeFirst();
    breathingPhasesQueue.value.add(null);
    breathingPhasesQueue.value =
        Queue<breathing_phase.BreathingPhase?>.from(
            breathingPhasesQueue.value);
    _stopTimer--;

    if (_stopTimer == 0) {

      _endingInitiated = true;
      soundManager.stopSound(_currentSound);
      _playShortSound(
        parser.training.sounds.stageChangeSound.name);
      _remainingTime = _endingDuration;
      _currentSound = _sounds.endingTrack.name;
      previousSecond = (_remainingTime ~/ 1000)+1;
      skipFirstCounting = true;

      if (_isUsingPlaylist) {
        playlistManager.completePlaylist();
        _isUsingPlaylist = false;
      }

      showLabels.value = false;
      _playEndingSound(_sounds.endingTrack.name, 500);

    } else {
      _remainingTime = _nextRemainingTime;
      previousSecond = (_remainingTime ~/ 1000)+1;
      skipFirstCounting = true;
    }
  }
```

Listing 6.14. TrainingController - koniec treningu

Gdy zakończenie bieżącej fazy nie powoduje jeszcze uruchomienia sekwencji końcowej, realizowane jest przejście do kolejnej fazy treningu właściwego. W tym celu z kolejki faz usuwana jest właśnie zakończona faza, a wartość pozostałego czasu zostaje ustawiona na wcześniej obliczony czas trwania kolejnej fazy. Bezpośrednio potem wywoływana jest metoda *_fetchNextBreathingPhase()*, która asynchronicznie wczytuje i przygotowuje dane kolejnej fazy, umożliwiając ciągłe wyprzedzenie w przetwarzaniu. Na koniec wywoływana jest metoda *_updateCurrentTrainingStageLabel()*, zapewniająca natychmiastowe odświeżenie etykiety aktualnego etapu w interfejsie użytkownika. Fragment przedstawiono w listingu 6.15.

```
else {
  tryUpdateStageCounter();
  final removedPhase = breathingPhasesQueue.value.removeFirst();
  _remainingTime = _nextRemainingTime;
  if (_trainingStageNameQueue.isNotEmpty) {
    _trainingStageNameQueue.removeFirst();
  }
  if (_trainingStageIdQueue.isNotEmpty) {
    _trainingStageIdQueue.removeFirst();
  }

  _nextRemainingTime = _newBreathingPhaseRemainingTime;
  previousSecond = (_remainingTime ~/ 1000)+1;
  skipFirstCounting = true;
  _fetchNextBreathingPhase();
  _nextPhaseSoundPlayed = false;
  breathingPhasesQueue.value =
      Queue<breathing_phase.BreathingPhase?>.from(
          breathingPhasesQueue.value);
  _updateCurrentTrainingStageLabel();
}
```

Listing 6.15. TrainingController - rozpoczęcie treningu

6.4.3. Klasa AnimatedCircle

Komponent `AnimatedCircle` odpowiada za wizualizację przebiegu fazy oddechowej w postaci animowanego koła. Wdech powoduje zwiększanie jego promienia, wydech - zmniejszanie, natomiast fazy retencji i regeneracji utrzymują stały rozmiar.

Obiekt przyjmuje dwa parametry — obiekt typu `BreathingPhase?`, który reprezentuje aktualną fazę oddechową bądź wartość null w przypadku zakończenia treningu oraz obiekt typu `bool` *isPaused*, reprezentujący stan wstrzymania treningu. Framgment tworzenia klasy przedstawiony został w listingu 6.16.

```
  final breathing_phase.BreathingPhase? breathingPhase;
  final bool isPaused;

  const AnimatedCircle({super.key, required this.breathingPhase,
     required this.isPaused});
```

Listing 6.16. AnimatedCircle - konstruktor

Po utworzeniu obiektu na początku zostaje obliczona wartość początkowa czasu trwania animacji, która zapisywana jest w zmiennej *duration* na podstawie czasu trwania danej fazy. Następnie inicjowany jest kontroler animacji *controller* oraz animacja zmiany promienia koła *circleAnimation*. Kontroler ustawiany jest na stan początkowy (*controller.value=0.0*). Jeżeli trening nie jest wstrzymany i dostępna jest faza oddechowa, uruchamiana jest odpowiednia animacja — rosnąca dla wdechu lub malejąca dla wydechu. Fragment funkcji init przedstawiono w listingu 6.17.

```
@override
void initState() {
  super.initState();

  duration = widget.breathingPhase == null ? 0 :
      (widget.breathingPhase!.duration * 1000).toInt();

  _controller = AnimationController(
    duration: Duration(milliseconds: duration),
    vsync: this,
  );

  _circleAnimation = Tween<double>(begin: 125.0, end:
      300.0).animate(
    CurvedAnimation(parent: _controller, curve:
        Curves.easeInOut),
  );
  _controller.duration = Duration(milliseconds: duration);

  _controller.value = 0.0;

  if (!widget.isPaused && widget.breathingPhase != null) {
    if (widget.breathingPhase!.breathingPhaseType ==
        breathing_phase.BreathingPhaseType.inhale) {
      _controller.forward(from: 0.0);
    } else if (widget.breathingPhase!.breathingPhaseType ==
        breathing_phase.BreathingPhaseType.exhale) {
      _controller.reverse(from: 1.0);
    }
  }
}
```

Listing 6.17. AnimatedCircle - fragment funkcji init

Zachowanie koła zależne jest od jego poprzedniego stanu i zmiany parametrów wejściowych. Jeżeli faza oddechowa się zmieniła, względem ostatniego stanu następuje reakcja zmiany animacji. Ponownie obliczany jest czas trwania animacji (zmienna *duration*) i ustawiane zostaje poprawne zachowanie — wzrost promienia dla wdechu i jego zmniejszenie dla wydechu. Aktualizację stanu przedstawiono w listingu 6.18

Koło `AnimatedCircle` reaguje również na zmianę parametru *isPaused*, która określa czy trening został zatrzymany. Jeśli tak, animacja zostaje wstrzymana, w innym wypadku, jeśli trening był zatrzymany i został wznowiony, animacja zostaje kontynuowana w kierunku wynikającym z bieżącej fazy. Fragment ten przedstawiono w listingu 6.19.

```
    if (widget.breathingPhase != oldWidget.breathingPhase &&
        widget.breathingPhase != null) {
      log("${widget.breathingPhase?.breathingPhaseType.name}");

      duration = widget.breathingPhase == null ? 0 :
          (widget.breathingPhase!.duration * 1000).toInt();
      _controller.duration = Duration(milliseconds: duration);

      if (!widget.isPaused && widget.breathingPhase != null) {
        if (widget.breathingPhase!.breathingPhaseType ==
            breathing_phase.BreathingPhaseType.inhale) {
          _controller.forward(from: 0.0);
        } else if (widget.breathingPhase!.breathingPhaseType ==
            breathing_phase.BreathingPhaseType.exhale) {
          _controller.reverse(from: 1.0);
        }
      } else {
        _controller.stop();
      }
    }
```

Listing 6.18. AnimatedCircle - aktualizacja stanu

```
if (widget.isPaused && !oldWidget.isPaused) {
      _controller.stop();
    } else if (!widget.isPaused && oldWidget.isPaused) {
      if (widget.breathingPhase != null) {
        if (widget.breathingPhase!.breathingPhaseType ==
            breathing_phase.BreathingPhaseType.inhale) {
          _controller.forward();
        } else if (widget.breathingPhase!.breathingPhaseType ==
            breathing_phase.BreathingPhaseType.exhale) {
          _controller.reverse();
        }
      }
    }
```

Listing 6.19. AnimatedCircle - reakcja na pauzę i wznowienie

Dodatkowo utworzony został obiekt łączący `AnimatedCircle` oraz dwa koła statyczne typu `Container` wizualizujące maksymalną i minimalną wartość promienia koła `AnimatedCircle`, by umożliwić użytkownikowi lepiej ocenić przebieg wdechu i wydechu.

6.4.4. Klasa InstructionSlider

`InstructionSlider` to animowany komponent odpowiadający za wizualizację ciągu instrukcji dla użytkownika w postaci trzech kafelków reprezentujących:

- poprzednią fazę oddechową,
- obecną fazę oddechową,
- nadchodzącą fazę oddechową.

Gdy zachodzi zmiana fazy, cała lista przesuwa się w lewo, a na końcu pojawia się nowy kafelek z kolejną instrukcją.

Obiekt przyjmuje trzy parametry — *preparationTime*, czyli czas trwania fazy przygotowania przed treningiem, kolejkę *breathingPhasesQueue*, która jest ciągiem faz oddechowych w treningu oraz *change*, która jest wyznacznikiem zmiany obecnie trwającej fazy. Fragment tworzenia klasy przedstawiony został w listingu 6.20.

```
double preparationTime;
Queue<breathing_phase.BreathingPhase?> breathingPhasesQueue =
    Queue<breathing_phase.BreathingPhase?>();
int change;

InstructionSlider({super.key,required this.preparationTime,
    required this.breathingPhasesQueue, required this.change});
```

Listing 6.20. InstructionSlider - konstruktor

Podczas inicjalizacji obiekt tworzy pierwszy kafelek reprezentujący fazę przygotowania. Następnie dodawane są dwie fazy z kolejki faz treningu w tej samej formie. Każdy kafelek przechowuje tekst w postaci instrukcji dla użytkownika i informację, na której pozycji ma się znaleźć. Fragment dodawania obiektów przedstawiono w listingu 6.21.

```
_blocks.add(
  InstructionBlock(
    text: translationProvider.getTranslation(
      "BreathingPage.InstructionSlider.get_ready_block_text") +
         "\n${widget.preparationTime} s",
    position: 0.0)
);

addNewBreathingPhase(widget.breathingPhasesQueue.elementAt(1));
addNewBreathingPhase(widget.breathingPhasesQueue.elementAt(2));
```

Listing 6.21. InstructionSlider - fragment funkcji init

`InstructionSlider` korzysta z `AnimationController`, który przy każdej zmianie fazy przesuwa wszystkie kafelki o jedną pozycję w lewo. Każdy kafelek ma przypisaną pozycję z przedziału od -2 do 2. Wartości skrajne (-2 oraz 2) znajdują się poza obszarem widocznym dla użytkownika i służą jedynie do zapewnienia płynniejszej i estetyczniejszej animacji. W centrum ekranu znajdują się trzy środkowe pozycje: -1 odpowiada fazie poprzedniej, 0 fazie bieżącej, a 1 fazie nadchodzącej. Gdy kafelek na pozycji -2 i ma zostać przesunięty, zostaje usunięty w celach optymalizacyjnych. Dokładną implementację animacji przedstawiono w listingu 6.22. Dodatkowo wykorzystywany jest mechanizm skalowania środkowego kafelka w celu wizualnego podkreślenia aktualnie trwającej fazy.

Każda zmiana fazy powoduje uruchomienie aktualizacji stanu komponentu. Mechanizm opiera się na porównaniu poprzedniej i bieżącej wartości parametru *change*. Dzięki temu komponent wykrywa moment rozpoczęcia nowej fazy. Obliczany jest wtedy czas trwania nowej fazy, wywoływana jest animacja oraz dodawany nowy kafelek z instrukcją. Funkcję aktualizacji przedstawiono w listingu kodu 6.23.

```
_controller = AnimationController(vsync: this, duration:
    duration);
_animation = Tween<double>(begin: 0.0, end: -1.0).animate(
  CurvedAnimation(parent: _controller, curve: Curves.easeInOut),
);
_controller.addStatusListener((status) {
  if (status == AnimationStatus.completed) {
    _controller.reset();
    setState(() {
      final removed = _blocks.where((b) => b.position <=
          -2).toList();
      _blocks.removeWhere((b) => b.position <= -2);
      _blocks.forEach((b) => b.position += _animation.value);
      _blocks.forEach((b) => b.position -= 1);
    });
  }
});
```

Listing 6.22. InstructionSlider - animacja

```
if(oldWidget.change != widget.change) {
  final int phaseDuration =
      (widget.breathingPhasesQueue.elementAt(0)?.duration.toInt()
      != null)
    ? (widget.breathingPhasesQueue.elementAt(0)!.duration *
        1000).toInt() - 50
    : 400;
  _controller.duration = Duration(milliseconds:
      min(phaseDuration,400));
  _controller.forward();
  if(_blocks.last.text!=translationProvider.getTranslation(
    "BreathingPage.InstructionSlider.ending_tile_text")) {
    addNewBreathingPhase(widget.breathingPhasesQueue.elementAt(2));
  }
}
```

Listing 6.23. InstructionSlider - aktualizacja stanu

6.5. *Dźwięki (Jakub Romanowski)*

Wysoki stopień konfigurowalności warstwy audio przez użytkownika i chęć postępowania zgodnie z *Single Responsibility Principle* (pol. zasadą pojedynczej odpowiedzialności), sformułowaną przez R. C. Martina [24], wymusiły dekompozycję logiki biznesowej aplikacji na klasy odpowiedzialne za poszczególne przypadki użycia. Podział ten reprezentuje rysunek 6.4.

Rysunek 6.4. Struktura klas obsługi dźwięków

6.5.1. SoundManager

Centralnym elementem modułu audio jest klasa `SoundManager`. Ze względu na konieczność zapewnienia globalnego punktu dostępu do listy załadowanych plików oraz potrzebę współdzielenia jednej instancji obiektu przez wiele komponentów aplikacji, zastosowano w niej wzorzec projektowy Singleton. Obiekt ten agreguje kluczowe struktury danych: mapy powiązań odtwarzaczy oraz zbiór dostępnych plików dźwiękowych. Ten ostatni został podzielony na dwie odrębne kolekcje dedykowane odpowiednio krótkim i długim formom audio.

Rozróżnienie na typy dźwięków było niezbędne ze względu na odmienną logikę ładowania i odtwarzania plików. Krótkie efekty dźwiękowe wymagają minimalnej latencji (czasu reakcji) przy odtwarzaniu i częstych powtórzeń, natomiast długie ścieżki dźwiękowe charakteryzują się rzadszą rotacją i mniejszą dynamiką zmian, przez co nie podlegają tak rygorystycznym wymogom

czasowym przy inicjalizacji. Do osiągnięcia tego wymagania należało zastosować oddzielne konfiguracje odtwarzaczy dźwięków (komponentów klasy `AudioPlayer`) — *low latency mode* (pol. tryb minimalnego czasu reakcji) lub *media player mode* (pol. tryb odtwarzania media).

Mimo zastosowania dedykowanych trybów, rozwiązanie to napotkało szereg barier technologicznych. Po pierwsze, tryb minimalnego czasu reakcji nie udostępniał mechanizmu zdarzeń, co uniemożliwiało detekcję momentu zakończenia odtwarzania. Z kolei własna implementacja powodowała wysoki narzut obliczeniowy, uniemożliwiając emisję sygnałów dźwiękowych z wymaganą precyzją. Ograniczenia te wymusiły wprowadzenie mechanizmu puli obiektów — `AudioPlayerPool` opisanego w rozdziale 6.5.4.

Kluczowym aspektem konfiguracji było również zapewnienie współbieżności emisji, umożliwiającej równoczesne odtwarzanie wielu nakładających się na siebie dźwięków. W połączeniu z mechanizmem płynnych przejść, który obejmuje obsługę efektów *fade-in* (pol. stopniowego podgłaśniania przy rozpoczynaniu odtwarzania) oraz *fade-out* (pol. wyciszania przy jego zatrzymywaniu), rozwiązanie to zapewnia profesjonalną i przyjemną w odbiorze warstwę audio.

6.5.2. SingleSoundManager

Klasa `SingleSoundManager` dedykowana jest do obsługi odtwarzania sygnałów w trybie wyłączności. W warstwie implementacyjnej deleguje ona zadania do globalnego `SoundManagera` opisanego w rozdziale 6.5.1, jednak w przeciwieństwie do niego, uniemożliwia współbieżne odtwarzanie wielu ścieżek — w danym momencie odtwarzany może być tylko jeden plik.

Informacja o aktywnym zasobie przechowywana jest w polu `currentlyPlaying`. Co istotne, odtwarzanie realizowane jest tutaj wyłącznie w trybie *media player*. Pozwala to na nasłuchiwanie zdarzeń zakończenia emisji i automatyczne czyszczenie stanu zmiennej. Mechanizm ten znajduje zastosowanie m.in. podczas wyboru dźwięków w edytorze treningu (w dodatku A, rozdział A.29), umożliwiając dynamiczną aktualizację interfejsu użytkownika w zależności od stanu odtwarzacza.

6.5.3. PlaylistManager

Za obsługę sekwencyjnego odtwarzania zbiorów plików muzycznych w trakcie treningu odpowiada klasa `PlaylistManager`. Enkapsuluje ona stan procesu, przechowując flagi sterujące (typu logicznego, `boolean`), kursor bieżącego utworu oraz listę ścieżek. Istotnym elementem jest również obiekt typu `Completer`, pełniący rolę prymitywu synchronizacyjnego służącego do obsługi zdarzeń przerwania odtwarzania.

Do realizacji swoich funkcji klasa deleguje zadania do `SingleSoundManagera` (rozdział 6.5.2). Decyzja ta podyktowana jest koniecznością precyzyjnej detekcji momentu zakończenia utworu, co jest natywną funkcjonalnością wykorzystywanego modułu. Mechanizm przełączania ścieżek opiera się na asynchronicznym oczekiwaniu na pierwsze z dwóch zdarzeń: naturalne zakończenie odtwarzania przez odtwarzacz lub wymuszone przerwanie (sygnalizowane przez `_playlistCompleter`). Logikę tę obrazuje listing 6.24.

```
1  await Future.any([player.onPlayerComplete.first,
       _playlistCompleter.future]);
2  if(_paused || _stopped) {
3      dev.log('PlaylistManager: Playback interrupted
           (paused=$_paused, stopped=$_stopped)');
4      return;
5    }
6
7    _currentIndex++;
8    _playNextSound();
```

Listing 6.24. PlaylistManager - obsługa przerwania lub zakończenia dźwięku

6.5.4. AudioPlayerPool

Implementacja klasy `AudioPlayerPool` wynika bezpośrednio ze zidentyfikowanych ograniczeń technicznych — braku obsługi zdarzeń zakończenia odtwarzania w trybie minimalnego czasu reakcji. Celem tego modułu jest umożliwienie emisji krótkich sygnałów dźwiękowych o wysokiej częstotliwości (w odstępach rzędu 100 ms).

Konstruktor klasy przyjmuje parametr całkowitoliczbowy, definiujący rozmiar bufora cyklicznego, w którym przechowywane są instancje odtwarzaczy. Mechanizm pobierania zasobu z puli realizuje strategię rotacji i odświeżania: obiekt znajdujący się pod bieżącym wskaźnikiem podlega zniszczeniu i na jego miejsce tworzony jest nowy. Podejście to jest kluczowe przy kolejnych obiegach listy, aby wyeliminować błędy wynikające z próby ponownego użycia odtwarzacza posiadającego zablokowany stan po poprzednim odtworzeniu. Logikę pobierania instancji przedstawia listing kodu 6.25.

```
1  AudioPlayer get next {
2    _cursor++;
3    if (_cursor >= _items.length) {
4      _cursor = 0;
5    }
6    _items[_cursor].dispose();
7    _items[_cursor] = AudioPlayer();
8    return _items[_cursor];
9  }
```

Listing 6.25. AudioPlayerPool - pobieranie kolejnego elementu z puli

6.6. Języki (Jakub Romanowski)

Interfejs użytkownika został przystosowany do obsługi wielu wersji językowych, oferując obecnie pełne wsparcie dla języka polskiego oraz angielskiego. Zasoby tekstowe przechowywane są w zewnętrznych plikach formatu JSON, umieszczonych w katalogu *assets/*. Przyjęta konwencja nazewnictwa plików opiera się na standardowych kodach lokalizacji (np. `pl_PL.json`).

Za warstwę logiczną odpowiada serwis `TranslationProvider`. Przechowuje on w pamięci operacyjnej stan wybranego języka oraz załadowany słownik tłumaczeń w strukturze `Map<String, dynamic>` zachowując pierwotną, hierarchiczną formę. Pobieranie fraz realizuje metoda `getTranslation`, która interpretuje argument wejściowy jako ścieżkę nawigacji w strukturze JSON (gdzie poszczególne węzły rozdzielone są separatorem kropki, np. `"home.title"`). Algorytm dzieli przekazany klucz, a następnie przechodzi przez kolejne poziomy mapy w poszukiwaniu docelowego tekstu. Jeśli wskazana ścieżka jest poprawna, przypisana do niej wartość zostaje zwrócona. W przeciwnym razie (np. błędny klucz lub brak węzła) system zwraca pierwotny klucz, co stanowi mechanizm zabezpieczający. Implementację tej logiki przedstawiono w listingu 6.26, natomiast przykładową strukturę danych wykorzystywaną przez moduł tłumaczeń zaprezentowano w listingu 6.27.

```
String getTranslation(String key) {
  if (_translations.isEmpty) {
    return key;
  }

  List<String> parts = key.split('.');
  dynamic value = _translations;

  for(String part in parts){
    if (value is Map<String, dynamic> &&
        value.containsKey(part)) {
      value = value[part];
    } else {
      return key;
    }
  }

  return value is String ? value : key;
}
```

Listing 6.26. TranslationProvider - pobieranie tłumaczenia

```
{
   "SoundsTab":{
      "TrainingSounds":{
        "title": "Training sounds",
        "counting_sound": "Counting",
        "NextBreathingPhaseSounds":{
          "title":"Next breathing phase: ",
          "none": "no sound",
          "global":"Global phase change sound",
          "voice":"lector",
          "for_each_breathing_phase":"change sound for each phase"
        },
        "stage_change_sound": "End of stage",
        "cycle_change_sound": "End of cycle"
      },
    }
}
```

Listing 6.27. Fragment struktury pliku tłumaczeń (en_EN.json)

6.7. Generator list dźwięków (Aleksandra Bujny)

W celu automatyzacji pracy i wyeliminowania ryzyka błędów z powodu ręcznego wprowadzania nazw oraz ścieżek do plików dźwiękowych, stworzony został prosty generator list dźwięków. Został on umieszczony w folderze *assets/*, w podfolderze *sounds/*, w pliku *generator.dart*. Narzędzie to generuje plik *Sounds_lists.g.dart* zapisywany w folderze *lib/* w podkatalogu *components/Global*. Zawiera on klasę `ReSpireSounds`, w której znajdują się trzy mapy z kluczem będącym łańcuchem znaków reprezentującym nazwę dźwięku oraz wartością będącą typu `SoundAsset`. Ma on parametry takie jak nazwa, ścieżka do pliku oraz rodzaj dźwięku podawane w konstruktorze. Wyróżnione zostały trzy rodzaje dźwięków opisywane przez typ wyliczeniowy `SoundType`: krótkie efekty dźwiękowe (`SoundType.cue`), dłuższe nagrania (`SoundType.melody`) oraz dźwięki odliczania (`SoundType.counting`). Każda z trzech map grupuje dźwięki konkretnego typu. Przykładowy fragment wygenerowanego pliku przedstawiono w listingu kodu 6.28.

```
  final Map<String, SoundAsset> countingSounds = {
    "Clock": SoundAsset(name: "Clock", path:
        "sounds/counting/Clock.mp3", type: SoundType.counting),
    "Metronome": SoundAsset(name: "Metronome", path:
        "sounds/counting/Metronome.mp3", type:
        SoundType.counting),
  };
```

Listing 6.28. Mapa dźwięków odliczania wygnerowana automatycznie

Uniknięcie powtarzalności kodu generatora zostało osiągnięte dzięki użyciu dodatkowej struktury - klasy `SoundsInfo` przedstawionej w listingu 6.29. Zawiera ona trzy pola typu tekstowego: *soundType* określające typ dźwięku, *directory* przechowujące nazwę podfolderu zawierającego dźwięk danego typu oraz *mapName* definiujące mapę dedykowaną temu typowi. Tworzona jest lista obiektów tej klasy, dzięki czemu można wykonać jedną pętlę po trzech różnych typach dźwięków, zmieniając jedynie parametry.

```
class SoundsInfo {
  final String soundType;
  final String directory;
  final String mapName;
}
```

Listing 6.29. Klasa `SoundsInfo`

W celu wywołania generatora należy uruchomić w terminalu nastepujące polecenie: `dart run assets/sounds/generator.dart`, będąc w katalogu głównym projektu (folder *ReSpire/*). Pliki dźwiękowe muszą być umieszczone w odpowiednich podfolderach w katalogu *assets/sounds/*, aby zostały poprawnie wykryte przez generator. Są to kolejno: *cues/* dla krótkich efektów dźwiękowych, *melodies/* dla dłuższych nagrań oraz *counting/* dla dźwięków odliczania, analogicznie jak w przypadku struktury map w wygenerowanym pliku. Istotne jest, aby pliki miały odpowiednie nazwy - są one później wykorzystywane jako nazwa tworzonego obiektu klasy `SoundAsset` oraz klucz w mapie, tak jak przedstawiono we fragmencie kodu 6.30. Znaki - oraz _ w nazwach plików są automatycznie zamieniane na spacje, aby zapewnić czytelność nazw dźwięków w interfejsie użytkownika, a pierwsza litera nazwy jest zamieniana na wielką w razie potrzeby. Dokonuje tego funkcja `capitalizeAndRemoveTextSeparators()` pochodząca z klasy `TextUtils`. Linijki tworzące kolejne wpisy w mapie są zapisywane do bufora, a następnie cały bufor jest zapisywany do pliku docelowego.

```
  for (var name in filesList[i]) {
      final displayName =
         TextUtils.capitalizeAndRemoveTextSeparators(name);
      buffer.writeln(
          '    "$displayName": SoundAsset(name: "$displayName",
             path: "sounds/${sounds.directory}/$name.mp3", type:
             SoundType.${sounds.soundType}),');
    }
```

Listing 6.30. Proces wpisywania do bufora linijek tworzących kolejne wpisy w mapie

6.8. Baza danych (Karol Zwierz)

W celu zapewnienia persystencji danych użytkownika w naszej aplikacji mobilnej zaimplementowaliśmy lokalną bazę danych. Zważywszy na wybrane wcześniej środowisko Flutter, wymaganie wspierania wielu platform oraz wysoką gęstość zapisu danych, wybraliśmy napisaną w języku Dart nierelacyjną bazę Hive w wersji 2.2.3. Jest to lekka baza typu *key-value store*, która przechowuje dane w formacie binarnym bezpośrednio na urządzeniu użytkownika, bez konieczności łączenia się z serwerem zewnętrznym.

6.8.1. Architektura i generowanie kodu

Hive wykorzystuje mechanizm generowania kodu adapterów zajmujących się niskopoziomowymi operacjami na plikach bazy danych. Dzięki bardzo dobrej integracji z językiem Dart umożliwia łatwe odwzorowanie klas w bazie danych poprzez adnotacje `@HiveType()` oraz `@HiveField()`. Adnotacja `@HiveType(typeId: n)` oznacza klasę jako typ możliwy do serializacji, gdzie `typeId` musi być unikalnym identyfikatorem liczbowym. Adnotacja `@HiveField(n)` oznacza poszczególne pola klasy, które mają być zapisywane do bazy.

Listing 6.31 przedstawia przykładową klasę `Settings` z adnotacjami Hive:

```
import 'package:hive_flutter/hive_flutter.dart';

part 'Settings.g.dart';

@HiveType(typeId: 8)
class Settings {

  @HiveField(0)
  int preparationDuration = 3;

  @HiveField(1)
  int endingDuration = 5;

  @HiveField(2)
  bool binauralBeatsEnabled = false;

  @HiveField(3)
  double binauralLeftFrequency = 200.0;

  @HiveField(4)
  double binauralRightFrequency = 210.0;
}
```

Listing 6.31: Klasa Settings z adnotacjami Hive

Po oznaczeniu klasy danych adnotacjami i uruchomieniu polecenia `flutter pub run build_runner build`, generator automatycznie tworzy pliki z rozszerzeniem `.g.dart` zawierające klasy `TypeAdapter`. Adaptery te implementują logikę serializacji i deserializacji obiektów do formatu binarnego:

- metoda `read()` – odczytuje dane binarne z bazy i odtwarza obiekt Dart,
- metoda `write()` – przetwarza obiekt Dart na ciąg bajtów do zapisu.

Listing 6.32 przedstawia fragment automatycznie wygenerowanego adaptera dla klasy `Settings`:

```
part of 'Settings.dart';

class SettingsAdapter extends TypeAdapter<Settings> {
  @override
  final int typeId = 8;

  @override
  Settings read(BinaryReader reader) {
    final numOfFields = reader.readByte();
    final fields = <int, dynamic>{
      for (int i = 0; i < numOfFields; i++)
        reader.readByte(): reader.read(),
    };
    return Settings()
      ..preparationDuration = fields[0] as int
      ..endingDuration = fields[1] as int
      ..binauralBeatsEnabled = fields[2] as bool
      ..binauralLeftFrequency = fields[3] as double
      ..binauralRightFrequency = fields[4] as double;
  }

  @override
  void write(BinaryWriter writer, Settings obj) {
    writer
      ..writeByte(5)
      ..writeByte(0)
      ..write(obj.preparationDuration)
      ..writeByte(1)
      ..write(obj.endingDuration)
      ..writeByte(2)
      ..write(obj.binauralBeatsEnabled)
      ..writeByte(3)
      ..write(obj.binauralLeftFrequency)
      ..writeByte(4)
      ..write(obj.binauralRightFrequency);
  }


  @override
  int get hashCode => typeId.hashCode;

  @override
```

```
43   bool operator ==(Object other) =>
44       identical(this, other) ||
45       other is SettingsAdapter &&
46           runtimeType == other.runtimeType &&
47           typeId == other.typeId;
48 }
```

Listing 6.32: Wygenerowany SettingsAdapter

6.8.2. *Model danych*

W aplikacji zdefiniowaliśmy następujące typy danych przechowywane w bazie Hive:

- `Training` (typeId: 1) – kompletny trening oddechowy zawierający tytuł, opis, listę etapów, ustawienia dźwięków oraz konfigurację,
- `TrainingStage` (typeId: 2) – pojedynczy etap treningu z liczbą powtórzeń, przyrostem czasowym oraz listą faz oddechowych,
- `BreathingPhaseType` (typeId: 3) – typ wyliczeniowy określający rodzaj fazy oddechowej (wdech, wydech, zatrzymanie, regeneracja),
- `BreathingPhase` (typeId: 4) – faza oddechowa z określonym czasem trwania, typem i przypisanymi dźwiękami,
- `BreathingPhaseIncrementType` (typeId: 5) – typ wyliczeniowy określający sposób inkrementacji czasu (procentowy lub stała wartość),
- `BreathingPhaseIncrement` (typeId: 6) – konfiguracja przyrostu czasowego dla poszczególnych faz,
- `Sounds` (typeId: 7) – kontener przechowujący kompletne ustawienia dźwiękowe dla treningu,
- `Settings` (typeId: 8) – globalne ustawienia treningu, w tym konfiguracja dudnień binauralnych,
- `BreathingPhaseSounds` (typeId: 9) – zestaw dźwięków przypisanych do konkretnej fazy oddechowej (dźwięk tła i wskazówka rozpoczęcia),
- `SoundAsset` (typeId: 10) – reprezentacja zasobu dźwiękowego z informacją o ścieżce pliku i jego typie,
- `SoundType` (typeId: 11) – typ wyliczeniowy kategoryzujący dźwięki (np. głos, melodia, sygnał),
- `SoundScope` (typeId: 12) – typ wyliczeniowy definiujący zakres obowiązywania tła muzycznego (np. globalnie, per etap).

6.8.3. *Organizacja danych w Box'ach*

Hive organizuje dane w strukturach zwanych *Box*'ami, które odpowiadają tabelom w bazach relacyjnych lub kolekcjom w bazach dokumentowych. Każdy Box przechowuje pary klucz-wartość i musi zostać otwarty przed użyciem. W naszej aplikacji zdefiniowaliśmy cztery Box'y:

- `respire` – główny Box przechowujący presety treningowe oraz ich wersję migracyjną,
- `userShortSounds` – Box na krótkie dźwięki dodane przez użytkownika,
- `userLongSounds` – Box na długie dźwięki tła dodane przez użytkownika,
- `userCountingSounds` – Box na dźwięki odliczania dodane przez użytkownika.

6.8.4. *Inicjalizacja bazy danych*

Inicjalizacja bazy danych odbywa się w funkcji `initialize()` wywoływanej przy starcie aplikacji. Proces ten obejmuje:

1. Inicjalizację silnika bazy danych – `Hive.initFlutter()`
2. Rejestrację wszystkich adapterów typów – `Hive.registerAdapter()`
3. Otwarcie wszystkich Box'ów – `Hive.openBox()`.

Została przedstawiona w listingu 6.33.

```
Future<void> initialize() async {
  await Hive.initFlutter();

  Hive.registerAdapter(BreathingPhaseAdapter());
  Hive.registerAdapter(BreathingPhaseTypeAdapter());
  Hive.registerAdapter(TrainingStageAdapter());
  Hive.registerAdapter(TrainingAdapter());
  Hive.registerAdapter(SoundAssetAdapter());
  Hive.registerAdapter(SoundsAdapter());
  Hive.registerAdapter(SettingsAdapter());
  Hive.registerAdapter(BreathingPhaseSoundsAdapter());

  await Hive.openBox('respire');
  await Hive.openBox('userShortSounds');
  await Hive.openBox('userLongSounds');
  await Hive.openBox('userCountingSounds');
}
```

Listing 6.33: Inicjalizacja bazy danych Hive

Operacje na danych wykonywane są poprzez metody Box'a: `put(key, value)` do zapisu, `get(key)` do odczytu oraz `delete(key)` do usuwania. Przykładowe operacje na danych - w tym przypadku ładowania treningów i dodawanie ich do listy presetów - pokazano w listingu kodu 6.34.

```
final _box = Hive.box('respire');

void loadData() {
  final stored = _box.get('presets');
  if (stored != null) {
    presetList = List<Training>.from(stored);
  }
}

void updateDataBase() {
  _box.put('presets', presetList);
}
```

Listing 6.34: Operacje na danych w Hive

Dzięki generowanym adapterom Hive automatycznie serializuje złożone obiekty wraz z ich zagnieżdżonymi strukturami, co umożliwia zapisanie całego drzewa obiektu `Training` wraz ze wszystkimi etapami, fazami i dźwiękami w jednej operacji.

7. TESTOWANIE APLIKACJI (Hanna Banasiak)

W tym rozdziale przedstawione zostały sposoby testowania tworzonej aplikacji mobilnej. Terminologia i podział technik testowych są zgodne z aktualnym syllabusem ISTQB [25] (*ang. International Software Testing Qualifications Board*) Foundation Level (wersja 4.0) oraz polskimi odpowiednikami. Ze względu na mały zespół i charakter projektu większość czynności miała charakter manualny i opierała się na wzajemnej weryfikacji członków zespołu i opiekuna pracy. Opis przeprowadzonego testowania podzielono na dwie części — testowanie statyczne i testowanie dynamiczne.

7.1. Testowanie statyczne

Testowanie statyczne stanowiło jeden z kluczowych elementów zapewnienia jakości w niniejszym projekcie. W odróżnieniu od testowania dynamicznego, wszystkie czynności statyczne realizowano bez konieczności uruchamiania kodu wykonawczego. Zamiast tego analizie poddane były artefakty wytwarzane na każdym etapie, w tym: kod źródłowy, modele i diagramy oraz wymagania. Dzięki temu możliwe było wczesne wykrycie błędów, niezgodności, problemów z jakością czy naruszenia standardów.

Główną formę testowania statycznego stosowanego w projekcie stanowiły *Code Review* (pol. przeglądy kodu). Każdy fragment kodu, niezależnie od tego, czy była to nowa funkcjonalność czy drobna poprawka, musiał przejść nieformalny przegląd przez co najmniej jedną, a najczęściej dwie inne osoby z zespołu. Czynności te odbywały się na bieżąco po każdym przyroście przed *merge* (pol. dołączeniem) do głównej gałęzi roboczej *dev*. Ich celem było wykrycie błędów przed połączeniem z już działającym kodem.

Często odbywały się również przeglądy techniczne, których celem było podjęcie decyzji w sprawie problemu technicznego. W spotkaniach tego typu brał udział cały zespół, co skutkowało większą liczbą wykreowanych pomysłów i prowadziło do osiągnięcia konsensusu. Zdarzało się jednak, iż przegląd odbywał się w formie *pair reviewing* (pol. przeglądu parami), gdy problem wymagał osób dysponujących specyficznymi kwalifikacjami technicznymi lub problem nie wymagał obecności wszystkich członków zespołu.

Wszystkie inne artefakty, takie jak wymagania, modele czy diagramy były wielokrotnie omawiane i poprawiane w trakcie wspólnych sesji. Często dochodziło do kilku iteracji, aż cały zespół w pełni rozumiał i akceptował przedstawione rozwiązanie. Szczególną uwagę zwracano na spójność, brak sprzeczności oraz jednoznaczność opisów.

Dzięki małemu zespołowi i kulturze otwartej komunikacji każdy mógł w dowolnym momencie zweryfikować kod innego członka zespołu i zadać pytanie, zasugerować zmiany lub wskazać potencjalny problem. Ta nieformalna, ale niezwykle częsta sytuacja była jednym z najskuteczniejszych mechanizmów wczesnego wykrywania defektów.

7.2. Testowanie dynamiczne

Testowanie dynamiczne obejmowało wszystkie czynności wymagające uruchomienia kodu źródłowego aplikacji mobilnej. Ze względu na rozmiar zespołu ograniczono się do testów manualnych, jednak prowadzonych systematycznie i z dużym nakładem czasowym. Skupiono się na tych rodzajach testów, które dają największą wartość w kontekście aplikacji czasu rzeczywistego pracującej bezpośrednio ze sprzętem.

Najważniejszą i najobszerniejszą część testowania dynamicznego stanowiły testy eksploracyjne. Regularnie, po każdym większym przyroście aplikacji i łączeniu zmian na główną gałąź roboczą, co najmniej dwie osoby z zespołu uruchamiały aplikację na fizycznych urządzeniach różniących się od siebie modelem i wersją systemu operacyjnego Android, a także wielkością. Przeprowadzali oni testy równocześnie projektowane i wykonywane w czasie zapoznawania się ze zmianami w oparciu o swoją wiedzę. Pozwoliło to na wykrycie większej ilości błędów interfejsu użytkownika, a także ze względu na indywidualne doświadczenie każdej osoby, wykrywane były błędy nieoczywiste, w tym defekty związane z wartościami brzegowymi, błędy związane z nietypowym wyborem opcji konfiguracyjnych czy problemy z użytkowaniem aplikacji.

Ważnym elementem testów dynamicznych były testy potwierdzające wykonywane po każdej zmianie usuwającej błęd. Ich celem było sprawdzenie, czy pierwotny defekt został pomyślnie usunięty. Wykonywane były wtedy wszystkie te przypadki testowe, które wcześniej nie zostały zaliczone lub tworzone nowe, w celu pokrycia ewentualnych zmian, które były niezbędne do wykonania.

Kolejnym stosowanym typem były manualne testy regresji przeprowadzane przed każdym udostępnieniem nowej wersji aplikacji. Wykonywano pełny, powtarzalny zestaw testów, który obejmował najważniejsze funkcje aplikacji, w tym tworzenie, edycję i przebieg treningu. Dzięki takiemu podejściu udało się znaleźć negatywne konsekwencje spowodowane przez wprowadzane zmiany w innych, nieedytowanych fragmentach oprogramowania.

Ostatnim rodzajem wykonywanych testów były testy akceptacyjne, przeprowadzane z udziałem interesariuszy, takich jak opiekun pracy i potencjalni użytkownicy aplikacji. Każde postanowienia były spisywane, omawiane przez zespół i po zatwierdzeniu, wprowadzane w nowej wersji aplikacji. Testy tego typu pozwoliły na dopracowanie użytkowalności aplikacji i wprowadzenie funkcji opartych na różnych perspektywach i potencjalnych wykorzystaniach aplikacji.

8. PODSUMOWANIE (Aleksandra Bujny, Karol Zwierz)

8.1. Osiągnięte rezultaty (Karol Zwierz)

W trakcie trwania naszego inżynierskiego projektu dyplomowego udało nam sie zrealizować w pełni funkcjonalną aplikację umożliwiającą kompleksową konfigurację skomplikowanych treningów oddechowych. Aplikacja umożliwia importowanie gotowych treningów lub grup treningów, tworzenie treningów, elastyczną konfigurację ich struktury oraz wielowarstwową edycję audio dla każdego elementu treningu. Następnie nasze rozwiązanie umożliwia odtworzenie treningu z wykorzystaniem wszystkich skofngiurowanych wskazówek audiowizualnych lub eksport jednego lub wybranej przez użytkownika grupy treningów do pliku JSON.

8.2. Napotkane problemy i ogarniczenia (Karol Zwierz)

Niestety, z powodu braku dostarczenia nam modelu lub serwisu z modelem do ewaluacji treningu przez nagrania audio użytkownika oraz decyzji o skupieniu się na rozbudowie konfigurowalności, nie udało nam się zaimplementować sprzężenia zwrotnego określającego jakość wykonania przez użytkownika treningu.

8.3. Wnioski z projektu (Karol Zwierz)

Projekt ten nauczył nas realizacji oraz dostarczenia prawdziwego produktu, którego stworzenie wymagało uwzględnienia zarówno potrzeb głównego interesariusza jak i przewidywanych użytkowników końcowych. Pozwolił nam także wykorzystać w praktyce wiedzę teoretyczną wyniesioną z przedmiotów z całego cyklu studiów, ucząc nie tylko syntezy, ale również organizacji w zespołowym procesie twórczym z pomocą metodyk zwinnych. W trakcie realizacji projektu, specyfika procesu wytwórczego oraz równoległe zdobywanie kompetencji technicznych wpłynęły na konieczność redefinicji niektórych założeń początkowych. Weryfikacja przyjętej strategii w fazie implementacji poskutkowała modyfikacją wymagań funkcjonalnych, co w konsekwencji doprowadziło do powstania produktu nieznacznie różniącego się od pierwotnej specyfikacji. Doświadczenia te podkreśliły kluczowe znaczenie etapu analizy przedwdrożeniowej oraz precyzyjnego planowania architektury systemu, co stanowi istotny wniosek dla przyszłych przedsięwzięć inżynierskich.

8.4. Porównanie opracowanego rozwiązania z istniejącymi rozwiązaniami (Aleksandra Bujny)

Wśród ograniczeń zidentyfikowanych w istniejących rozwiązaniach (rozdział 2.2) wskazywano utrudnioną edycję i tworzenie treningów, nawet w aplikacjach deklarujących pełną personalizację. Część z nich nie umożliwiała definiowania długości trwania faz oddechowych, inne narzucały sztywne szablony treningów, nie umożliwiając ich edycji lub ograniczały funkcjonalność użytkownikom bez subskrypcji. Jako mankamenty rozwiązań wymieniano również przestarzały lub nieintuicyjny interfejs, problemy z poprawnym odtwarzaniem dźwięków oraz brak możliwości zatrzymania treningu.

ReSpire oferuje nowoczesny i czytelny interfejs, obsługuje wstrzymywanie sesji oraz eliminuje główny problem związany z ograniczoną elastycznością. Treningi są w pełni konfigurowalne przy pomocy zaawansowanego edytora, który umożliwia dobór typów i długości faz oddechowych oraz ich dowolne zestawianie. Mimo to aplikacja nadaje się zarówno dla osób doświadczonych, jak i początkujących. Zaoferowane gotowe treningi pozwalają na połączenie potrzeb obydwu grup bez konieczności rezygnowania z możliwości konfiguracji. Elementem wyróżniającym ReSpire jest również rozbudowana oprawa dźwiękowa. Użytkownik może wybierać spośród różnych poziomów szczegółowości, ustawiając dźwięki od faz w poszczególnych etapach po muzykę w tle treningu.

Opracowana aplikacja rozwiązuje problemy zauważone w istniejących rozwiązaniach. Choć oferuje przestrzeń do dalszego rozwoju, stanowi przykład podejścia innowacyjnego — wprowadzającego nowe, bardziej elastyczne możliwości.

8.5. Możliwości rozwoju (Aleksandra Bujny)

Aplikacja jest obecnie w pełni funkcjonalna i stabilna, natomiast posiada perspektywy dalszego rozwoju. Jedną z nich jest integracja z modelem oceniającym w czasie rzeczywistym na podstawie nagrań audio dokładności wykonywanych treningów i wyświetlanie ich w formie przyjaznych statystyk. System można by także wzbogacić o mechanizm powiadomień, aby użytkownik pamiętał o regularnych treningach oddechowych, a także wprowadzić system gratyfikacji motywujący do używania ReSpire. Kolejnym potencjalnym obszarem rozwoju jest wprowadzenie innych wersji językowych — obecnie aplikacja obsługuje jedynie polski oraz angielski. Pomimo tego, że aplikacja posiada silną i spójną identyfikację wizualną, warto rozważyć także dodanie motywu ciemnego lub innych motywów kolorystycznych. Etapem pozwalającym udostępnić aplikację szerszemu gronu użytkowników byłoby wprowadzenie jej do sklepu *GooglePlay* i/lub *Apple App Store*, co z kolei stanowiłoby podstawę do dodania funkcjonalności społeczności do ReSpire — na przykład udostępniania treningów między użytkownikami bezpośrednio przez platformę.

BIBLIOGRAFIA

[1] C. A. Group, „Breathing techniques to reduce symptoms in people with serious respiratory illness: a systematic review," *European Respiratory Review*, 2024. dostęp 26 listopada 2025. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC11522968/

[2] V. Prem, R. Sahoo i R. Adela, „Comparison of the effects of Buteyko and pranayama breathing techniques on quality of life in patients with asthma: a randomized controlled trial," *Clinical Rehabilitation*, t. 27, nr 2, s. 133–141, 2013. dostęp 26 listopada 2025.

[3] E. Holloway i R. West, „Integrated breathing and relaxation training (the Papworth method) for adults with asthma in primary care: a randomised controlled trial," *Thorax*, t. 62, nr 12, s. 1036–1042, 2007. dostęp 26 listopada 2025.

[4] Z. Dodange, A. Darvishpour i in., „Comparison of the Effects of Diaphragmatic Breathing and Pursed-lip Breathing Exercises on the Sleep Quality of Elderly Patients with Chronic Obstructive Pulmonary Disease (COPD): A Clinical Trial Study," *Therapeutic Advances in Pulmonary and Critical Care Medicine*, 2024. dostęp 26 listopada 2025. URL: https://journals.sagepub.com/doi/10.1177/29768675241302901

[5] R. Blades, W. B. Mendes i in., „A randomized controlled clinical trial of a Wim Hof Method intervention in women with high depressive symptoms," *Comprehensive Psychoneuroendocrinology*, 2024. dostęp 26 listopada 2025. URL: https://www.sciencedirect.com/science/article/pii/S2666497624000481

[6] C. Choi, J. Lee i in., „Clinical Efficacy of Mobile App-Based, Self-Directed Pulmonary Rehabilitation for Patients With Chronic Obstructive Pulmonary Disease: Systematic Review and Meta-Analysis," *JMIR mHealth and uHealth*, 2024. dostęp 26 listopada 2025. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC10786334/

[7] S. Laborde i in., *The influence of breathing techniques on physical sport performance: a systematic review and meta-analysis*. dostęp 26 listopada 2025. URL: https://www.tandfonline.com/doi/full/10.1080/1750984X.2022.2145573

[8] M. Thomas i in., *Breathing exercises for asthma: a randomised controlled trial*. dostęp 26 listopada 2025. URL: https://thorax.bmj.com/content/64/1/55

[9] G. S. Naik, G. Gaur i G. Pal, *Effect of Modified Slow Breathing Exercise on Perceived Stress and Basal Cardiovascular Parameters*. dostęp 29 listopada 2025. URL: https://journals.lww.com/ijoy/fulltext/2018/11010/effect_of_modified_slow_breathing_exercise_on.6.aspx

[10] H.-C. Chien, Y.-C. Chung, M.-L. Yeh i J.-F. Lee, *Breathing exercise combined with cognitive behavioural intervention improves sleep quality and heart rate variability in major depression*. dostęp 26 listopada 2025. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/jocn.12972

[11] W. Hof, *Wim Hof Method Mobile App*, 2021. dostęp 23 maja 2025. URL: https://www.wimhofmethod.com/wim-hof-method-mobile-app

[12] M. Gomez, *Breathwrk: Breathing Exercises App*. dostęp 23 maja 2025. URL: https://www.breathwrk.com

[13] Y. Maryevich, *STAmina Apnea Trainer App*. dostęp 23 maja 2025. URL: https://getstamina.app

[14] *Breathe App*. dostęp 23 maja 2025. URL: https://havabee.com/#portfolio

[15] Google, *Flutter App Architecture*. dostęp 10 marca 2025. URL: https://docs.flutter.dev/app-architecture/concepts

[16] B. Fallon, *A (blue) nt: Beyond the Symbology of the Colour Blue*, 2014. dostęp 1 grudnia 2025. URL: https://openjournals.test.library.sydney.edu.au/LA/article/view/8539/8623

[17] H. D. Co., *Glacial Indifference font*, 2015. dostęp 20 listopada 2025. URL: https://cargocollective.com/hanken/Glacial-Indifference-Open-Source-Font

[18] *SIL OPEN FONT LICENSE Version 1.1*, 2007. dostęp 20 listopada 2025. URL: https://openfontlicense.org/

[19] Google, *Google Fonts Material Icons*. dostęp 20 listopada 2025. URL: https://fonts.google.com/icons

[20] D. B. Inc., *Lottie Files*. dostęp 20 listopada 2025. URL: https://lottiefiles.com/

[21] CosmoYo, *Free Background Grey Wave Animation*. dostęp 20 listopada 2025. URL: https://lottiefiles.com/free-animation/background-grey-wave-jb3K4GlFay

[22] S. Riznyk, *Free Boat on peaceful water Animation*. dostęp 20 listopada 2025. URL: https://lottiefiles.com/free-animation/boat-on-peaceful-water-lHlU2lslJN

[23] D. B. Inc, *Lottie Simple License (FL 9.13.21)*, 2021. dostęp 20 listopada 2025. URL: https://lottiefiles.com/page/license

[24] M. Martin i R. C. Martin, *Agile principles, patterns, and practices in C*. Pearson Education, 2006.

[25] *ISTQB glossary*. dostęp 7 grudnia 2025. URL: https://glossary.istqb.org/pl_PL

[26] M. Pipkin, *Freediving Apnea Trainer App*. dostęp 23 maja 2025. URL: https://freedivingapp.pro

[27] R. Fired, *Biofeedback and Self-Regulation, Integrating Music in Breathing Training and Relaxation: II. Applications*. New York, 1990. dostęp 26 listopada 2025.

[28] L. Chittaro i R. Sioni, „Evaluating mobile apps for breathing training: The effectiveness of visualization," *Computers in Human Behavior*, 2014. dostęp 26 listopada 2025.

[29] A. S. Foundation, *Apache License, Version 2.0*. dostęp 20 listopada 2025. URL: https://www.apache.org/licenses/LICENSE-2.0

A. INSTRUKCJA UŻYTKOWANIA (Aleksandra Bujny)

Celem tego rozdziału jest przedstawienie zbioru funkcji aplikacji ReSpire. W kolejnych podrozdziałach opisano sposób instalacji aplikacji oraz wszystkie ekrany aplikacji zaprezentowane poprzez zrzuty ekranu. Aplikacja jest opisywana i prezentowana w polskiej wersji językowej.

A.1. Instalowanie, odinstalowanie, uruchamianie i aktualizacja aplikacji

W sekcji opisane zostały kroki niezbędne do zainstalowania, uruchomienia, odinstalowania oraz aktualizacji aplikacji ReSpire na urządzeniu mobilnym z systemem Android.

A.1.1. Instalowanie aplikacji

Instalacja aplikacji wymaga pobrania pakietu instalacyjnego w formacie `.apk` do pamięci urządzenia mobilnego. Proces instalacji inicjowany jest poprzez uruchomienie pobranego pliku. Należy postępować zgodnie z komunikatami wyświetlanymi przez instalator systemu Android. Na ekranie telefonu może się pojawić okno (przedstawione na rysunku A.1) z zapytaniem o zezwolenie na instalację aplikacji spoza Sklepu Google Play, jeśli to ustawienie jest wyłączone. Należy kliknąć przycisk *Ustawienia*, a następnie zezwolić na instalację przesuwając przycisk w prawo. Można tego dokonać również poprzez wejście w odpowiednią zakładkę bezpośrednio z poziomu aplikacji ustawień (*Aplikacje* -> *Specjalny dostęp do aplikacji* -> *Instalowanie nieznanych aplikacji* -> *Files by Google lub inny eksplorator plików zainstalowany na urządzeniu*). Strona ustawień z włączoną opcją przedstawiona jest na rysunku A.2.

Rysunek A.1. Okno dialogowe informujące o konieczności zezwolenia na instalację

Rysunek A.2. Strona ustawień z włączoną opcją instalacji aplikacji spoza Sklepu Google Play

Po pojawieniu się okna z zapytaniem o instalację widocznego na rysunku A.3 należy kliknąć przycisk *Zainstaluj* i poczekać na zakończenie instalacji. Okno informujące o postępie zostało przestawione na rysunku A.4. Do anulowania akcji konieczne jest naciśnięcie przycisku *Anuluj* w oknie z zapytaniem o instalację.

Rysunek A.3. Okno z zapytaniem o instalację aplikacji

Rysunek A.4. Okno informujące o postępie w instalacji aplikacji

A.1.2. Uruchamianie aplikacji

Jeśli instalacja przebiegła pomyślnie, aplikacja powinna następnie pojawić się na liście aplikacji na urządzeniu i być gotowa do uruchomienia. Podczas startu widoczny jest ekran przedstawiony na rysunku A.5.

Rysunek A.5. Ekran uruchamiania aplikacji ReSpire

A.1.3. Odinstalowanie aplikacji

Aplikację można odinstalować w standardowy sposób - należy nacisnąć i przytrzymać przez kilka sekund ikonę aplikacji, a następnie przeciągnąć ją na ikonę kosza na śmieci lub przycisk *Odinstaluj*, który pojawi się na ekranie. Akcja musi zostać potwierdzona w oknie dialogowym, które się pokaże.

A.1.4. Aktualizacja aplikacji

W przypadku, gdy aplikacja jest już zainstalowana na urządzeniu, można ją zaktualizować poprzez pobranie pliku w formacie *.apk* z nową wersją aplikacji i pojedyczne kliknięcie w plik, a nastepnie przycisk *Aktualizuj*. Okno z zapytaniem o aktualizację przedstawione jest na rysunku A.6. Po zakończeniu aktualizacji aplikacja będzie gotowa do uruchomienia z nową wersją.

Rysunek A.6. Okno z zapytaniem o aktualizację aplikacji

A.2. Strona główna

Ekran główny aplikacji ReSpire pokazuje się po jej uruchomieniu i stanowi jej centralną część. Składa się z dwóch komponentów: u góry znajduje się pasek z opcjami i logo, natomiast resztę ekranu zajmują kafelki z dostępnymi treningami oddechowymi. Na dole ekranu widoczna jest również animacja fal, które delikatnie poruszają się w tle, stanowiąc element dekoracyjny. Interfejs strony głównej został przedstawiony na rysunku A.7.

Rysunek A.7. Strona główna aplikacji ReSpire widoczna po jej uruchomieniu

A.2.1. Lista treningów

Na stronie widoczna jest lista wszytskich treningów dostępnych dla użytkownika, przedstawiona w postaci klikalnych kafelków z nazwą treningu i ozdobnym detalem symbolizującym podmuch powietrza. Naciśnięcie kafelka przenosi na stronę szczegółów treningu opisaną dokładnie w sekcji A.3. Zawiera ona przegląd, możliwość edycji, usunięcia, eksportu czy rozpoczęcia treningu. Domyślnie, po zainstalowaniu aplikacji, dostępne są trzy predefiniowane treningi - *Box Breathing*, *4-7-8* oraz *Coherent Method*. W przypadku, gdy użytkownik chce dodać swój trening, może to zrobić klikając ikonę plusa w białym okręgu z jasnoniebieskim cieniem. Zostanie wtedy przeniesiony na stronę edycji treningu, która opisana jest w sekcji A.4. Po dodaniu indywidualnie skonfigurowanego treningu wyświetli się on na końcu listy, tak jak przedstawiono na rysunku A.8.

Rysunek A.8. Lista predefiniowanych treningów oraz dodany przez użytkownika trening *Mój Trening* umieszczony na końcu listy

A.2.2. Pasek z opcjami

Na środku paska z opcjami znajduje się logo aplikacji, a w prawej części - dwie ikony: koło zębate przenoszące użytkownika na stronę ustawień (omówioną w rozdziale A.6) oraz symbol ze strzałką w dół umożliwiający import zapisanych treningów. Po kliknikniętciu w ikonę importu następuje przeniesienie do eksploratora plików na urządzeniu z domyślnie otworzonym folderem *Pobrane*, widocznym na rysunku A.9, skąd można wybrać plik z zapisanym wcześniej treningiem lub treningami. Folder można zmienić nawigując w ekploratorze. Po imporcie zakończonym sukcesem trening pojawia się na końcu listy treningów opisanej w podsekcji A.2.1. Użytkownik zostaje także powiadomiony o pomyślnym imporcie poprzez wyświetlenie w aplikacji komunikatu widocznego na rysunku A.10.

Rysunek A.9. Eksplorator plików otwarty w folderze *Pobrane* na urządzeniu podczas wyboru pliku do importu treningów lub treningów oddechowych

Rysunek A.10. Komunikat informujący o pomyślnym imporcie treningu, na liście widoczny jest zaimportowany trening o nazwie *Mój Trening*

Po lewej stronie paska umieszczona jest ikona symbolizująca tryb zaznaczania opisany w podrozdziale A.2.3, który staje się aktywny po kliknięciu. Tryb zmienia wygląd paska w sposób przedstawiony na rysunku A.11.

Rysunek A.11. Porównanie paska z opcjami w trybie standardowym (po lewej) oraz z aktywowanym trybem zaznaczania (po prawej)

A.2.3. Tryb zaznaczania

Tryb zaznaczania służy do zbiorowego eksportu zaznaczonych trenigów. Użytkownik może wybrać dowolne z nich z listy poprzez kliknięcie kafelków z ich nazwami - zaznaczone elementy zostają oznaczone ciemniejszą, pogrubioną linią. Jeśli użytkownik chce wybrać wszystkie treningi, może to zrobić w szybki i wygodny sposób - klikając przycisk *Zaznacz wszystkie*, który zaznaczy automatycznie wszystkie dostępne opcje z listy. Po wybraniu co najmniej jednej z nich pojawia się informacja o liczbie wybranych treningów. Przykładowe zaznaczenie dwóch z nich zostało ukazane na rysunku A.12. Treningi można także odznaczyć, jeśli użytkownik zmieni zdanie. Jeśli wybór jest satysfakcjonujący dla użytkownika należy nacisnąć ikonę z prawej strony paska. Po wybraniu miejsca eksportu w eksploratorze plików, który się otworzy, opcjonalnie zmianie nazwy pliku i kliknięciu przycisku *Zapisz* plik w formacie *JSON* zostanie zapisany na urządzeniu. W przypadku eksportu kilku treningów, zostaną one zapisane w jednym pliku. Opcja eksportu dostępna jest także z poziomu strony szczegółów treningu A.5, jednak w przeciwieństwie do strony głównej, nie pozwala na eksport grupowy, a jedynie indywidualny.

Rysunek A.12. Strona główna aplikacji w trybie zaznaczania z wybranymi dwoma treningami

W aplikacji wyświetla się także odpowiedni komunikat mówiący o tym, czy zapis się powiódł. Na rysunku A.13 pokazany jest komunikat o pomyślnym eksporcie, natomiast na rysunku A.14 - komunikat o próbie zapisania treningów bez wybrania żadnego, co skutkuje brakiem zapisu.

Rysunek A.13. Komunikat o pomyślnym eksporcie zestawu zaznaczonych treningów

Rysunek A.14. Komunikat informujący o nieudanym zapisie z powodu braku zaznaczonych treningów

Eksport zachowuje nie tylko strukturę treningu, ale również ustawienia dźwiękowe i inne, skonfigurowane przez użytkownika. Istotny jest jednak fakt, że eksportowane są jedynie dźwięki domyślnie dostępne w aplikacji. Te dodane przez użytkownika zastępowane są natomiast brakiem dźwięku. Eksportowane treningi można przesyłać na inne urządzenia, a następnie je tam odtwarzać. Przykładowe pliki z zapisanymi treningami pokazane są na rysunku A.15. Domyślnie nadawana nazwa pliku występuje w formacie *respire_trainings_ZNACZNIK-CZASOWY.json*.

Rysunek A.15. Zapisane pliki z treningami w eksploratorze plików na urządzeniu, u góry - pakiet treningów z nazwą nadaną przez użytkownika, na dole - nazwa nadawana domyślnie przez aplikację przy eksporcie

A.3. Strona szczegółów treningu

Na stronę można przejść, klikając w kafelek z wybranym treningiem na stronie głównej. Interfejs strony został przedstawiony na rysunku A.17. Wzbogaca go animacja łódki delikatnie kołyszacej się na fali.

Na pasku u góry znajduje się tytuł treningu oraz strzałka umożliwiająca powrót na stronę główną. Niżej znajdują się klikalne ikony pozwalające na akcje związane z treningiem. Z lewej strony umieszczona została opcja eksportu, która działa analogicznie do tej ma stronie głównej, z tą różnicą, że eksportujemy tylko jeden trening. Po prawej natomiast znajduje się opcja usunięcia treningu (oznaczona symbolem kosza na śmieci), a także możliość edycji (symbol ołówka). Przy akcji usuwania wyświetla się okno z prośbą o jej potwierdzenie, co zapobiega przypadkowemu usunięciu treningu. Okno to zostało pokazane na rysunku A.16. Po wybraniu opcji edycji aplikacja przenosi użytkownika do opisywanej w kolejnej sekcji strony edytora A.4.

Rysunek A.16. Okno dialogowe z prośbą o potwierdzenie chęci usunięcia treningu

Niżej znajduje się sekcja opisu treningu. Można go edytować lub dodać na stronie edycji treningu. Predefiniowane treningi zawierają opisy, jednak nie jest to element wymagany.

Następnym elementem jest *Przegląd treningu*, który po naciśnięciu strzałki po prawej stronie rozwija się, pokazując wszystkie etapy treningu wraz z ich nazwami, fazami oddechowymi i długościami ich trwania, a także liczbą powtórzeń oraz przyrostem. Dzięki takiemu podsumowaniu użytkownik może łatwo przejrzeć strukturę treningu.

Na stronie znajduje się również przycisk *Rozpocznij trening*, który przenosi użytkownika na stronę treningu opisaną w sekcji A.5, tym samym rozpoczynając ćwiczenie oddechowe odpowiadające szczegółom widocznym na tej stronie.

Rysunek A.17. Strona szczegółów treningu oddechowego ze zwiniętym przeglądem treningu po lewej oraz rozwiniętym po prawej

A.4. Strona edycji treningu

Edytor składa się z trzech głównych zakładek: *Trening*, *Dźwięki* oraz *Inne*, tak jak na widocznym rysunku A.18 (oznaczenie bordową ramką). Można pomiędzy nimi przechodzić poprzez kliknięcie na nazwę zakładki, która ma zostać otwarta. Podstrona, na której przebywa użytkownik w danym momencie wyróżnia się spośród pozostałych - jej nazwa posiada białe tło i niebieski kolor czcionki, podczas gdy zakładki nieaktywne mają odwrotną kolorystykę, czyli białą czcionkę na niebiekim tle. Strona edytora pozwala na szczegółowe definiowanie struktury oraz oprawy muzycznej treningu, umożliwiając użytkownikowi precyzyjne dostosowanie treningu do indywidualnych potrzeb.

Na pasku znajdującym się u góry strony edycji treningu umieszczone są następujące elementy: nazwa edytowanego treningu, strzałka umożliwiająca powrót do strony ze szczegółami treningu (strona opisana została w sekcji A.3) oraz ikona ołówka. Po kliknięciu ikony na ekranie aplikacji wyświetla się okno edycji nazwy treningu ukazane na rysunku A.19. Po wprowadzeniu wybranej nazwy lub edycji istniejącej, należy ją następnie zatwierdzić przyciskiem *Zapisz* w celu zapisu. Można także wyjść bez zapisu po kliknięciu *Anuluj*. Długość nazwy treningu ograniczona jest do piętnastu znaków. Po osiągnięciu tego limitu dalsze symbole nie będą pojawiały się w polu edycji. Na dole, po jego prawej stronie, znajduje się licznik, tak aby użytkownik wiedział, ile znaków pozostało mu jeszcze do wykorzystania. Przy tworzeniu nowego treningu nazwa jest nadawana automatycznie - w polskiej wersji językowej aplikacji jest to *Mój Trening*, natomiast w angielskiej *My Training*. Pasek nie zmienia się w zależności od wybranej zakładki i pozostaje widoczny podczas edycji treningu.

Rysunek A.18. Widok edytora treningu z trzema zakładkami oznaczonymi bordową ramką

Rysunek A.19. Okno edycji nazwy treningu z ograniczeniem jej długości do piętnastu znaków

A.4.1. Zakładka Trening

Pierwsza z zakładek edytora - *Trening* - służy do tworzenia struktury nowego treningu lub edycji już istniejącego. Przyciśnięcie przycisku *Dodaj etap treningu* umożliwia użytkownikowi utworzenie i dołączenie nowego etapu do treningu, co skutkuje pojawieniem się na ekranie kafelka reprezentującego uwtorzony etap pod istniejącymi etapami. Można zmienić kolejność kafelków poprzez naciśnięcie i przytrzymanie symbolu dwóch równoległych kresek umieszczonych po lewej stronie, a następnie przesunięcie palcem po ekranie w górę lub dół, do momentu gdy kafelek znajdzie się w pożądanym miejscu. Symbol przesuwania oznaczony został bordowym okręgiem na rysunku A.20, a ruch palca po ekranie symbolicznie przedstawiono przy pomocy strzałki w tym samym kolorze. Każdy kafelek zawiera nazwę etapu, która jest dodawana domyślnie przy jego tworzeniu. Ma ona postać *Etap treningu X*, gdzie *X* to kolejne numery etapów, zaczynając od jedynki i może zostać edytowana poprzez naciśnięcie na pole znajdującą się w zaokrąglonej ramce, a następnie wprowadzenie zmian przy pomocy klawiatury, która pokaże się na ekranie. Długość nazwy etapu, podobnie jak w przypadku treningu, jest ograniczona. Limit stanowi dwadzieścia pięć znaków, a po jego osiągnięciu użytkownik nie będzie miał możliwości wpisania ich więcej. Licznik wyświetlany jest pod polem z nazwą, co pozwala kontrolować liczbę wykorzystanych symboli.

Usunięcie etapu jest możliwe poprzez kliknięcie ikony kosza na śmieci znajdującej się po prawej stronie pola z nazwą treningu, a następnie potwierdzenie wykonania akcji klikając przycisk *Usuń* w okienku, które się pojawi na środku ekranu. Zostało ono przedstawione na rysunku A.21. Usuwanie można także anulować klikając przycisk *Anuluj*.

Rysunek A.20. Edytor treningu podczas zmieniania kolejności etapów treningu

Rysunek A.21. Okno pozwalające na potwierdzenie lub anulowanie akcji usunięcia etapu treningu

Poniżej znajdują się pola dedykowane liczbie powtórzeń danego etapu oraz przyrostowi wyrażonemu w sekundach. Liczba powtórzeń definiuje, ile razy po sobie będzie odtwarzany dany etap, natomiast przyrost - o ile sekund będzie dłuższa każda faza oddechowa w kolejnych powtórzeniach. Wartości można edytować poprzez naciśnięcie ikon plusa / minusa (odpowiednio zwiększenie / zmniejszenie wartości) lub klikając w pole z aktualną liczbą, co spowoduje pokazanie klawiatury na ekranie i umożliwi użytkownikowi wpisanie wybranej wartości. W przypadku używania przycisków plusa lub minusa dla powtórzeń wartość będzie się zmieniać o jedno powtórzenie, natomiast dla przyrostu - o jedną dziesiątą sekundy w przedziale od zera do jednej sekundy lub o jedną sekundę dla wartości większych niż jedna sekunda. Domyślnymi ustawieniami jest jedno powtórzenie (cykl) etapu oraz brak przyrostu (ustawiony na zero sekund).

Za pomocą przycisku *Dodaj fazę oddechu* użytkownik może dodać fazy oddechu do etapu. Są one wyświetlane w postaci listy kafelków. Każda faza oddechu zawiera parametry takie jak czas trwania oraz typ. Czas trwania wyrażany jest w sekundach i może być zmodyfikowany poprzez naciśnięcie symbolu plus / minus (odpowiednio zwiększenie / zmniejszenie) lub kliknięcie w pole z aktualną wartością i wpisanie żadanej wartości z klawiatury, która się pokaże na ekranie. Wartości zmieniają się o jedną dziesiątą sekundy w zakresie od zera do jednej sekundy, natomiast później o pół sekundy. W przypadku wpisywania z klawiatury, jeśli podana wartość ma większą dokładność niż obsługiwana, wartość jest automatycznie zaokrąglana. Przykładowo, jeśli użytkownik wpisze wartość cztery sekundy i osiem dziesiątych, zostanie ona zaokrąglona do wartości pięciu sekund. Typ można ustawić natomiast poprzez kliknięcie w pole pod napisem *Typ* - pojawia się wówczas rozwijana lista, z której użytkownik może wybrać żądany typ po kliknięciu w wybraną opcję. Dostępne typy faz oddechu - *wdech*, *wydech*, *wstrzymanie* oraz *retencja* przedstawione zostały na rysunku A.22. Fazę można usunąć poprzez kliknięcie ikony kosza na śmieci

umiejszonego po prawej stronie kafelka, a następnie potwierdzenia akcji za pomocą przycisku *Usuń* w oknie, które się pojawi. Zostało ono przedstawione na rysunku A.23.

Rysunek A.22. Lista z dostępnymi typami faz oddechu - *wdech*, *wydech*, *wstrzymanie* i *retencja*

Rysunek A.23. Okno usuwania fazy oddechu z etapu treningu

Można także zmienić kolejność faz oddechowych w danym etapie poprzez naciśnięcie i przytrzymanie symbolu dwóch równoległych linii umieszczonych po lewej stronie kafelka, a następnie przesunięcie palcem w górę lub dół ekranu, tak aby kafelek znalazł się w pożądanym miejscu, analogicznie jak w przypadku etapów. Pozwala to na łatwe i szybkie zarządzanie strukturą treningu tworzononego przez użytkownika. Zmiana kolejności faz oddechu została przedstawiona na rysunku A.24 - symbol przesuwania oznaczono bordowym okręgiem, a ruch palca po ekranie w górę - strzałką w tym samym kolorze.

Dostępna jest także opcja zwinięcia kafelka za pomocą strzałki w górę umieszczonej z jego prawej strony, obok ikony kosza na śmieci - nie jest wyświetlana wtedy lista faz oddechu dla danego treningu. Kafelek można z powrotem rozwinąć klikając strzałkę w dół znajdującą się w tym samym miejscu, co uprzednio strzałka w górę. Stronę edytora ze zwiniętymi kafelkami przedstawiono na rysunku A.25.

Wszystkie zmiany wprowadzone do edytora zostają zapisane automatycznie po jego opuszczeniu - kliknięciu strzałki w lewym górnym rogu paska strony. W przypadku próby zapisania kompletnie pustego treningu lub z pustymi etapami użytkownik otrzyma odpowiedni komunikat. Okna dialogowe z informacją dla obu z opisanych sytuacji widoczne są na rysunkach A.26. Użytkownik może wybrać, czy chce powrócić do edycji - przycisk *Wróć do edycji*, czy wyjść z edytora bez zapisu pustych etapów bądź całego treningu - przycisk *Wyjdź mimo to*.

Rysunek A.24. Zmiana kolejności faz oddechu w etapie treningu poprzez przeciąganie kafelka w górę

Rysunek A.25. Widok edytora treningu ze zwiniętymi kafelkami etapów treningu - bez wyświetlonych faz oddechu

Rysunek A.26. Okna dialogowe pojawiające się przy próbie opuszczenia edytora treningiem z brakujacymi fazami w etapach (po lewej) lub całkowicie pustym treningiem (po prawej)

A.4.2. Zakładka Dźwięki

Zakładka *Dźwięki* zawiera rozbudowany zbiór ustawień związanych z oprawą dźwiękową treningu, która umożliwia wykonanie treningu z zamkniętymi oczami i kontroli kolejnych elementów treningu dzięki zmysłowi słuchu, a nie wzroku (obserwowanie animacji). Zakładka przedstawiona została na rysunku A.27. Składa się ona z trzech sekcji: *Dźwięki treningu*, *Muzyka treningu* oraz *Dźwięki binauralne*. Sekcje *Dźwięki treningu* oraz *Muzyka treningu* podzielone są na podsekcje, które są oznaczone niebieską, zaokrągloną ramką. Każda z tych podsekcji odpowiada za inny element oprawy dźwiękowej treningu. Niektóre, bardziej rozbudowane podsekcje, składają się natomiast z opcji.

Przy tworzeniu nowego treningu domyślnie ustawione są ściśle określone dźwięki i muzyka wybrane przez twórców aplikacji przy jej tworzeniu, natomiast można je zmienić lub wyłączyć postępując zgodnie z instrukcjami znajdującymi się w dalszym opisie. Użytkownik może wybierać spośród dźwięków dostępnych w aplikacji lub własnych, które samodzielnie doda. W niektórych podsekcjach dostępny jest także lektor.

Rysunek A.27. Zakładka *Dźwięki* edytora treningu umożliwiająca konfigurację oprawy dźwiękowej treningu

Istotnym faktem, który jest kluczowy dla zrozumienia sposobu działania aplikacji, jest istnienie trzech zestawów dźwięków. Dźwięki zostały podzielone na krótkie - dla sekcji *Dźwięki treningu* oraz długie (muzyka) - dla sekcji *Muzyka treningu*. Wyjątkiem jest podsekcja *Odliczanie* należąca do sekcji *Dźwięki treningu*, która posiada trzeci, osobny zestaw. Dodanie własnego pliku nagrania dźwiękowego do jednego z tych zestawów spowoduje, że będzie on wyświetlany na liście w każdej podsekcji, która korzysta z danego zestawu dźwięków. Ważne, aby pamiętać, że w przypadku dźwięków krótkich najlepiej dodawać maksymalnie kilkusekundowe nagranie dźwiękowe, w celu uzyskania jak najlepszych wrażeń z korzystania z aplikacji - dłuższe dźwięki zostaną ucięte.

W każdej podsekcji, nazwa obecnie ustawionego nagrania dźwiękowego (lub informacja o jego braku czy wyborze lektora) wyświetlana jest po lewej stronie ikony nuty. Po kliknięciu w napis lub ikonę nuty otwiera się okno wyboru dźwięku lub muzyki zawierające listę dostępnych opcji. Szarym cieniem zaznaczony jest obecny wybór, tak jak przedstawiono to na rysunku A.28. Jeśli użytkownik chce odsłuchać, jak brzmi dane nagranie dźwiękowe może to zrobić po kliknięciu w zieloną, trójkątną ikonę odtwarzania. Przycisk *Anuluj* przerwa akcję wyboru dźwięku / muzyki, natomiast przycisk *Dodaj własny dźwięk* otwiera okno z plikami na telefonie użytkownika (domyślnie w folderze *Pobrane*, jednak można przejść także do pozostałych folderów) i pozwala na wybór pliku nagrania dźwiękowego w formacie *.mp3*. Dodany przez użytkownika dźwięk lub muzyka pojawi się następnie pod listą domyślnych dźwięków, pod napisem *Dźwięki użytkownika* - obrazuje to rysunek A.29.

Rysunek A.28. Przykładowa lista dźwięków z obecnym wyborem oznaczonym szarym cieniem

Rysunek A.29. Muzyka użytkownika dodana do listy dostępnych nagrań długich

W celu usunięcia dodanego przez użytkownika pliku należy nacisnąć czerwoną ikonę kosza na śmieci, a następnie potwierdzić tą akcję przyciskiem *Usuń* w oknie, które się pojawi - przedstawiono je na rysunku A.30. Skutkiem będzie usunięcie dźwięku z listy dźwięków użytkownika. Aby wyłączyć dźwięk lub muzykę w danej podsekcji należy wybrać opcję *Brak* z listy dostępnych dźwięków lub muzyki. W przypadku podsekcji *Następna faza oddechu* lub *Muzyka treningu*, które są bardziej rozbudowane, należy wybrać również opcję *Brak*, natomiast znajduje się ona na innej liście - tekstowej, a nie z dźwiękami - którą można otworzyć poprzez kliknięcie niebieskiej strzałki w dół w miejscu, gdzie w pozostałych podsekcjach znajduje się ikona nuty.

Sekcja *Dźwięki treningu* korzysta z dwóch różnych zestawów dźwięków krótkich i składa się z czterech podsekcji. Pierwsza z nich to *Odliczanie*, zawierająca dźwięk odliczania czasu trwania faz oddechowych. Posiada ona własny zestaw dźwięków, składający się on z dwóch nagrań oraz lektora. Po dodaniu własnego dźwięku przez użytkownika do tego zestawu nie pojawi się on na listach w pozostałych podsekcjach, ponieważ korzystają one z osobnego zestawu dwunastu dźwięków.

Kolejną, bardziej rozbudowaną podsekcją niż opisywane wcześniej, jest *Następna faza oddechu*. Zawiera cztery opcje do wyboru. Są one osiągalne z listy, która rozwija się po kliknięciu niebieskiej strzałki. Po dokonaniu wyboru, informacja o obecnie obowiązującej opcji wyświetla się po dwukropku za nazwą podsekcji. Lista została przedstawiona na rysunku A.31.

Rysunek A.30. Okno potwierdzenia usunięcia dźwięku użytkownika z listy dostępnych nagrań

Rysunek A.31. Opcje dostępne do wyboru w podsekcji *Następna faza oddechu*

Opisywana podsekcja edytora dźwięków definiuje dźwięk odtwarzany po zmianie fazy oddechowej na inną. Można wyłączyć ten dźwięk klikając opcję *Brak*. Kolejną opcją jest wybór globalnego dźwięku - tego samego dla każdej fazy (wspomniany wcześniej zestaw dwunastu krótkich dźwięków wzbogacony jest tu o lektora, który czyta nazwę fazy oddechowej). Następymi możliwościami są osobne dźwięki dla każdej z czterech faz oddechu - dla wszystkich etapów takie same lub różne. Pozwala to użytkownikowi na stworzenie skojarzeń faza oddechowa - dźwięk. Dzięki temu może wykonywać trening z zamkniętymi oczami i bez głosu lektora, wiedząc jaka jest kolejna faza, rozpoznając ją po nagraniu. Opisane opcje globalnego dźwięku oraz osobnych dźwięków dla faz oddechowych przedstawione zostały na rysunku A.32.

Rysunek A.32. Interfejs podsekcji *Następna faza oddechu* z wybraną opcją ogólnego dźwięku zmiany fazy (po lewej) oraz z wybranymi osobnymi dźwiękami dla każdej z faz oddechu - ogólnymi dla wszystkich etapów (na środku) i dla konkretnych etapów (po prawej)

Kolejną podsekcją jest *Zmiana etapu*, w której użytkownik może wybrać dźwięk sygnalizujący zmianę etapu treningu na kolejny.

Ostatnia podsekcja to natomiast *Zmiana cyklu*, która definiuje dźwięk otwarzany po zakończeniu cyklu (czyli powtórzenia) w obrębie etapu. Przy ostatnim cyklu etapu dźwięk ten nie jest odtwarzany - zamiast niego następuje przejście do kolejnego etapu (i odtworzenie dźwięku zmiany etapu, jeśli został on zdefiniowany) lub zakończenie treningu, jeśli był to ostatni etap.

W sekcji *Muzyka treningu* również znajdują się trzy podsekcje. Każda z nich korzysta z zestawu szesnastu dźwięków długich (muzyki), co oznacza, że po dodaniu przez użytkownika własnej muzyki wyświetli się na listach we wszystkich podsekcjach. Na liście oprócz nazwy nagrania wyświetlany jest także jego czas trwania.

Pierwsza z podsekcji opisywanej sekcji - *Muzyka w tle* - jest bardziej rozbudowana, analogicznie do podsekcji *Następna faza oddechu*. Po rozwinięciu listy za pomocą kliknięcia w niebieską strzałkę, pozwala ustawić muzykę w tle treningu na różnym poziomie szczegółowości lub ją wyłączyć (opcja *Brak*). Dostępne poziomy szczegółwości przedstawione są na rysunku A.33. Zaczynając od najmniej szczegółowego, poziomy te to: muzyka na cały trening (rysunek A.34), muzyka zdefiniowna dla kolejnych etapów treningu lub muzyka dla konkretnych faz oddechowych - taka sama dla różnych etapów lub różna dla poszczególnych etapów (rysunek A.35). Dźwięki w tle są zapętlane w trakcie trwania treningu, etapu lub fazy oddechowej, w zależności od wybranej opcji. Można wybrać więcej niż jeden plik dźwiękowy poprzez stworzenie *playlisty* (pol. listy odtwarzania) dla etapów trenigu lub muzyki w tle dla całego treningu, co zostało opisane w dalszej części instrukcji A.4.2.

Rysunek A.33. Opcje dostępne do wyboru w podsekcji *Muzyka w tle*

Rysunek A.34. Opcja muzyki na cały trening wybrana w podsekcji *Muzyka w tle*

Rysunek A.35. Pozostałe opcje dostępne do wyboru w podsekcji *Muzyka w tle*. Od lewej: muzyka dla etapów treningu, muzyka dla faz oddechowych - ta sama dla różnych etapów oraz muzyka dla faz oddechowych - różna dla poszczególnych etapów

W celu uwtorzenia kolejki muzyki należy kliknąć przycisk *Dodaj muzykę*, wybrać nagranie z listy, a następnie powtórzyć te kroki żądaną ilość razy. W celu usunięcia muzyki z kolejki należy kliknąć ikonę kosza na śmieci. Symbol dwóch równoległych kresek służy natomiast, podobnie jak w przypadku kafelków z etapami czy fazami, do zmieniania kolejności. Przytrzymując symbol, a następnie przesuwając palcem w górę lub dół ekranu użytkownik może ułożyć nagrania w dowolnej kolejności. Jeśli użytkownik nie doda żadnego nagrania do kolejki otrzyma o tym stosowny komunikat w miejscu, gdzie wyświetlany jest aktualny stan kolejki. W trakcie treningu muzyka będzie odtwarzana według kolejności określonej w kolejce. Jeśli zawiera ona tylko jeden plik z muzyką, będzie on odtwarzany w zapętleniu. W przypadku muzyki w tle dla etapów treningu lub faz oddechowych w celu zwinięcia lub rozwinięcia widoku kolejki należy kliknąć w czarną strzałkę skierowaną odpowiednio w górę lub w dół. Zmienianie kolejności nagrań, komunikat o pustej kolejce oraz zwinięty widok kolejki przedstawiono na rysunku A.36.

Rysunek A.36. Różne widoki kolejki muzyki: zmienianie kolejności nagrań (po lewej), komunikat o pustej kolejce (w środku) oraz zwinięty widok kolejki (po prawej)

Podsekcje *Przygotowanie* oraz *Zakończenie* definiują natomiast muzykę (lub jej brak) sygnalizujac odpowiednio rozpoczęcie treningu oraz jego zakończenie.

Sekcja *Dźwięki binauralne* domyślnie jest wyłączona. Można ją aktywować za pomocą kliknięcia w przełącznik znajdujący się po prawej stronie napisu *Włącz dźwięki binauralne*. Istotny jest fakt, że w przypadku włączenia tej opcji automatycznie nastąpi wyłączenie opcji muzyki w tle - zostanie ona zablokowana w interfejsie i oznaczona jaśniejszym kolorem, jak na rysunku A.37. Po aktywacji, sekcja się rozwija i pokazują się dwa suwaki, za pomocą których użytkownik może ustawić częstotliwości dudnienia synchronicznego (czyli dźwięku binauralnego) dla prawego oraz lewego ucha. Sumaryczna częstotliwość uderzenia wyświetlana jest na dole sekcji. Dudnienie synchroniczne odtwarzane jest w tle treningu, zamiast muzyki w tle - stąd opisywana blokada tej funkcjonalności.

Rysunek A.37. Aktywowane dudnienia synchroniczne wraz z zablokowaną opcją muzyki w tle

A.4.3. Zakładka Inne

W zakładce *Inne*, której interfejs został ukazany na rysunku A.38, znajdują się pozostałe ustawienia. Możliwe jest dodanie nowego lub modyfikacja istniejącego opisu treningu. Po kliknięciu wewnątrz ramki pod napisem *Opis treningu* na ekranie pokazuje się klawiatura oraz kursor, a ramka zmienia kolor na niebieski - aktywowany zostaje tryb edycji opisu. Po schowaniu klawiatury i przejsciu do innej zakładki lub wyjściu z edytora zmiany zostają zapisane. Pole opisu rozszerza się w miarę potrzeby, gdy wpisywany tekst jest długi. Edycja opisu została przedstawiona na rysunku A.39.

Dostępna jest także opcja ustawiania długości trwania przygotowania. Należy to zrobić używając przycisków plusa lub minusa (odpowiednio zwiększenie lub zmniejszenie wartości) lub klikając w okno z aktualnie ustawioną liczbą, a następnie wpisując wybraną liczbę z klawiatury, która się pokaże. Domyslnie czas przygotowania do treningu ustawiony jest na trzy sekundy.

Poniżej znajduje się sekcja, która pozwala ustawić długość zakończenia treningu, domyślnie trwającego pięć sekund. Jej obsługa jest analogiczna do sekcji z długością czasu przygotowania. W przypadku wyboru czasu przygotowania lub zakończenia treningu równego zeru sekund oraz równoczesnym ustawieniu muzyki dla tych elementów treningu, będą one trwały tyle, ile trwa wybrana muzyka. Użytkownik zostaje powiadomiony o takim zachowaniu aplikacji poprzez wyświetlony żółty trójkąt ostrzegawczy, po kliknięciu którego pokazuje się stosowny komunikat pokazany na rysunku A.40.

Rysunek A.38. Zakładka *Inne* w edytorze treningu oddechowego, ramka opisu treningu jest szara, ponieważ nie jest w trybie edycji

Rysunek A.39. Proces edycji opisu treningu, ramka opisu zmieniona na kolor na niebieski, aby zasygnalizować tryb edycji

Rysunek A.40. Ostrzeżenie informujące o zachowaniu aplikacji przy ustawieniu czasu przygotowania lub zakończenia treningu na zero sekund wraz z jednoczesnym ustawieniem muzyki

A.5. Strona treningu oddechowego

Podczas wczytywania strony treningu oddechowego może się pojawić podstrona z komunikatem informujący o ładowaniu dźwięków przedstawiona na rysunku A.41. Sygnalizuje ona, że trening jest przygotowywany do poprawnego startu.

Rysunek A.41. Podstrona informująca o ładowaniu dźwięków przed rozpoczęciem treningu

U góry strony treningu oddechowego, której interfejs ilustruje rysunek A.42, widoczny jest tytuł odtworzonego treningu, a także strzałka umożliwiająca powrót do strony szczegółów treningu oraz po prawej stronie przycisk pauzy, jeśli użytkownik chce wstrzymać trening. Następuje wówczas zastopowanie dźwięków i muzyki w tle, a także ruchu kafelków. W celu wznowienia treningu należy nacisnąć ikonę odtwarzania, która pojawia się w miejsce ikony pauzy lub napis *Wznów* umieszczony w środku opisanej poniżej animacji. Zatrzymany trening został ukazany na rysunku A.43. Podczas trwania treningu ekran urządzenia nie wygasza się, aby użytkownik mógł w pełni skupić się na ćwiczeniu oddechowym i nie musiał odblokowywać ekranu po jego wygaszeniu w celu kontynuowania treningu.

Na stronie treningu wyświetlana jest nazwa aktualnego etapu treningu, a poniżej niej - informacja na którym etapie z ilu łącznie jest użytkownik. Wyświetlany jest także licznik wszystkich faz, co pozwala mniej więcej zorientować się, jak dużo faz oddechowych zostało do końca treningu. Na ekranie znajduje się także karuzela z kafelkami, które przesuwają się wraz z przbiegiem treningu. Kafelki zawierają nazwę kroku (fazy oddechowej, rozpoczęcia lub zakończenia treningu) oraz czas jego trwania. Centralny, największy kafelek symbolizuje obecnie trwający krok, kafelek na lewo od niego - poprzedni krok, a kafelek na prawo od centralnego - kolejny krok. Dzięki temu użytkownik może śledzić przebieg treningu oraz odpowiednio wcześniej przygotować się na nadchodzący krok. Poniżej karuzeli dostępna jest także informacja o tym, który cykl powtórzeń etapu jest aktualnie wykonywany.

Rysunek A.42. Strona treningu oddechowego podczas aktywnego treningu

Rysunek A.43. Zatrzymany trening oddechowy - na środku koła widoczny jest napis *Wznów*

Głównym elementem strony jest animacja koła, które zmienia się zgodnie z przebiegiem treningu. Podczas wdechu koło powiększa się, a podczas wydechu - pomniejsza. W czasie trwania fazy regneracji lub wstrzymania koło jest natomiast statyczne. Obrazuje to użytkownikowi, jaką fazę oddechową powinien obecnie wykonywać. Na środku animacji znajduje się także licznik czasu przeznaczonego na dany krok.

Jeśli użytkownik kliknie strzałkę powrotu do strony szczegółów treningu, zostanie zapytany o potwierdzenie swojej akcji, w celu zapobiegnięcia przypadkowemu opszczeniu treningu. Okno dialogowe wyświetlane użytkownikowi przedstawia rysunek A.44. Po zakończeniu treningu oraz upływie czasu zakończenia użytkownik zostaje przeniesiony automatycznie z powrotem na stronę szczegółów treningu.

Rysunek A.44. Okno dialogowe z prośbą o potwierdzenie chęci opuszczenia treningu wyświetlane podczas próby opuszczenia treningu

A.6. Strona ustawień

Strona ustawień przedstawiona na rysunku A.45 zawiera dwie sekcje: wybór języka aplikacji oraz notatkę o ReSpire informującą, jaki jest cel aplikacji. W celu zmiany języka aplikacji należy nacisnąć strzałkę obok informacji o obecnie ustawionym języku, a następnie dokonać wyboru

poprzez kliknięcie jednej z dwóch opcji - języka angielskiego lub języka polskiego, które pokazują się na rozwijanej liście ukazanej na rysunku A.46.

Rysunek A.45. Strona ustawień w aplikacji ReSpire

Rysunek A.46. Wybór języka aplikacji

Przykładowe strony aplikacji po zmianie języka widoczne są na rysunku A.47.

Rysunek A.47. Widok przykładowych stron aplikacji - strony ustawień (po lewej) oraz edytora dźwięków (po prawej) - po zmianie języka na angielski