

**Robert Górska**

numer albumu: 42613

kierunek studiów: Informatyka

specjalność: Systemy komputerowe i oprogramowanie

forma studiów: studia niestacjonarne

**PROJEKT I IMPLEMENTACJA SYSTEMU WSPOMAGAJĄCEGO  
TRWAŁE ZAPAMIĘTYWANIE OBCOJĘZYCZNEGO SŁOWNICTWA Z  
WYKORZYSTANIEM TEORII KRZYWEJ ZAPOMINANIA HERMANNA  
EBBINGHAUSA**

**DESIGN AND IMPLEMENTATION OF A COMPUTER SYSTEM  
SUPPORTING MEMORISATION FOREIGN-LANGUAGE LEXICAL  
MATERIAL.**

praca dyplomowa magisterska

napisana pod kierunkiem:

**mgr inż. Artura Karczmarczyka**

Katedra Metod Sztucznej Inteligencji i Matematyki Stosowanej

Data wydania tematu pracy: 15.06.2021

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

Szczecin, 2021



## Oświadczenie autora pracy dyplomowej

Oświadczam, że praca dyplomowa magisterska pn. *Projekt i implementacja systemu wspomagającego trwałe zapamiętywanie obcojęzycznego słownictwa z wykorzystaniem teorii krzywej zapominania Hermanna Ebbinghausa* napisana pod kierunkiem mgr inż. Artura Karczmarczyka jest w całości moim samodzielnym autorskim opracowaniem sporządzonym przy wykorzystaniu wykazanej w pracy literatury przedmiotu i materiałów źródłowych. Złożona w dziekanacie Wydziału Informatyki treść mojej pracy dyplomowej w formie elektronicznej jest zgodna z treścią w formie pisemnej.

Oświadczam ponadto, że złożona w dziekanacie praca dyplomowa ani jej fragmenty nie były wcześniej przedmiotem procedur procesu dyplomowania związanych z uzyskaniem tytułu zawodowego w uczelniach wyższych.

Podpis autora: .....



Szczecin, dnia: ..... 15.06.2021 .....

## Streszczenie

Opanowanie słownictwa jest niezbędne do umiejętności posługiwania się obcym językiem, jednak na przeszkodzie do tego celu staje nieodłączna przypadłość ludzkiej pamięci - zapominanie.

Połączenie dostępnej obecnie technologii informatycznej z osiągnięciami psychologii, stwarza możliwość zapanowania nad utratą informacji przyswojonej w toku nauki.

Praca prezentuje przykład praktycznego wykorzystania teorii krzywej zapominania Hermanna von Ebbinghausa, do zaprojektowania systemu wspomagającego zapamiętywanie obcojęzycznego słownictwa, poprzez wyznaczanie i przeprowadzanie rozłożonej w czasie serii powtórek, wzmacniających trwałość śladu pamięciowego.

**słowa kluczowe:** np. krzywa zapominania, Hermann von Ebbinghaus, rozłożone w czasie powtórki, zapamiętywanie słów

## Abstract

Mastering the vocabulary is necessary for the efficient use of a foreign language, but an obstacle to this goal is the inherent affliction of human memory - forgetfulness.

The combination of currently available information technology with the achievements of psychology, makes it possible to control the loss of information acquired in the course of learning.

The work presents an example of the practical use of the Hermann von Ebbinghaus theory of the forgetting curve, to design a system supporting the memorization of foreign vocabulary, by determining and carrying out a series of repetitions over time, strengthening the durability of the memory trace.

**keywords:** forgetting curve, Ebbinghaus, spaced repetitions system, memorizing second language words

# Spis treści

<b>1</b>	<b>Wstęp</b>	7
<b>2</b>	<b>Pamięć i zapominanie</b>	9
2.1	Jak działa pamięć	9
2.2	Zapominanie w badaniach H. Ebbinghausa	10
<b>3</b>	<b>Od teorii naukowej do praktyki rynkowej</b>	12
3.1	Metody wspomagające zapamiętywanie	12
3.2	Porównanie istniejących rozwiązań rynkowych	14
3.2.1	Łatwość i komfort pracy z programem	18
3.2.2	Możliwość Importu, eksportu i edycji kart (fiszek)	19
3.2.3	Sposób przeprowadzenia fazy kodowania – nauki	20
3.2.4	Czasowa organizacja powtórek	20
3.2.5	Zastosowane metody samooceny	21
3.2.6	Podsumowanie porównania	22
3.3	Założenia projektu w świetle konkurencyjnych rozwiązań	23
<b>4</b>	<b>Projekt</b>	26
4.1	Algorytmy i logika pracy programu	26
4.1.1	Praca z kartą w fazie nauki	27
4.1.2	Rozłożone w czasie powtórki	31
4.1.3	Obliczanie odstępów pomiędzy powtórkami	33
4.2	Baza danych	36
4.3	Bezpieczeństwo	36
4.3.1	Zabezpieczenie procesu rejestracji i logowania	36
4.3.2	Autoryzacja zapytań do bazy danych	40
4.3.3	Inne zastosowane zabezpieczenia	44
4.4	Wymagania funkcjonalne i niefunkcjonalne	45
4.5	Diagramy LOFI	48

<b>5</b>	<b>Wybór technologii</b>	56
5.1	Kliencka strona aplikacji	56
5.2	Back-end	60
<b>6</b>	<b>Implementacja projektu</b>	66
6.1	Opis procesu tworzenia produktu	66
6.2	Ekrany gotowej aplikacji	67
6.3	Sprawdzenie działania systemu w kontakcie z użytkownikiem	75
<b>7</b>	<b>Podsumowanie</b>	78
7.1	Ocena stanu końcowego	78
7.2	Możliwości dalszego rozwoju	79
	<b>Podsumowanie</b>	78
	<b>Bibliografia</b>	80
<b>A</b>	<b>Dodatek</b>	83

# 1. Wstęp

Pomysł projektu, stojący za tematem niniejszej pracy, powstał z motywacji autora do polepszenia własnych umiejętności komunikacyjnych w języku angielskim, poprzez intensywną naukę słownictwa. Po doświadczeniach z zapamiętywaniem całych list słów, tworzonych na papierze, została odniesiona przynajmniej ta korzyść, że poznano wszystkie ograniczenia, czyniące na adepta obcego języka.

Można, oczywiście, zastanawiać się, czy warto aż tyle wysiłku poświęcać na opanowanie słownictwa obcego języka, skoro do osiągnięcia płynności w porozumiewaniu się w obcym języku słownictwo potrzebne jest na równi z opanowaniem gramatyki [1]. Niemniej, o znaczającej wadze posiadania bogatej leksyki, niech zaświadczą chociażby słowa prof. Davida Wilkinsa [2](tłum. z ang.): „Niewiele można przekazać nie znając gramatyki, ale niczego się nie powie, nie znając słów”.

Każdy, kto uczy się języka obcego, zadaje sobie w końcu pytanie o sposób nie tyle zapamiętania jeszcze większej ilości słów, co raczej ich nie zapominania z czasem [3]. Proces utraty raz pozyskanej wiedzy wydaje się nieunikniony i być czymś, do czego powinniśmy się przyzwyczać. Niemniej jednak, bolesne doświadczenie, jakim jest świadomość straty śladu w pamięci, prowadzącego do potrzebnej informacji, skłania nas do ponawiania poszukiwań metody, która zagwarantowałaby zapanowanie nad własną pamięcią.

Czy współczesna technologia, tak wszechobecna w wielu innych dziedzinach, jest w stanie poradzić sobie w organizowaniu pracy naszego własnego mózgu? Jeśli tylko połączymy ją z wiedzą, jaką o pracy tego organu dostarcza współczesna nauka, jest to jak najbardziej możliwe.

Celem niniejszej pracy stało się więc zaprojektowanie systemu informatycznego, które korzystając ze zdobytych nauki o pamięci, przejmie od użytkownika ciężar organizacji procesu nauki i pokieruje nim tak, jak to sugeruje nowoczesna psychologia, czyli kładąc nacisk tak na proces samego zapamiętania, jak i na późniejsze przypominanie nauczonych faktów, zanim zostaną one zapomniane.

Powyższe zamierzenie autora, jest wynikiem przeświadczenia, opartego zarówno na doniesieniach naukowych na temat procesów zachodzących w mózgu [4], oraz własnym doświadczeniu, że nie tyle zaawansowanie algorytmów obliczeniowych, ile usystematyzowanie pracy w pierwszych dniach nauki, jest gwarancją obniżenia stopnia porażek.

W realizacji założonego celu, wykorzystano teorię krzywej pamięci, opracowaną na początku XX wieku przez niemieckiego psychologa, Hermanna von Ebbinghausa, zgodnie z którą, prawidłowe przypominanie raz przyswojonego faktu, skutkuje jego trwałym zapamiętaniem [5]. Realizujący założenia powyższej teorii system webowy, staje się więc niejako zewnętrznym organizerem pamięci ucznia. Odpowiada za przypominanie mu o

robieniu powtórek i dba o to, żeby mógl wykorzystać każdy dogodny czas na wykonanie przygotowanych zadań. Odciąża go, co pozwala mu się skupić na nauce, bez niepotrzebnego balastu pilnowania terminów. Nauka staje się przez to przyjemniejsza, zwłaszcza, że szybko, bo już po pierwszych dwóch dniach, obserwuje on zwiększone efekty swojego wysiłku. Przekonanie, że praca nie idzie na marne, jest wtedy największym stymulantem do jej kontynuowania. Niniejszą pracę podzielono na siedem rozdziałów. Następują cy po tym wprowadzeniu, drugi, przedstawia aktualny stan wiedzy psychologicznej i neurobiologicznej związanej z pracą mózgu w kontekście zapamiętywania informacji przez człowieka. Autor, zapoznaje tu również czytelnika, z tajemnicą wiecznej pamięci, jaką odkrył profesor Ebbinghaus. Wnioski, będące owocem jego prac, oraz późniejszych mu naukowców w tej dziedzinie, starają się wykorzystywać twórcy komercyjnych narzędzi wspomagających ucznia w jego wysiłkach, o czym traktuje rozdział trzeci. Kolejny, czwarty rozdział, przybliża poszczególne etapy pracy projektowej nad modelem użytkowym, w którym zastosowane zostaną, razem niespotykane w innych rozwiązaniach, strategie pracy z uczniem. Rozdział piąty stanowi wprowadzenie do części praktycznej pracy i poświęcony został problemowi doboru technologii do założonych celów funkcjonalnych projektu. Zwieńczenie tworzenia działającego produktu, przedstawia część szóstą, w której autor przybliża etapy implementacji zaprojektowanego rozwiązania. Na koniec, w podsumowaniu, będzie możliwość poznania perspektyw dalszego rozwoju projektu, zmierzających do wzbogacenia go o kolejne użyteczne z punktu widzenia ucznia, funkcjonalności, oraz transformowania go w platformę do nauki z nauczycielem.

## 2. Pamięć i zapominanie

### 2.1 Jak działa pamięć

Pomimo tego, że człowiek jest najbardziej przystepnym obiektem wszelkich badań naukowych, o pracy jego pamięci nadal niewiele wiemy na pewno. Niemal od 150-ciu lat prowadzone doświadczenia psychologów, nie pozwoliły nawet na stwierdzenie, czy pamięć ma określone położenie w mózgu, czy jest rozproszonym środowiskiem sieciowym, albo, czy istnieje jeden, czy wiele sposobów powstawania śladu pamięciowego [6].

Na chwilę obecną, najbardziej prawdopodobnym wyjaśnieniem problemu, jest konsepcja D.O. Hebb'a, która zakłada, że proces uczenia się, a więc i zapamiętywania, wpływa na wzrost siły połączeń synaptycznych pomiędzy neuronami mózgu [6]. Obecne badania, nie tylko to potwierdzają, ale są też dowodem na trwałość zachodzących zmian w tkance ośrodkowego układu nerwowego, pod wpływem przetwarzanej informacji[7]. Część tych zmian rzeczywiście zmienia na dłucho strukturę mózgu, część jednak ulega szybkiej rewersji, nawet już po ułamku sekundy [6], co zasugerowało badaczom istnienie dwóch głównych typów pamięci: pamięci krótkotrwałej i długotrwałej.

Okazało się jednak, że podstawą zmian synaptycznych w każdym z wymienionych rodzajów pamięci jest inny mechanizm [8]. W toku powstawania trwałego śladu pamięciowego, dochodzi do faktycznych zmian fizycznych w mózgu, w odróżnieniu od łatwo odwracalnych zmian chemicznych, będących podstawą "pracy" pamięci krótkotrwałej.

Z koniecznością zachowania jasności tego wprowadzenia, pomijam fakt, że sprawa jest o wiele bardziej skomplikowana, niż ją tutaj przedstawiam. Choćby z tej przyczyny, że sama pamięć długotrwała ma, według współczesnych badaczy, wiele odmian. Jednak z punktu widzenia niniejszej pracy, najważniejszym nurtującym pytaniem jest: jakie czynniki sprzyjają uzyskaniu największej trwałości i czy jest możliwość zaimplementowania ich w postaci systemu informatycznego, wspomagającego naukę.

Jest to o tyle istotne, że niektóre badania sugerują nieograniczoną trwałość śladu pamięciowego, co znaczyłoby, że człowiek przechowuje w swojej pamięci dosłownie wszystko, czego się był w stanie nauczyć, zapamiętać. Tylko niemożność wydobycia informacji, skłania nas do przyjęcia założenia, że ją zapomnieliśmy na zawsze. Na pewno jednak, nie ulega w chwili obecnej wątpliwości fakt, że kluczem do przeniesienia danych z magazynu pamięci krótkotrwałej do długotrwałej, jest powtarzanie tejże informacji [6].

Koło wiedzy ludzkości o samej sobie, zamknięte w starożytnym stwierdzeniu: *Repetitio mater studiorum est... .* Po dwudziestu z górami wiekach, potwierdzamy tę prawdę, że powtarzanie leży u podstawa nauczania.

Pierwszym pytaniem, które ciśnie się teraz na usta, jest: Zatem, ile razy trzeba coś powtórzyć, aby zachować to na zawsze? Choć wydaje się ono dziecięco naiwne, było

drogowskazem poszukivań tego Świętego Graala pamięci. Niektórzy [9], znają już odpowiedź: dwanaście razy.

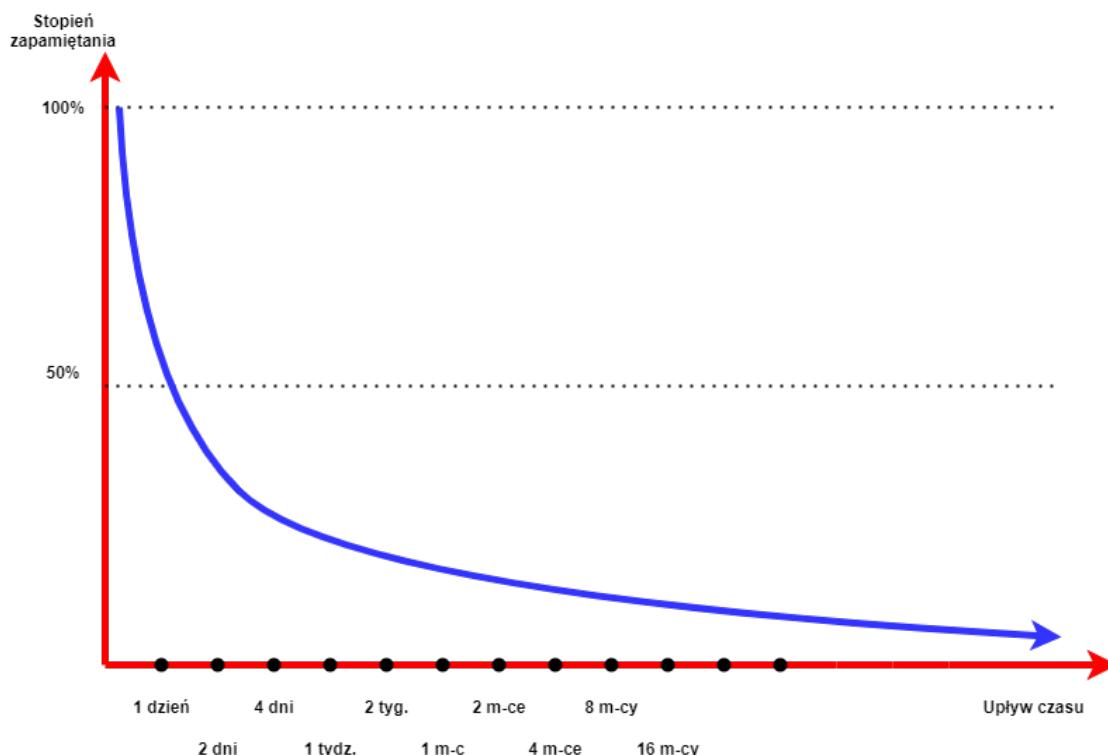
Oczywiście, nie chodzi tu o proste wypowiedzenie na głos tego co chcemy zapamiętać dwanaście razy pod rząd, jednym tchem. Trzeba jeszcze wiedzieć, kiedy powtarzać. Byli jednak badacze, którzy i na to pytanie znaleźli odpowiedź.

## 2.2 Zapominanie w badaniach H. Ebbinghausa

W poprzednim rozdziale wspomniałem o dwudziestu wiekach zmagania człowieka z badaniem własnej pamięci. Trudno tak naprawdę ocenić to rzetelnie, jednak więcej racji w tym względzie ma Hermann von Ebbinghaus, autor nieocenionej pracy, *Über das Gedächtnis* (O pamięci) (1885), który upatrując w początkach badania pamięci, założka samej psychologii, jako nauki, napisał: "Psychologia ma długą przeszłość, ale krótką historię" [10].

Był pierwszym badaczem, który podjął się systematycznych badań nad fenomenem pamięci, a zwłaszcza jej trwałości. Jako pierwszy też, wprowadził metodyczne podejście do swoich eksperymentów, zmierzając do ograniczenia ilości zmiennych, mogących wpływać na pracę mózgu ludzkiego podczas procesu zapamiętywania. W swoich badaniach nad trwałością efektu pamięciowego, manipułował długością okresu czasu pomiędzy zapamiętaniem, zakodowaniem informacji, a jej odtworzeniem.

Żmudne prace, pozwoliły mu na określenie dynamiki zapominania w funkcji czasu, którą obecnie przedstawia się jako krzywą zapominania Ebbinghausa (Rys.2.1).



Rysunek 2.1: Model przechowywania informacji w pamięci człowieka (Opracowanie własne na podstawie *Replication and Analysis of Ebbinghaus' Forgetting Curve* JMJ Murre)

I chociaż nie można przywiązywać dużej wagi do określonych wartości wskazywanych przez ten wykres, to jednak jego kształt, obrazujący szybkość zmian ilości przechowywanego w pamięci materiału, pozostaje taki sam [11], pomimo upływu lat i wielu badań, przeprowadzonych przez jego następców. Niezmienny zatem, pozostaje również, wynikający z kształtu krzywej fakt, że największy spadek ilości zapamiętanych danych następuje zawsze krótko po ich zakodowaniu.

Oceniając wielkość wysiłku, jaki trzeba włożyć w ponowne opanowanie informacji po upływie zmiennego okresu czasu od jej zakodowania, czyli pierwotnego zapamiętania, Ebbinghaus wyprowadził pojęcie efektu przerw [6]. Pojęcie to odpowiada angielskiemu określeniu *spacing effect* i jest określeniem zjawiska, zgodnie z którym dwie kolejne próby odtworzenia zakodowanej informacji zbliżone do siebie czasowo, skutkują słabszym utrwaleniem (retencją), niż te, bardziej od siebie odległe [12]. Formalną wersję tego zjawiska, przedstawia prawo Josta, które mówi, że "jeżeli dwa skojarzenia mają jednakową siłę, ale różny jest ich wiek, to dalsze uczenie się przyniesie większą korzyść skojarzeniu starszemu"[13].

W ciągu wielu lat badań, nie zabrakło też i takich, które starały się wyjaśnić, jak prawidłowości odkryte przez H. Ebbinghausa i pozostałych badaczy, można odnieść do rzeczywistych sytuacji, gdzie w grę wchodzi zapominanie uczonego materiału języka obcego. Wspomnienie tego aspektu badań, jest o tyle istotne, że w sposób bezpośredni odnosi się do tematu niniejszej pracy.

Doświadczenia takie przeprowadzał choćby Bahrick (1984) [14], a dotyczyły retencji słownictwa hiszpańskiego u studentów, którzy ukończyli naukę tego języka nawet przed 50-ciu laty. Eksperyment ten udowodnił, że uczniowie uczący się języka jedynie przez pół roku szkolnego, po 5-ciu latach nie pamiętali już niczego. Dla kontrastu, czas pierwotnej nauki sięgający 5-ciu semestrów, pozwalał na utrzymanie znajomości ponad 50% słownictwa, nawet po upływie ponad 25-ciu lat od ukończenia szkoły. Oprócz potwierdzenia ogólnego kształtu krzywej zapominania, uzyskał on też dowód, że długość fazy kodowania i lepsze początkowe utrwalenie materiału, pozwala na osiągnięcie znacząco lepszej retencji zapamiętanych informacji w pamięci długotrwałej, a może trzeba by powiedzieć, zakładając, że pamiętamy niemal wszystko: łatwiejsze jej wydobywanie.

Uzyskując tak dobrze udokumentowaną instrukcję sposobu użytkowania własnej pamięci, powinniśmy spodziewać się wyraźnego przełożenia na efekty nauczania w systemie szkolnictwa, co jednak nie ma miejsca. Teoria, mimo swojej słuszności, musi jeszcze pozostać wprowadzona w życie, a w tym przypadku, wiele zależy od zastosowanych metod "implementacji", co poruszę w kolejnym rozdziale.

### 3. Od teorii naukowej do praktyki rynkowej

#### 3.1 Metody wspomagające zapamiętywanie

Nauka słownictwa jako dział nauki języka obcego, to pole bogate w półprawdy i złudzenia. Czy, dla przykładu, jakakolwiek metoda zapamiętywania jest tu potrzebna w ogóle? Bo jeśli do komunikacji mogłoby wystarczyć opanowanie listy 1000 słów, jak twierdzą niektórzy blogerzy internetowi, to żadna metoda nie jest potrzebna - taką ilość danych można opanować mimochoDEM. Patrząc jednak na problem fachowym okiem badaczy jazykoznawcy, wygląda to zgoła inaczej. Hazenberg i Hulstijn [15], twierdzą, że do w miarę sprawnej komunikacji, potrzebny jest zasób 3000 - 10000 słów. Bardziej konkretny jest Laufer[16], uznający, że do rozumienia tekstu pisanego wymagana jest znajomość 95% zastosowanego słownictwa, a Wilkinson [17], że nawet 98%. Przy takiej skali "problemu", nauka pamięciowa słownictwa powinna być metodyczna.

Ponieważ w toku opanowywania słownictwa obcego, każdy uczeń wypróbowuje różne metody, można po pewnym czasie, dokonać samemu oceny ich użyteczności i zaobserwować ścieżkę ewolucji metod nauki własnej.

Pierwsza, z którą niemal każdy uczeń ma do czynienia na samym początku nauki zapamiętywania, opiera się na powtarzalnym przepisywaniu informacji mającej ulec zapamiętaniu. Wymaga ona co prawda najmniejszego wkładu pracy umysłowej, ale poświęcony czas i aktywność fizyczna, każe zastanowić się nad rachunkiem zysków.

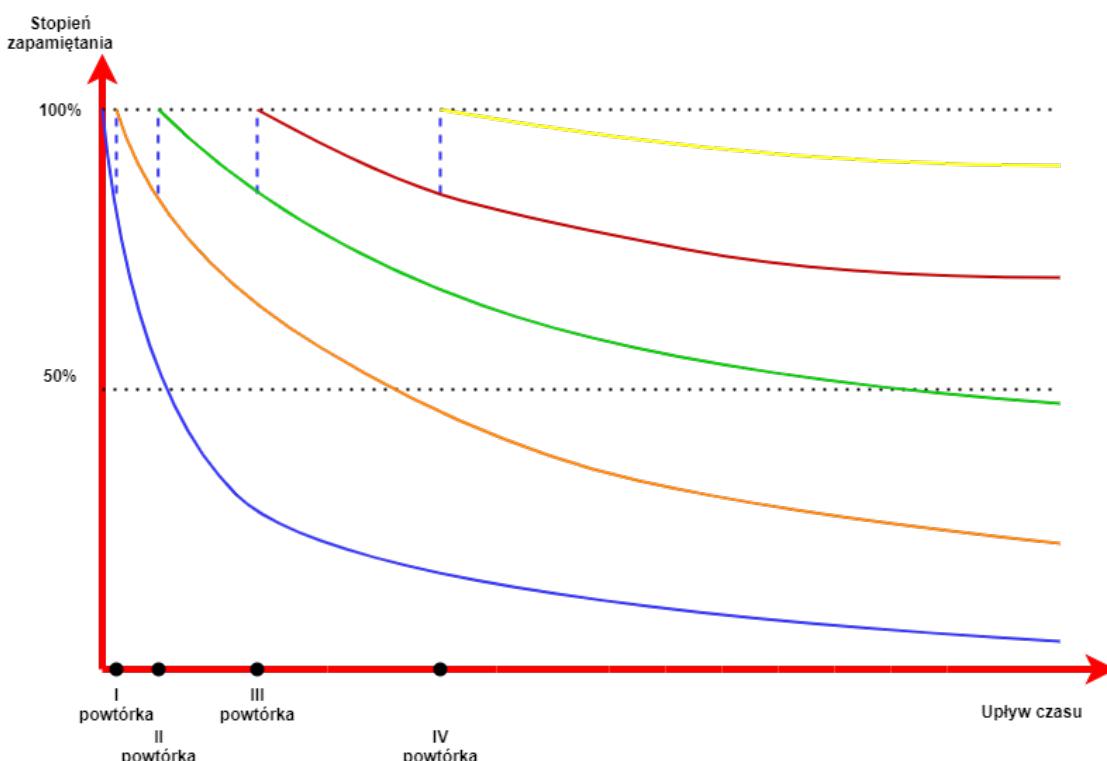
Dowód na mniejszą skuteczność przepisywania uczonego materiału, dostarcza chociażby Joe Barcroft [18], w którego doświadczeniu, grupa studentów z wyraźnie gorszym skutkiem pamiętała po 2 dniach od nauki, te pary (słowo – tłumaczenie), które przepisywali, od tych, które starali się zapamiętać bez przepisywania ich. Autor dowodzi przez to rolę większego zaangażowania mózgu w proces pamięciowy, co skutkuje większym stopniem retencji opanowywanego materiału.

Główne nakład pracy, stojący za tą metodą, powoduje, że w celu zwiększenia efektywności zapamiętywania, uczeń przechodzi przy nauce słownictwa, do tworzenia list, które poprzez wielokrotne czytanie, opanowuje pamięciowo. Polepsza to ilość materiału, który uczeń jest w stanie zapamiętać w tym samym czasie, co przy poprzednim sposobie. Nie stanowi teraz problemu zapamiętanie (rozumiane jako umiejętność odtworzenia informacji w czasie do 24 godzin po nauce), 100 obcych słów I zwrotów dziennie. Przy czym osiągnięcie to, jest jak najbardziej do powtórzenia w kolejnych dniach. Kłopot zaczyna sprawiać jednak próba odtworzenia zapamiętanego materiału po np kilku dniach. Właściwie, gdyby nie omówione w poprzednim rozdziale zapominanie, listy byłyby wystarczająco efektywną metodą organizującą do nauki słów.

Jednak, problemem nie jest zapamiętywanie, ale utrata raz opanowanej pamięciowo

informacji, wraz z upływem czasu. I tutaj wróćmy na moment do praktycznego wymiary wspomnianej wcześniej teorii krzywej pamięci.

Bazując na badaniach Hermanna von Ebbinghausa, Pimsleur w 1967 roku zaproponował nowy psychologiczny model uczenia pamięciowego, określany jako „graduated interval recall”. Założył, że przypomnienie poprzednio zapamiętanej informacji powinno nastąpić tuż przed jej całkowitą niemożnością wydobycia z pamięci. Opierając się na fakcie, że po każdym kolejnym przypomnieniu, krzywa znajomości (zapominania) opada coraz łagodniej w czasie, stosował skutecznie, wydłużone każdorazowo interwały czasowe pomiędzy powtórkami (Rys.3.1). Wysoką skuteczność takiej strategii, udokumentowano kilku innych badaczy Lado, Oxford.



Rysunek 3.1: Rozłożone w czasie powtarzanie, jako metoda przeciwdziałania zapominaniu (Opracowanie własne na podstawie *Replication and Analysis of Ebbinghaus' Forgetting Curve* JMJ Murre)

Udowodniona tym samym konieczność przypominania raz opanowanych danych, sprawia, że trwałość tak stworzonej listy zaczyna być największą przeszkodą w przeprowadzaniu rozłożonych w czasie powtórek. Rozbieżności stopnia zapamiętania pomiędzy poszczególnymi słowami, wymuszają konieczność wyznaczania indywidualnych terminów powtórek dla każdej pary słownikowej. Aby dana lista mogła spełnić to wymaganie, należało ją fizycznie podzielić, wręcz porozcinać na pojedyncze rekordy. Zapewne właśnie to, stało za pomysłem tworzenia indywidualnych kart, zwanych też fiszkami, zawierającymi pojedyncze słowo wraz z jego tłumaczeniem w obcym języku.

O ile jednak pierwotny pomysł wykorzystania do tego celu papierowych kart, zyskał bardzo szybko na popularności, o tyle równie szybko użytkownicy zaczęli dostrzegać niedogodności systemu. Ręczne przekładanie kart do kolejnych przegródek z datami

powtórek, było dużym ograniczeniem w skutecznym wykorzystaniu tych tzw „fiszek”.

Problem został rozwiązyany wraz z szybkim rozwojem technologii informatycznych. Od teraz, pieczę nad organizacją powtórek materiału, przejął na siebie komputer. Nowoczesna technologia pozwala przede wszystkim na wprowadzenie dużego stopnia elastyczności i szybkości reakcji, w pracy ucznia z posiadającym wiedzę o jego postępach, systemem informatycznym.

Szeroki wachlarz możliwości, oraz pomysłowość twórców tego typu aplikacji, zaowocowały w przeciągu ostatnich 20-tu lat, powstaniem licznych przykładów tego rodzaju programów. Porównaniem ich głównych założeń, poświęć kolejny rozdział niniejszego opracowania.

## 3.2 Porównanie istniejących rozwiązań rynkowych

Ponieważ pomysł stworzenia niniejszego projektu, wynikł z osobistej potrzeby posiadania zaufanego narzędzia do organizacji nauki obcojęzycznego słownictwa, niniejszy przegląd i ocena rozwiązań rynkowych, pozostaje pod wpływem moich własnych oczekiwani, stawianych tego typu aplikacji.

Postaram się jednak, aby dokładność opisu funkcjonalności porównywanych systemów, pozwoliła także innym osobom, zainteresowanym tym tematem, na wypracowanie własnego zdania o prezentowanych aplikacjach, tym bardziej, że w przeprowadzonym opisie porównawczym, opieram się na naukowo potwierdzonych warunkach i zależnościach, jakim powinna odpowiadać nauka pamięciowa obcojęzycznej leksyki.

Sporym zaskoczeniem, po rozpoczęciu poszukiwań dostępnych na rynku rozwiązań, była dla mnie, ich ogromna liczba, sięgająca 30-tu aplikacji. Busuu, Memrise, HelloTalk, Mondly, FluentU, TinyCards, Quizlet, Flashcards+, czy FlashcardDeluxe, to tylko niektóre z nich. Nawet po oczywistym odjęciu od tej ilości, systemów nie wykorzystujących w pracy z użytkownikiem, automatycznego przygotowywania powtórek na bazie krzywej Ebbinghausa, pozostało i tak ok 12-tu, wymagających bliższego przyjrzenia się. Były to, między innymi: SuperMemo, Fiszkontakte, Anki, Mnemosyne, Memrise, oraz Repetitions.

Ta obfitość, sugeruje nie tylko duże zapotrzebowanie na podobne narzędzia, ale świadczy też o ogromnej różnorodności zastosowanych rozwiązań, co z kolei pozwala podejrzewać, że preferencje użytkowników są równie zróżnicowane. Stanowi to poniekąd uzasadnienie subiektywności niniejszego przeglądu konkurencyjnych, dla mojego projektu, programów.

Całokształt oceny danego systemu, można uzyskać dopiero po jego przetestowaniu, z użyciem odpowiednio licznego pakietu kart, pytanie-odpowiedź, w okresie czasu, pozwalającym na wykształcenie pamięci długotrwałej.

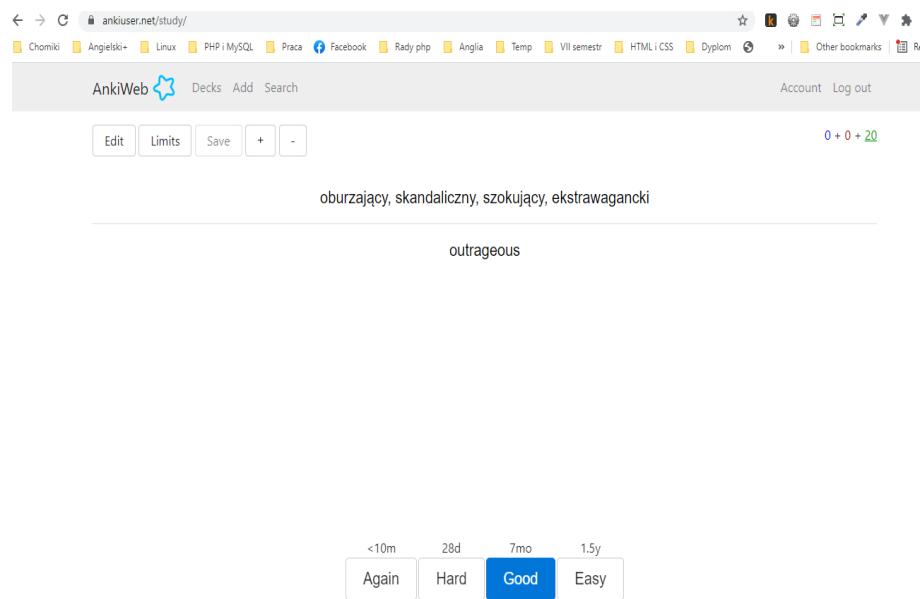
Dokonując wyboru aplikacji do testów, kierowano się kryteriami, które miała spełnić również ta, powstająca w ramach bieżącego projektu. Pod uwagę wzięto zatem, następujące funkcjonalności:

- możliwość wprowadzania własnych zestawów fiszek
- możliwość odzyskiwania fiszek z aplikacji jako zrzut do pliku
- możliwość uruchamiania aplikacji na urządzeniach mobilnych i praca w przeglądarce internetowej
- zasada działania oparta o teorię rozłożonych w czasie powtórek

Ostatecznie, na podstawie wymienionych kryteriów, udało się wybrać cztery systemy, do dalszego testowania. Były to następujące produkty rynkowe: Anki, Mnemosyne, SuperMemo i Fiszkontakte.

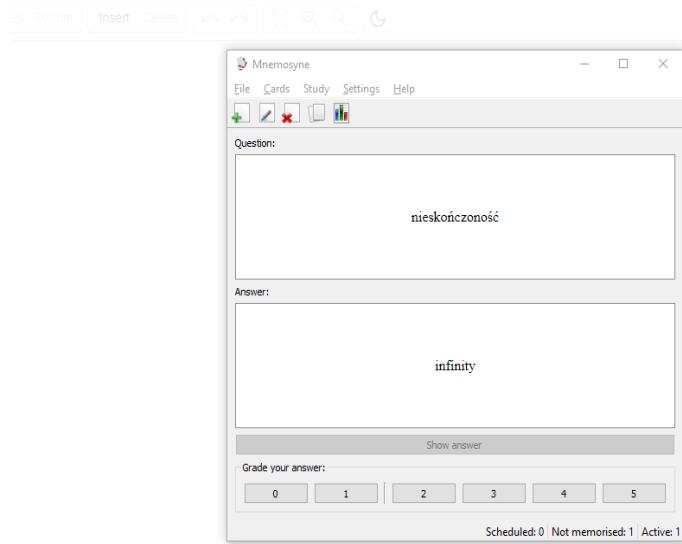
### Anki

(Rys.3.2) Platforma do nauki pamięciowej w oparciu o system rozłożonych w czasie powtórek, udostępniona do użytku 5 października 2006 roku. Anki to program udostępniany na zasadzie open source, umożliwia zarówno korzystanie z udostępnionych przez społeczność zbiorów kart, jak i dostarczanie własnych. Jego nazwa, w tłumaczeniu z języka japońskiego, oznacza zapamiętywanie. Rozbudowa jest możliwa poprzez instalowanie dodatkowych wtyczek rozszerzających możliwości programu. Aplikacja uruchamia się na komputerach z systemami Windows, Linux, macOS, a także na urządzenia z systemem Android i w przeglądarkach internetowych. Działanie programu jest oparte o pierwsze wersje algorytmu zastosowanego przez Piotra Woźniaka w jego aplikacji SuperMemo.



Rysunek 3.2: Wygląd interfejsu programu Anki

**Mnemosyne** (Rys.3.3) To, podobnie jak powyżej opisana platforma, również system open-source'owy, którego zasada działania jest oparta o model krzywej zapominania Ebbinghausa. Stworzony w 2003 roku przez Petera Bienstmana. Nazwa jego pochodzi od imienia greckiej bogini pamięci, posiada kod źródłowy w języku Python. Podobnie jak Anki, korzysta z jednej z pierwszych wersji algorytmu SuperMemo, SM-2. uruchamiany na komputerach z systemami Windows, Linux, macOS. Korzystanie z niego na urządzeniach z systemem android, wymaga synchronizacji poprzez program na komputerze desktopowym.



Rysunek 3.3: Wygląd interfejsu programu Mnemosyne

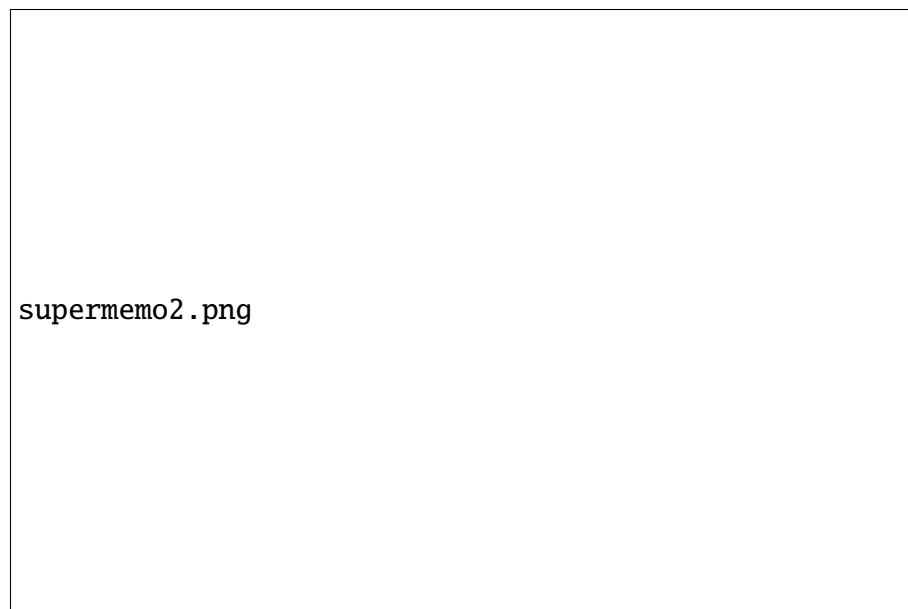
**Supermemo** (Rys.3.4) Stworzony w 1985 roku przez dr Piotra Woźniaka, w oparciu o model opracowany przy współpracy z dr Edwardem Gorzeląńczykiem. Występuje jedynie w wersji komercyjnej, dostępnej na komputerach z systemami Windows, Linux, macOS, urządzeniach z systemem Android i iOS. Można z niego korzystać także w przeglądarkach internetowych. Po 2000 roku, powstały 2 niezależne linie programowe rozwoju aplikacji: SuperMemo 2000, opracowywane nadal przez dr Piotra Woźniaka, oraz SuperMemo MSM, które jest wersją porównywana w niniejszym opracowaniu, tworzoną przez polską firmę SuperMemo.

**Fiszkontakte** (Rys.3.5) Program powstał w 2010 roku, początkowo jako serwis internetowy, w późniejszych latach, również dostępny na komputerach desktop z systemami Windows, Linux i macOS, oraz urządzeniach mobilnych Android i iOS. Obecnie występuje w wersji bezpłatnej z ograniczeniami, oraz komercyjnej Premium, bez ograniczeń, w tym dotyczących ilości posiadanych kart słownikowych na koncie. Strona internetowa Fiszkontakte, zdobyła w 2011 roku tytuł *Złota Strona Roku* tygodnika *Wprost*.

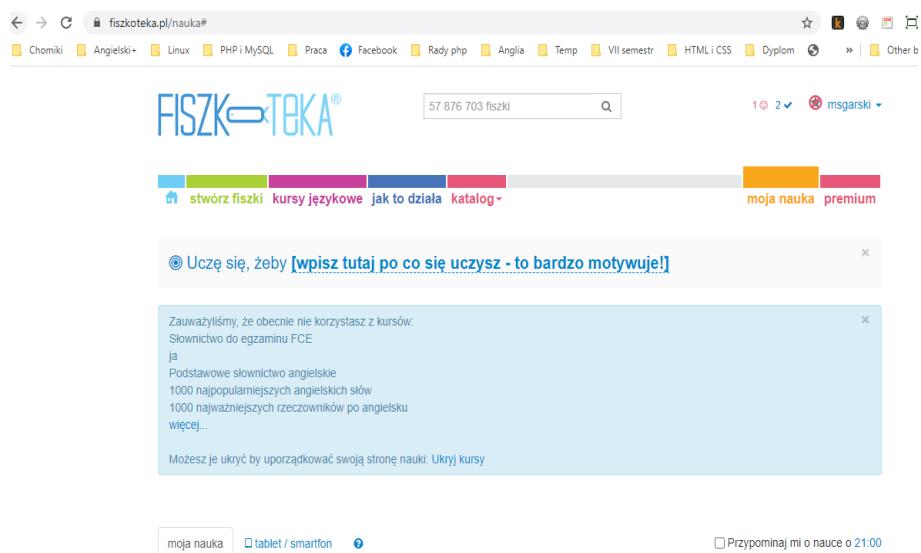
Porównawcza charakterystyka, wymienionych wyżej programów, przeprowadzona w odniesieniu do wspólnego zestawu cech, została przedstawiona poniżej.

Ocenie poddano następujące parametry:

- Łatwość pracy z aplikacją – przyjazność dla użytkownika
- Możliwość Importu, eksportu i edycji kart (fiszek)
- Sposób przeprowadzenia fazy kodowania - nauki
- Czasowa organizacja powtórek
- Zastosowane metody samooceny



Rysunek 3.4: Wygląd interfejsu programu SuperMemo



Rysunek 3.5: Wygląd interfejsu programu Fiszkontakteka

### 3.2.1 Łatwość i komfort pracy z programem

Idealnym narzędziem jest takie, które można w pełni używać od samego początku. Użytkownik, w tym wypadku uczeń, ma za zadanie nauczyć się na pamięć słówek, przy pomocy aplikacji – nie zaś nauczyć się obsługi aplikacji, przy pomocy nauki słówek. Dlatego program powinien być prosty w obsłudze i nie wprawiać ucznia w zakłopotanie, co do prawidłowego ustawienia wszystkich opcji. Samodzielne „testowanie” wszystkich ustawień, tylko rozprasza użytkownika, powodując, że ewentualne niepowodzenia skłonny jest złożyć na konto programu, nie zaś np swojego braku konsekwencji w pracy.

Wymóg prostoty i łatwości startu pracy z aplikacją, spełniają tu wszystkie wymienione, oprócz Anki. Praktycznie, po dokonaniu importu kart, w przypadku Mnemosyne, Fiszkontakte i SuperMemo, można rozpocząć ich niezakłóconą naukę. Umożliwiają, co prawda ingerencję w ilość materiału, jaki jesteśmy w stanie przerobić, ale nie wpływa to na utrudnienie pracy z systemem. Programy te, nie wymagają żadnych innych regulacji, czy ustawień. W przypadku Anki jest wręcz konieczność wstępnego dopasowania programu do swoich wymogów. Domyślne ustawienia tej aplikacji, dopuszczają dla przykładu, przemieszanie nauki nowych kart z powtórkami. Jest to powodem rozproszenia uwagi użytkownika, która powinna być skupiona na kodowaniu nowych treści i kotwiczeniu ich w pamięci. Innym przykładem takiej problematyycznej, domyślnej opcji, jest jednoczesne dwukierunkowe przeprowadzanie powtórek. Zgodnie z wprowadzeniem naukowym do niniejszej pracy, należy rozróżnić i oddzielić czasowo od siebie dwa kierunki nauki słów, tzn L1 -> L2 i L2 -> L1, ponieważ każdy z nich odpowiada innemu rodzajowi znajomości obcego słownictwa. Jeden kierunek powinien być dominujący, np L1 -> L2 (produktywny) w przypadku potrzeby używania słownictwa w mowie, podczas gdy drugi (reproduktywny), należy wprowadzać na dalszym etapie opanowania leksyki. Tutaj, albo opcja działa od samego początku, albo wyłączamy ją na dobre. Program Anki, narzuca początkującemu użytkownikowi konieczność ingerencji w opcje systemu już na samym początku pracy z tymże.

Wiele tych opcji, jak: zmiana długości odstępu czasowego dla „łatwy” kart, procentowe ustawienie początkowej łatwości, premia odpowiedzi „łatwa” (opcja oceny znajomości hasła), czy modyfikator przerw, wymagają fachowej wiedzy z dziedziny psychologii pamięci, więc nie powinny w ogóle być dostępne z poziomu ucznia. Ten z kolei, szukając rozwiązania jednej niedogodności, napotyka po drodze, możliwość regulacji wielu innych ustawień, które mogą mieć wyraźne przełożenie na efekty pracy z aplikacją. Udostępnione przez społeczność Anki, liczne rozszerzenia i wtyczki, poprawiające wygląd i działanie aplikacji, dodatkowo potęgują frustrację i niepewność. Uczeń, zamiast poświęcić cały wysiłek na pracę ze słownikiem, rozprasza swoje myśle i czas na próbach „poprawy” programu. W ten sposób, uświadomiona możliwość, stwarza nieświadomiony przymus skorzystania z niej.

Trudno także, od powyższego, oddzielić odczucie przyjemności pracy z daną aplikacją, bo po części wpływa na nią złożoność samego programu, po części zaś, opracowanie graficzne i umiejscowienie (dostępność) dla ucznia, poszczególnych modułów systemu.

Pod tym względem najlepiej ocenić można Fiszkontakte. Jej interfejs jest przezroczysty, a dostęp do wszystkich funkcji i zasobów, intuicyjny. Praca z kartami, zarówno w fazie nauki, jak i powtórek, jest przyjemna, szata graficzna niedrażniąca kolorami lub nadmiarem elementów. Najważniejsze treści, takie jak pytanie, hasło, odnośniki do

plików dźwiękowych, są pozycjonowane zgodnie z oczekiwaniami użytkownika, który dzięki temu sprawniej pracuje z aplikacją.

Na drugim miejscu, pod tym względem, plasuje się SuperMemo, którego głównym mankamentem jest utrudnione wyszukiwanie niektórych opcji, np: importu i eksportu. Minusem w jego przypadku jest także, wyświetlanie testowanych haseł i ich odpowiedzi przy lewej krawędzi strony, gdy zaraz potem wzrok użytkownika musi przewędrować na środek dolnego paska w celu dokonania oceny.

Nieco gorzej pracuje się natomiast, z pozostałymi dwoma systemami, Mnemosyne i Anki, których szara kolorystyka tła, słabo kontrastuje z czarnym kolorem treści haseł. To słabe wyróżnienie najważniejszych informacji programu, dodatkowo w Mnemosyne potęgowane jest zbyt małą czcionką. Ogólnie mówiąc, praca z dwoma ostatnimi aplikacjami w warunkach słabszego oświetlenia, jest nieco męcząca.

### 3.2.2 Możliwość Importu, eksportu i edycji kart (fiszek)

Import własnych kart – fiszek, jest podstawową funkcjonalnością oczekiwana na początku pracy z testowanymi systemami. Trzeba zaznaczyć, że wszystkie umożliwiają ładowanie gotowych, komercyjnych, pakietów kart, w postaci plików z rozszerzeniami specyficznymi dla każdego z programów, oraz plików przygotowanych w dwóch najpopularniejszych, czyli Supermemo i Anki. Ich użytkownicy mogą dzięki temu zmienić używany program i nadal korzystać z dotychczasowych zestawów danych. Ze względu jednak na indywidualne potrzeby leksykalne każdego ucznia, najlepszą metodą jest stworzenie własnej listy słówek do nauki i wprowadzenie ich do programu. Każdy z opisanych programów daje możliwość wprowadzania pojedynczych fiszek przy pomocy klawiatury. Brak jest pod tym względem różnic – w przypadku każdego z systemów, można zapisać w pojedynczej karcie różnorodne informacje, łącznie z przykładowym zdaniem i obrazkiem ilustrującym hasło do zapamiętania. Rozbieżności pojawiają się, gdy zachodzi potrzeba przeniesienia do programu wielu kart jednocześnie, wcześniej np zapisanych w arkuszu kalkulacyjnym, lub pliku tekstowym.

Fiszkontakte i Supermemo są tu najbardziej przyjazne, bo umożliwiają użytkownikowi proste wklejenie skopiowanej zawartości zbioru do okna formularza importu, z opcją wyboru znaczników formatujących: końca linii oraz rozdziału kolumn. Końcowy efekt widoczny jest dla użytkownika jeszcze przed zatwierdzeniem akcji. Program Anki, z kolei, ładuje dane jedynie w postaci przygotowanego pliku csv. W Mnemosyne, import własnej listy kart wymaga przygotowania go w postaci arkusza kalkulacyjnego programu OpenOffice, co jest dużym utrudnieniem dla przeciętnego użytkownika peceta, bazującego na Microsoft Office.

Eksport fiszek z programu jest gwarancją zachowania na przyszłość, osobistego zbioru słówek. Jedynie Anki umożliwia odzyskanie swoich kart w postaci pliku tekstowego. W Fiszkontakte, użytkownik uzyska tylko plik w formacie pdf, podczas gdy pozostałe dwa programy, Mnemosyne i SuperMemo, nie udostępniają takiej możliwości w ogóle. W tych dwu przypadkach, użytkownik musi liczyć się z całkowitą utratą tworzonej kolekcji, w wypadku rezygnacji z korzystania z aplikacji.

### 3.2.3 Sposób przeprowadzenia fazy kodowania – nauki

Pierwszy kontakt z przeznaczonym do nauki materiałem, i następujący wraz z nim etap zapamiętywania (kodowania), jest z punktu widzenia psychologii, najważniejszą fazą, determinującą późniejszą trwałość śladu pamięciowego.

Program Anki, jako jedyny zapewnia przeprowadzenie podsumowania partii materiału, poprzez odpytanie z niego, po upływie 1 minuty od skutecznego zapamiętania hasła. Również, jednie ta aplikacja, wyznacza pierwszą regularną powtórkę, jeszcze w okresie czasu, gdy hasło przechowywane jest w pamięci krótkotrwałej , czyli po 10-ciu minutach. Są to psychologicznie uzasadnione działania, mające na celu utrwalenie śladu pamięciowego na kolejne kilkadziesiąt godzin. Pozostałe systemy, pozwalają użytkownikowi wrócić do zapamiętanych treści, dopiero po 24 godzinach (Mnemosyne i Fiszkołek), lub nawet dopiero po 48 godzinach – SuperMemo, kiedy ryzyko utraty wiedzy, jest już bardzo duże.

Samo przeprowadzenie fazy demonstrowania pytań i odpowiedzi, nie jest źródłem znaczących różnic, pomiędzy badanymi programami. Jednym, wspólnym brakiem wszystkich systemów, jest nieuwzględnienie opcji przeuczenia, o której wspomniano w przeglądzie literatury naukowej. Posiada ona udowodniony wpływ na utrwalenie haseł, wymagających kilkukrotnej prezentacji do zapamiętania, a wymaga jedynie dodatkowych prezentacji uczonego słowa, już po stwierdzeniu przez ucznia faktu jego zapamiętania.

### 3.2.4 Czasowa organizacja powtórek

Głównym zastrzeżeniem podnoszonym na forach internetowych przez doświadczonych użytkowników testowanych aplikacji, był fakt powstawania częstych kumulacji powtórek do przeprowadzenia jednego dnia, znacznie przekraczających możliwości czasowe uczniów.

Trzeba tu zaznaczyć, że wszystkie wymienione aplikacje, pozwalają na wprowadzenie przez użytkownika ograniczenia ilości powtórek do przeprowadzenia w ciągu jednego dnia, ale nie przekłada się to na jednokrotnie ograniczenie ilości uczonych słów, których przewidywalne przecież terminy pierwszych powtórek spowodują przekroczenie wspomnianego ograniczenia. Dlatego też, ograniczenie to powszechnie nie zdaje egzaminu.

Co bardziej znamienne, każdy z wymienionych programów, pozwala przecież użytkownikowi na wprowadzenie ograniczeń również na ilość nowych słów do nauki jednego dnia. Pozornie takie rozwiązywanie powinno być wystarczające. Niestety, oba wspomniane ograniczenia, nie są nigdzie ze sobą zsynchronizowane. Wtedy, co prawda, uczeń nie może jednorazowo przekroczyć w nauce pewnej ilości materiału, ale przy kolejnym otwarciu aplikacji, znowu może korzystać z nieuszupełnionego limitu, lub też jest mu to zaproponowane. Zabrakło tu, ewidentnie, mechanizmu, który ograniczałby możliwość nauki nowego materiału, gdy spodziewane terminy powtórek były przeładowane.

Opisane problemy, dotyczące nadmiernego kumulowania się powtarzanego materiału, powodują ryzyko nieumyślnego opuszczania powtórek, z powodu zmęczenia nadmiarem pracy. To z kolei prowadzi szybko do pojawienia się niepowodzeń, które zniechęcają do nauki i korzystania z programu. Niestety ułomność ta nie została w żadnym testowanym systemie naprawiona. Anki, a właściwie społeczność tej aplikacji, wprowadziła kilka poprawek, mających pomóc w radzeniu sobie z tym problemem, jednak ich znalezienie, nauczenie się, oraz instalacja, stanowią wyraźny kłopot dla przeciętnego użytkownika, o

czym wspomniałem wcześniej.

### 3.2.5 Zastosowane metody samooceny

Praca z aplikacją testującą stopień utrwalenia pamięciowego słówek obcojęzycznych, polegać musi z konieczności na samoocenie przeprowadzanej przez użytkownika. Tylko on jest w stanie stwierdzić, czy jest w stanie dane słowo przywołać z pamięci, lub czy sprawiło mu to trudność. Uczciwość w stosunku do samego siebie, przejawia się w momencie, kiedy trzeba tę ocenę przenieść do programu testującego, wybierając jedną z dostępnych opcji. W tym aspekcie testowane aplikacje demonstrują chyba największą różnorodność.

Fiszkontakte posiada najprostszy system oceny, bo zarówno w fazie nauki, kiedy dopiero poznajemy hasło, jak i w czasie jego powtórki, pozwala jedynie na opcje oceny własnej wiedzy: wiem, lub nie wiem. Przy czym ocena „wiem” w powtórce, nie skutkuje poinformowaniem ucznia, kiedy będzie kolejne testowanie danego słowa. SuperMemo, z kolei, wprowadza stopień pośredni: „prawie”, mający odzwierciedlać pewne trudności, jakie napotkał uczeń podczas przypominania sobie hasła. Dodatkowo, program ten informuje, kiedy będzie kolejne spotkanie z tym słowem. Stopniując złożoność zastosowanej skali ocen, w aplikacji Anki, oprócz skali podobnej do SuperMemo, dochodzi jeszcze ocena „Łatwe”, co ma pomóc oddzielić słowa znane bardzo dobrze, z którymi uczeń na pewno nie ma żadnego problemu i pozwolić przydzielić im bardzo odległe terminy powtórek, bez obawy o ich utratę z pamięci.

Na samym końcu, pod względem stopnia skomplikowania samooceny, pozostał system Mnemosyne, posiadający aż 6-cio stopniową skalę liczbową, odpowiadającą kolejnym poziomom znajomości danego hasła. Niestety, mimo opisu słownego tej skali, umieszczonego w dokumentacji aplikacji, właściwe przyporządkowanie stopnia znajomości jest ogromnym problemem. Przy czym sam program, już ocenę na poziomie 2, a więc bardzo niskim, kwalifikuje jako zapamiętanie hasła.

W odróżnieniu od innych aspektów testowanych w wymienionych programach, sposób samooceny, zastosowany w każdym z nich, jest chyba najbardziej uzależniony od osobistych preferencji ucznia, choćby dlatego, że trudno tu się powołać na niepodważalne doniesienia naukowe, uzasadniające jedną bądź inną metodę. Niemniej jednak, istnieją pewne praktyczne przesłanki, które pomagają w podsumowaniu zastosowanych rozwiązań.

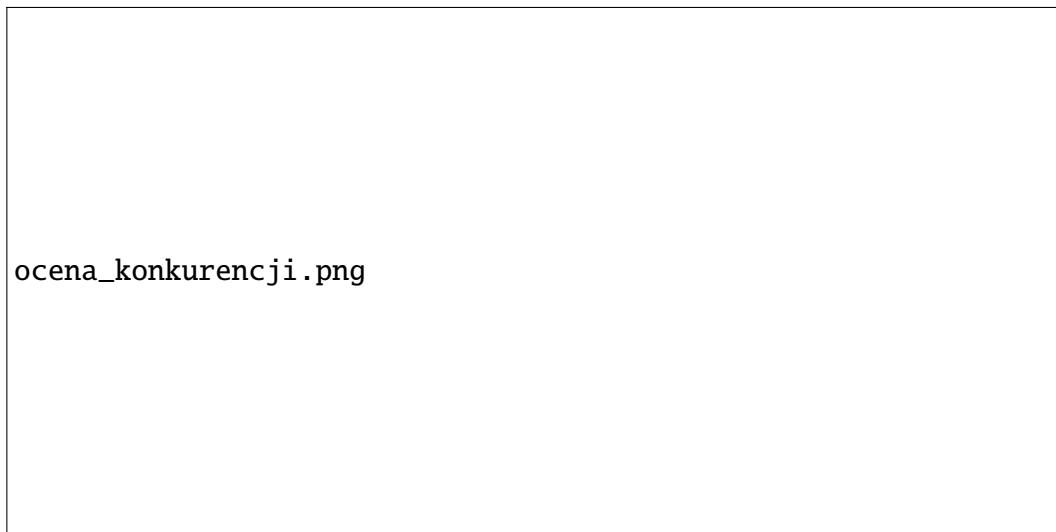
Zacznijmy od doświadczenia każdego z nas, jakie wynieśliśmy z wielu lat nauki szkolnej i nauki czegokolwiek. Nie ma w praktyce życia, czegoś takiego jak „wiem średnio”, bo taką wiedzą nie można się posłużyć; albo się coś wie, albo nie, a brak pewności powinien być interpretowany jako niewiedza. Z tego powodu skale samooceny, wykraczające poza dwa stopnie prezentowane w Fiszkontakte, są mylące, zwłaszcza 6 stopni zastosowanych w Mnemosyne.

Pewne uzasadnienie, ma dopuszczenie możliwości odznaczenia łatwych słów, zwłaszcza podczas pierwszej ich prezentacji, bo to pozwala ograniczyć ilość przyszłych niepotrzebnych powtórek. Niemniej jednak, informowanie ucznia o tym, za ile dni ponownie zobaczy dane słowo, może być źródłem powstania u niego wątpliwości, czy to nie za długo, lub czy nie za krótko – lepiej pozostawić go w niewiedzy, niech zda się na algorytm programu. Poza tym znajomość terminu testu, wg doniesień literatury, może wpływać na powstanie efektu egzaminu, co przejawia się trudnością w przypomnieniu sobie tego słowa

zarówno przed tym terminem, jak i po nim. Podsumowując, sposób i skala oceny zastosowana w programie Fiszkoteka, wydaje się być najbardziej przemyślana z praktycznego punktu widzenia i efektywności nauki użytkownika.

### 3.2.6 Podsumowanie porównania

W powyższym porównaniu ująłem celowo jedynie najistotniejsze funkcjonalności, jakie powinien posiadać system wspomagający zapamiętywanie w oparciu o krzywą zapomnienia. Ich krótka może, lista, odzwierciedla jednak wymogi, jakie zarówno użytkownik, jak i współczesna psychologia pamięci, stawia tego typu narzędziom. Wprowadzenie większej ilości oznaczeń, mogłoby tylko utrudnić ocenę. Pomimo to, trzeba jednak stwierdzić, że pomiędzy testowanymi aplikacjami, brak jest wyraźnie najlepszej, a więc takiej, która mogłaby być powodem zaniechania tworzenia kolejnych podobnych rozwiązań. Na załączonym zestawieniu (Rys.3.6), w którym podsumowano w skali liczbowej, porównywane cechy czterech aplikacji, widać wyraźnie, jak duża pozostaje jeszcze przestrzeń do rozwoju dla tego typu systemów.



Rysunek 3.6: Sumaryczna ocena jakościowa występowania porównywanych cech w konkurencyjnych systemach

Podsumowując, żaden z porównanych programów, nie spełnia w zadowalającym stopniu wszystkich najważniejszych wymogów, stawianych takim narzędziom. Ani Fiszkoteka, Anki, Mnemosyne, czy SuperMemo, nie dają możliwości testowania odwrotnego słów, zgodnego z założeniami nauki języków obcych. Nie ma też wśród nich programu, który rozwiązałby problem kumulacji powtórek, będącego najczęstszym powodem rezygnacji uczniów z nauki przy pomocy komputera. Jeśli chodzi zaś o sposób przeprowadzenia fazy nauki nowego materiału, to najlepiej jest to rozwiązane w aplikacji Anki. Niemniej, przyjazność dla użytkownika tego konkretnego systemu, pozostawia wiele do życzenia i jest chyba najgorsza z przetestowanych. Z kolei inna, mająca duży wpływ na wyniki pracy, cecha, czyli sposób przeprowadzania samooceny przez ucznia, w większości programów,

jest nieadekwatna do rzeczywistości. Tutaj, jedynie Fiszkoteka, spełnia wymogi zarówno prostoty, jak i przejrzystości (zrozumienia przez użytkownika).

Rozwiążanie wymienionych mankamentów jest jak najbardziej możliwe. W kolejnym rozdziale przedstawię założenia niniejszego projektu, których głównym celem jest skupienie się na komforcie użytkownika i zapewnienie wysokiej skuteczności jego wysiłkom.

### 3.3 Założenia projektu w świetle konkurencyjnych rozwiązań

Po przeprowadzeniu tego rekonesansu wśród rozwiązań obecnych już na rynku, pozostało odpowiedzieć na pytanie: jaki ma zatem być kolejny system bazujący na kontrolowanym powtarzaniu uczonego materiału?

Pewności nie ulegał fakt, że porzucić trzeba niesprawdzającą się ideę organizowania materiału leksykalnego w postaci list, wielokrotnie w literaturze naukowej deprecjonowanej [19], [20], [21].

Autor, zdecydował się na ponadto, na przeprowadzenie praktycznego eksperymentu, w celu zbadania efektów uczenia przy użyciu wspomnianych czterech programów, oraz stosowanej dotychczas przez autora, metody list, co miało na celu wyznaczenie wielkości praktycznego wpływu poszczególnych cech na praktyczny wynik nauki w czasie.

Doświadczenie przeprowadzono w okresie od 13-go stycznia do 15-go marca 2021 roku. Zasada jego polegała na nauce w każdym z porównywanych programów, partii wyznaczonego zbioru kart słownikowych (budowa karty to pytanie: język polski - odpowiedź: język angielski) i testowaniu ich znajomości zgodnie z oferowanym przez konkretny program wzorcem. Każda partia testowa liczyła po 50 kart/rekordów, na które składały się w równym stosunku słowa pojedyncze i wyrażenia, wcześniej nieznane autorowi. Jeśli chodzi o porównawcze wyniki nauki z list, skorzystano z udokumentowanych wyników dotyczących 40-tu list, liczących każda po 25 słów, których nauka rozpoczęła się w 2020 roku.

Faza kodowania, czyli zapamiętywania, w każdej z czterech aplikacji, obejmowała jednorazowo od 10 do 15-tu haseł, podczas gdy zapamiętywanie każdej z list, odbywało się jednorazowo.

Z eksperymentu na jego początkowym etapie, usunięto program Mnemosyne, z uwagi na częste trudności w stosowaniu 6-cio stopniowej skali samooceny, oraz powodującą zmęczenie szatę graficzną programu. Zapis wyników poszczególnych powtórek, wyznaczanych przez każdy z testowanych systemów, odbywał się w arkuszu kalkulacyjnym Excel (Rys.3.7).

Wyniki zdecydowano się oceniać dwukrotnie:

Jako procent porażek stwierdzonych po pierwszej regularnej powtórce (po 24h lub 48h od zakończenia fazy kodowania)

Jako procent porażek po 5-tej powtórce, co jest bardziej miarodajne niż ocenianie po określonym czasie, z uwagi na dużą rozpiętość przerw pomiędzy powtórkami różnych słów

W tabeli (Rys.3.8), gdzie przedstawiono wyniki, autor wyszczególnił również cechy najbardziej różniące poszczególne metody, jako mogące mieć największy wpływ na

### 3. Od teorii naukowej do praktyki rynkowej

	A	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1				nauka		I powtórka		II powt.	lipx	III powt.		IV powt.			
2	beczeć, jeczeć	3 x	13-Jan		14-Jan	x		15-Jan		17-Jan		25-Jan	~	06-Feb	x
3	chodziłyśmy ze sobą		16-Jan	2	17-Jan		20-Jan		25-Jan		23-Feb	x	25-Feb		
4	czuły uściszk, przytulenie	4 x	18-Jan	2	19-Jan	x1	20-Jan	x1	21-Jan		25-Jan		31-Jan		
5	drobny, wiotki	1 x	18-Jan	2	19-Jan		23-Jan		27-Jan		23-Feb	x	25-Feb		
6	dzikus		18-Jan	2	19-Jan		23-Jan		31-Jan		23-Feb				
7	gderliwy, zrzędziawy	1 x	16-Jan	3	17-Jan		20-Jan		25-Jan		23-Feb	x	25-Feb		
8	godzina za godziną	3 x	18-Jan	2	19-Jan	x1	20-Jan		23-Jan		31-Jan	~	23-Feb	~	
9	jesteś obiąkany/szalony		13-Jan		14-Jan		16-Jan		23-Jan		23-Feb				
10	klucz do nakrętek	1 x	16-Jan	3	17-Jan	x1	18-Jan		21-Jan		27-Jan		13-Feb		
11	kończę pracę za dwie godziny	2 x	16-Jan	3	17-Jan	x1	18-Jan	x1	19-Jan		21-Jan		26-Jan		
12	krach finansowy, wzburzenie, furia (infor		13-Jan		14-Jan		16-Jan		23-Jan		23-Feb				
13	kretyn	2 x	13-Jan		14-Jan		16-Jan		23-Jan		13-Feb	x	23-Feb	x	
14	lubię takie jak jest		16-Jan	2	17-Jan		19-Jan		26-Jan		27-Jan		31-Jan		
15	negatywna strona		16-Jan	2	17-Jan		20-Jan		27-Jan		23-Feb				
16	nic z tego nie będzie	5 x	13-Jan		14-Jan		16-Jan		25-Jan	x1	26-Jan	x1	27-Jan		
17	nie pochwalać	1 x	13-Jan		14-Jan	x	15-Jan		18-Jan	x1	19-Jan		23-Jan		
18	nieodparcie, przemożenie	7 x	16-Jan	2	17-Jan	x1	18-Jan	x1	19-Jan		21-Jan	x1	23-Jan		
19	niewiarygodne, niepojęty	2 x	16-Jan	2	17-Jan	x1	18-Jan	x1	19-Jan		23-Jan		03-Feb		
20	nikczemny	1 x	13-Jan		14-Jan		16-Jan		25-Jan	x	06-Feb		10-Mar		
21	nucenie, śpiewanie pod nosem		16-Jan	2	17-Jan		20-Jan		25-Jan		13-Feb				
22	obalać, udowadniać błędność	3 x	16-Jan	3	17-Jan	x1	18-Jan		21-Jan		27-Jan		13-Feb	x	
23	oczarować, olśniewać	2 x	13-Jan		14-Jan	x	15-Jan		17-Jan		23-Jan	x1	25-Jan		
24	odór smród	2 x	13-Jan		14-Jan		16-Jan		23-Jan		23-Feb	x	25-Feb	x	
25	odwrócić uwagę od czegoś	2 x	16-Jan	1	17-Jan	x1	23-Jan		26-Jan		03-Feb		27-Feb	x	
26	otarcie, zniecierpliwienie, pojry	3 x	16-Jan	4	17-Jan	x1	18-Jan		21-Jan		31-Jan	x1	03-Feb	x	
27	otoczyć	2 x	13-Jan		14-Jan		17-Jan		25-Jan	~	06-Feb	x1	08-Feb		
28	palant, głupek		13-Jan		14-Jan		17-Jan		23-Jan		08-Feb				
29	para uprawia seks	1 x	18-Jan	2	19-Jan	x1	20-Jan		23-Jan		31-Jan		27-Feb		
30	plama, smuga	2 x	16-Jan	2	17-Jan	x1	18-Jan	x1	20-Jan		25-Jan		23-Feb		
31	ponosić pełną odpowiedzialno	1 x	16-Jan	3	17-Jan		20-Jan	x1	21-Jan		25-Jan		03-Feb		
32	posada		13-Jan		14-Jan		17-Jan		21-Jan		23-Feb				

Rysunek 3.7: Przykładowy wygląd arkusza wynikowego - Fiszkontakte

końcowy rezultat.

	Podsumowanie	Powtórka po 10'	Powtórka po 24h	Procent porażek po 1-szej powtórkę	Sumaryczny procent porażek po 5-ej powtórkę
<b>SuperMemo</b>	-	-	-	50%	70%
<b>Fiszkontakte</b>	-	-	+	40%	68%
<b>Anki</b>	-	+	-	30%	38%
<b>Lista</b>	+	+	+	15%	20%

Rysunek 3.8: Sumaryczne wyniki eksperymentu

Obserwowalne w przypadku SuperMemo i Fiszkontakte bardzo duża ilość niezapamiętyanych słów stwierdzana podczas pierwszej powtórki, pozostaje w kontraste z pozostałymi metodami. Nasuwa się wniosek, że obecność podsumowania fazy zapamiętywania, czyli wykonania ostatniej iteracji po całej partii uczonych słów, znacznie ogranicza ilość niepowodzeń. Odnosi się to również do wyników w późniejszym okresie nauki. Fakt, że najlepsze wyniki osiągnięto przy użyciu prymitywnej metody list, a dodatkowo objętość partii jednorazowo uczonych słów w tym przypadku była ponad trzykrotnie większa niż w pozostałych, sugeruje, że to nie zastosowane medium prezentacji, ale sposób przeprowadzenia nauki w ciągu pierwszych 24 godzin, ma największe znaczenie dla trwałości śladu pamięciowego.

Wniosek powyższy, pozostaje w zgodzie z doniesieniami literatury, podkreślającymi

kluczowe znaczenie takich składowych fazy kodowania, jak podsumowanie [22], a także przeprowadzenia pierwszego testowania już po kilkunastu minutach od zapamiętania słowa dla dalszego skutecznego testowania.

Wartą wprowadzenia do funkcjonalności systemu, jest także wg autora, możliwość zastosowania przeuczenia, z ang. *Overlearning*, który sam w sobie jest powtórzeniem prezentacji danej karty użytkownikowi tyle razy, dodatkowo po fakcie zapamiętania, ile razy było potrzebne, aby uczeń ją zapamiętał. Psychologowie wielokrotnie zwracali dotąd uwagę [23], [24], na znaczenie jakie przeuczenie może mieć dla trwałości śladu w pamięci. W projektowanym systemie, zdecydowano jednak, że po pierwsze, funkcjonalność ta będzie opcjonalna, po drugie zaś, idąc za radą prof. Marii Jagodzińskiej [6], ograniczyć wielkość przeuczenia do 50% ilości prezentacji, które były wymagane do zapamiętania hasła.

Na bazie doświadczeń zebranych w toku osobistej nauki w oparciu o technikę list, stosowaną w przeciągu ostatnich 2 lat, oraz powyższego eksperymentu, podjęto decyzje o wprowadzeniu do programu **Repeater**, kluczowych funkcjonalności, będących jednocześnie jego wyróżnikami spośród innych rozwiązań, z którymi spotkał się autor opracowania.

Pozostaje jeszcze na koniec kwestia kierunku testowania, czyli odpowiedź na pytanie, czy powtarzanie zapamiętanego hasła powinno odbywać się zgodnie z kierunkiem nauki, co by oznaczało, że uczeń posiadłby co prawda umiejętność przetłumaczenia słowa z języka ojczystego na obcy, ale już bez pewności odcyfrowania znaczenia napotkanego sowa obcego na język naturalny. Ta druga jakby część znajomości słowa, pozwala na rozumienie zarówno tekstu pisanych, ale też mowy obcej.

Jednym słowem, posiadanie umiejętności przetłumaczenia słowa ojczystego na obcy język nie gwarantuje, że będzie ono przez nas zrozumiane, gdy je np usłyszmy, lub przeczytamy. Ponieważ wprowadzenie możliwości testowania odwrotnego już na samym początku nauki, niepotrzebnie rozprasza ucznia, jak to zostało podkreślone przy opisie tej funkcji w programie Anki, jako jedynym ją posiadającym, zdecydowano o znaczącej modyfikacji w przypadku bieżącego projektu.

Wspomniana funkcja jest zastosowana dla kart, które osiągnęły zaawansowany stopień znajomości, co określone zostało przez osiągnięcie przez taką kartę odstępu pomiędzy powtórkami rzędu 28 dni.

Zatem, podsumowując, do tych unikalnych własności systemu, któremu poświęcony jest niniejszy projekt, możemy ostatecznie zaliczyć:

- Przeprowadzanie każdorazowo podsumowania zakończonej fazy zapamiętywania
- Umożliwienie użytkownikowi poszerzenie fazy zapamiętywania o przeuczenie
- Obligatoryjne przeprowadzanie pierwszego testowania już po kilkunastu minutach od zakończenia nauki
- Obligatoryjne wyznaczanie kolejnej powtórki po 24 godzinach od końca nauki
- Umożliwienie testowania kart słownikowych również w przeciwnym do początkowego, kierunku

# 4. Projekt

## 4.1 Algorytmy i logika pracy programu

Niniejszy rozdział poświęcony jest wyjaśnieniu działania logiki systemu, odpowiadającej za skuteczność wysiłków użytkownika włożonych w opanowanie uczonego materiału, oraz czynników wpływających bezpośrednio i pośrednio na jej działanie.

Wszelkie działania, jakim powinno być poddane słowo, w tym głównie określanie terminów jego powtórek, zależeć muszą od stopnia jego opanowania, czy raczej, bieżącej łatwości wydobycia go z pamięci. Ponieważ jest to sprawa bardzo indywidualna, nie istnieje metoda, która by to umożliwiała, bez przeprowadzenia rzeczywistego odpytania. Jednak sam fakt testowania, jest już formalną powtórką i wpływa na wynik kolejnych oznaczeń dla danego słowa, a poza tym, uzyskanie oznaczenia dla jednego hasła, nie umożliwia przewidzenia wyników dla innych. Autor pracy, zdecydował się zatem, wykorzystać do tego wartości, które dają się oznaczyć już w trakcie wstępного zapamiętywania hasła.

Wyizolowanie tych czynników, pozwoliło podzielić materiał uczony na 3 grupy, w zależności od stopnia trudności w opanowaniu pamięciowym: słowa łatwe zarówno do zapamiętania i odtworzenia, o normalnej (średniej) trudności, oraz materiał trudny. Każda z tych grup jest w aplikacji prowadzona wg odmiennego algorytmu obliczania odstępów pomiędzy powtórkami, co ma zapewnić jak najmniejszą ilość niepowodzeń przy przypominaniu, a jednocześnie zoptymalizować długość odstępów pomiędzy kolejnymi powtórkami.

Podział został dokonany na podstawie kryteriów, opracowanych w wyniku przeprowadzonego samodzielnie eksperymentu porównawczego trzech rynkowych systemów oraz doświadczeń własnych z pracą z listami par słów. Szczegółowe opracowanie dotyczące wspomnianego eksperymentu, zostało umieszczone w rozdziale poświęconym rozwiązaniom obecnym na rynku. Przypomnieć warto jednak tutaj główne wnioski z niego wynikające, a mianowicie wpływ sposobu przeprowadzenia etapu nauki na stopień zapamiętania materiału. Jak wspomniano w końcowych wnioskach eksperymentu, jednoczesne zubożenie procesu kodowania o końcowe podsumowanie, sprawdzenie po 10 minutach, oraz przeprowadzenie pierwszej powtórki dopiero po upływie 48 godzin (SuperMemo), skutkowało ponad 50-cio procentowym prawdopodobieństwem porażki w pierwszej powtórce. Z kolei uzupełnienie nauki o przeprowadzenie pierwszej powtórki po 24 godzinach od zakodowania informacji (Fiszkontakte), zmniejszyło to ryzyko do 40%.

Najlepsze wyniki osiągnięto jednak w przypadku programu Anki, który, co prawda nie zapewniał podsumowania zapamiętywania, ani powtórki już po 24 godzinach, ale pozwalał przeprowadzać testowanie po 10-ciu minutach od skutecznego zakodowania informacji,

co pozwoliło zmniejszyć ryzyko niepowodzenia w pierwszej powtórce, do 30%.

Dla uzupełnienia, trzeba dodać, że praca z listami słów, gdzie w toku procesu nauki przeprowadzane było zarówno podsumowanie, testowanie po 10-ciu minutach, jak i powtórka już po 24 godzinach, pozwalało na powtarzalne zmniejszenie ryzyka niepowodzeń w ciągu 2 pierwszych powtórek do 15%, a w toku 5-ciu kolejnych, nie przekroczyło 20%. Poza tym, w rozdziale poświęconym temu eksperymentowi, znajduje się również podsumowanie wpływu fazy nauki na stopień retencji materiału słownikowego po upływie ponad 2 miesięcy od momentu zakodowania.

#### 4.1.1 Praca z kartą w fazie nauki

Skoro zatem, faza nauki ma sama w sobie tak duże znaczenie dla trwałości śladu pamięciowego, właśnie w jej przebiegu, należy upatrywać kryteriów oceny stopnia trudności słowa.

Przyjęto więc, że daną kartę/fiszkę można zaliczyć do łatwo zapamiętywalnych, gdy spełnione są wszystkie poniższe warunki jednocześnie:

- przeprowadzono podsumowanie
- przeprowadzono testowanie po 10 minutach
- przeprowadzono powtórkę po 24 godzinach
- na żadnym z tych etapów nauki, nie miało miejsca niepowodzenie

Dodatkowym warunkiem jest zapamiętanie słowa już po jednokrotnej prezentacji go użytkownikowi, zakładając, że liczba uczonych słów w partii, przekraczała 5 kart, co uzasadnia [przypis], powołując się na badania, wskazujące, że człowiek jest w stanie przechować w pamięci migawkowej do 7 – 9 pojedynczych wyrazów i być w stanie je odtworzyć w czasie kilkunastu sekund do kilku minut po fakcie prezentacji.

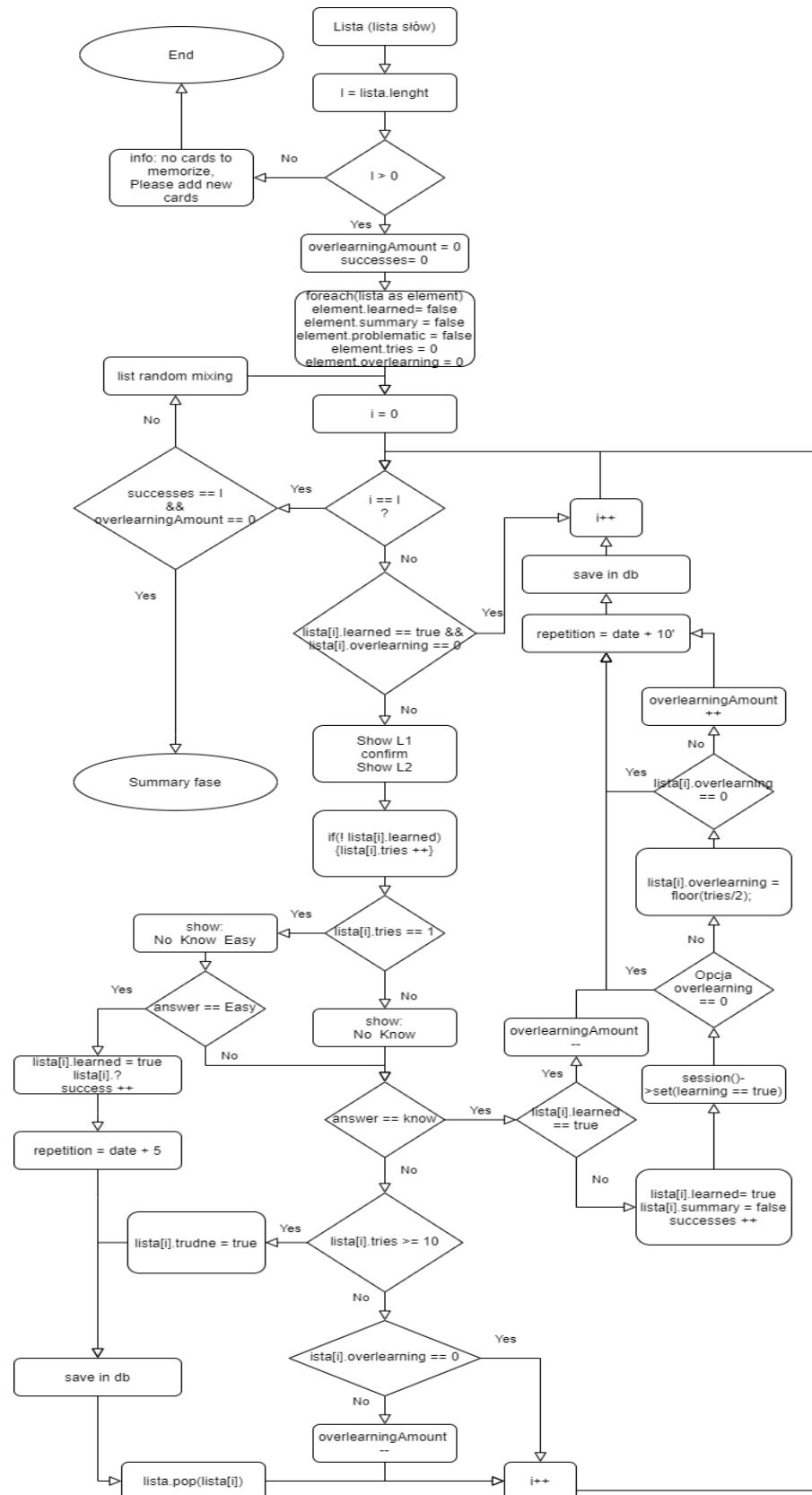
Z kolei, słowo staje się trudne, gdy albo nie przeprowadzono któregokolwiek z wyżej wymienionych etapów, albo pojawiło się w ich trakcie chociaż jedno niepowodzenie, albo też ilość prezentacji koniecznych do zapamiętania karty, przekroczyła 10, przy założeniu, że partia uczonego materiału to minimum 5, a maksymalnie 20 kart/fiszek.

Do słów o normalnym stopniu trudności, zaliczono więc ostatecznie takie, które w toku pełnej fazy nauki nie zanotowały niepowodzenia, a wysiłek włożony w zapamiętanie, był niewielki.

Jak jednak go wyznaczyć, skoro jest to sprawa wybitnie indywidualna? Wystarczającym wyznacznikiem tego, może być ilość prezentacji każdej karty, jaka była konieczna do uzyskania od użytkownika potwierdzenia jej zapamiętania, przy czym najmniejsza możliwa wielkość, to 1, co odpowiadać może słowi bardziej łatwemu do zapamiętania, być może dzięki istniejącym skojarzeniom z posiadanymi przez ucznia informacjami, bądź też dzięki zastosowaniu przez niego pewnych mnemotechnik.

Mając tak zdefiniowane wymagania, pozostawało jedynie opracowanie schematu postępowania, algorytmu programowego, który sprostałby im, a jednocześnie odpowiadałby za całą logikę interakcji pomiędzy uczniem i uczyonym materiałem.

Stopień złożoności problemu nauki pamięciowej, wykluczał jednak zastosowanie jednego wspólnego algorytmu pracy programu z kartą słownikową, obejmującego wszystkie jej fazy. Z tego powodu, opracowano na potrzeby niniejszej pracy, osobne sekwencje działań w przypadku kodowania informacji w pamięci i przeprowadzania rozłożonych w czasie powtórek.



Rysunek 4.1: Algorytm fazy nauki nowych kart

Logika, na której bazować powinien algorytm etapu zapamiętywania, musiała brać pod uwagę zróżnicowanie szybkości opanowywania konkretnych informacji, a także możliwość zaistnienia porażek na każdym etapie kontaktu użytkownika z testowaną kartą. Niemożność wykluczenia nawet najmniej prawdopodobnego scenariusza, który mógłby zaistnieć na styku użytkownik – karta, spowodowała, że właśnie schemat przeprowadzenia fazy nauki, stał się najbardziej skomplikowanym z zastosowanych w badanym systemie, algorytmów. Poniżej został przedstawiony omawiany algorytm. W celu lepszej przejrzystości, został on rozdzielony na etap kodowania (Rys.4.1), oraz podsumowania (Rys.4.2) i 10-cio minutowej powtórki, przedstawione na kolejnych rysunkach.



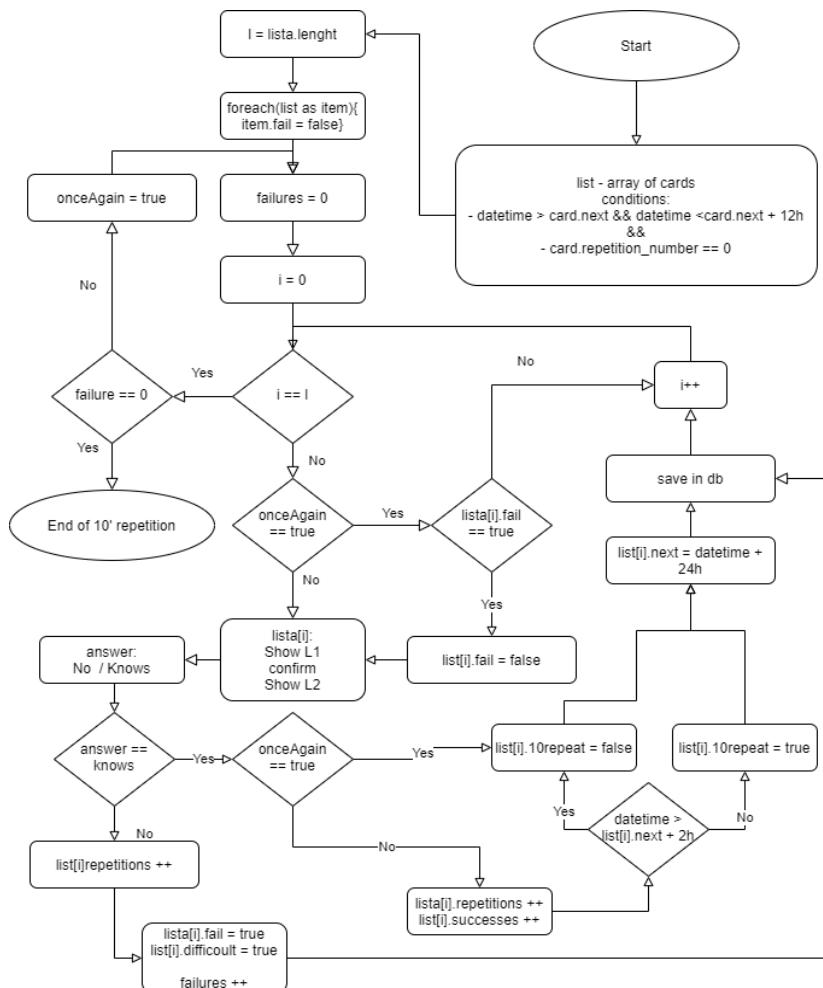
Rysunek 4.2: Algorytm podsumowania nauki nowych kart

Już na samym początku pracy z programem, kiedy to dochodzi do kontaktu ucznia z hasłem, wprowadzona została możliwość zidentyfikowania słów, które użytkownik już dobrze zna i wyznaczenie im od razu odległych w czasie powtórek, z pominieniem prezentacji, przeuczenia, podsumowania, czy nawet testu po 24 godzinach od nauki.

Pozostałe karty podlegają już wtedy typowej, pełnej procedurze, która dzięki wspomnianemu zabiegowi, przebiega szybciej i nie irytuje użytkownika, mogącego dzięki temu łatwo uzupełnić zasoby programu o karty, stosowane przez niego wcześniej w innych sys-

temach i nie tracić czasu na ich zbędną, pełną „naukę”. W wyniku samej fazy prezentacji, uzyskiwane są od użytkownika dane, odzwierciedlające jego interakcję z uczonym materiałem. Co warto zauważyć, rejestrówka jest przy tym także wielkość wysiłku, jaki został włożony we wstępne zapamiętanie karty, poprzez zliczanie ilości wymaganych do tego, prezentacji materiału.

Algorytm nauki (Rys.4.1), przewiduje również włączenie w tok nauki, opcji przeuczenia, dostosowując jej przebieg do łatwości zapamiętania słowa. Zrealizowana jest ona tutaj, jako cykl ponawianych prezentacji, jedynie tych słów, których zapamiętanie wymagało dużego wysiłku.



Rysunek 4.3: Algorytm 10-cio minutowej powtórki

Możliwość przerwania nauki i związane z tym faktem ryzyko utraty uzyskanych przez program informacji o jej przebiegu, zostało zażegnane poprzez natychmiastowe, asynchroniczne zapisywanie uzyskanych od użytkownika danych, w bazie MySQL. Fakt zaistnienia, lub nie, danego działania, jak na przykład podsumowania (Rys.4.2), czy powtórki po 10-ciu minutach (Rys.4.3), również są rejestrowane jako osobne właściwości karty, w tabeli wspomnianej bazy danych.

Istotne jest także zastosowana w powyższym algorytmie zmiana kolejności prezen-

wanych do nauki kart, za każdym razem, gdy zachodzi potrzeba ponownego przedstawienia uczonego materiału użytkownikowi. Pozwala to na uniknięcie tzw efektu sąsiedztwa [6], zgodnie z którym, powtarzanie materiału w tej samej kolejności, ułatwia jego zapamiętanie, z uwagi na możliwość skojarzenia danego słowa z innymi, występującymi tuż przed i tuż po nim. Odtworzenie z pamięci tak zakodowanego hasła będzie z oczywistych względów utrudnione, o ile nie będzie się odbywało w tej samej kolejności słów, jak podczas procesu zapamiętywania.

#### 4.1.2 Rozłożone w czasie powtórki

Nauka, pomimo złożoności przedstawionego algorytmu, jest jednak dość przewidywalnym etapem w pracy z kartą słownikową, co pozwala na wprowadzenie stałych warunków i wielkości, pomagających korygować pojawiające się trudności. Natomiast, niepowodzenia powstające w toku rozłożonych w czasie powtórek, nie są już tak łatwo i szybko identyfikowane, a sam sposób reagowania na nie, powinien uwzględnić nowe czynniki, ujawniające swój wpływ w miarę upływu czasu.

Porażka na etapie okresowych powtórek wymaga więc, na bieżąco korekty stosowanego sposobu obliczania kolejnych przerw, gdyż sama w sobie jest sygnałem na obecność okoliczności lub zmiennych nie ujętych w pierwotnych założeniach, lub takich, które nie dały znać o swoim istnieniu w czasie fazy nauki, a powodujących zaburzenia toku pracy z hasłem.

Co więcej, nie są tu już zbyt pomocne dane wyciągnięte z procesu kodowania, a dodatkowym utrudnieniem staje się brak pośredniego stopnia oceny znajomości słowa, pomiędzy „nie wiem” i „pamiętam”, który mógłby być w takich wypadkach wczesnym wskaźnikiem narastających trudności, dotyczących retencji konkretnej karty. Ponieważ jednak autor dostrzegł więcej korzyści w rezygnacji z pośrednich stopni autooceny przez użytkownika, należało dostosować dalszy tok działania do dostępnych źródeł danych.

Ważne było zatem przyjęcie nowego sposobu postępowania w przypadkach trudności z wydobyciem informacji z pamięci, co umożliwiły zapewnienie, wspomnianej na początku niniejszej pracy, wymaganej elastyczności algorytmu aplikacji.

Na niepowodzenie w odzyskaniu danych z pamięci, porównywane programy rynkowe, reagują wszystkie bardzo podobnie: przeprowadzana jest ponownie faza nauki, w której biorą udział karty, które odpadły w bieżącej powtórce. Słowa takie, następnie, podlegają standardowemu obliczaniu kolejnych powtórek od stanu zerowego. Jest to model bezpieczny, ale opóźniający proces nauki, bo nie uwzględniający już dokonanych postępów. Poza tym, jak wynika z doświadczeń H. Ebbinghausa, przypomnienie nawet zapomnianej informacji, wymaga znacznie mniej pracy niż pierwotnie i pozwala na szybkie ponowne osiągnięcie poprzednich długości odstępów pomiędzy kolejnymi powtórkami.

Sposobem, który nasuwa się na myśl jako całkowite przeciwieństwo powyższego, jest dopuszczenie takiego hasła do kontynuowania dotychczasowego trendu w przyroście czasu przerw, po powtórzeniu dla niego fazy nauki. Stwarza to jednak duże ryzyko szybkiego, ponownego zaistnienia niepowodzenia danego słowa, zwłaszcza, że zdarzyło się ono już przy krótszym odstępie między testami. Autor pracy zdecydował się na zachowanie w jak największym stopniu osiągniętych postępów w nauce danej karty, przy ograniczeniu ryzyka ponownej porażki. Wziął również pod uwagę klasyfikację trudności hasła, jaka dokonywana jest po fazie nauki, a która w największym stopniu rzutuje na zachowanie się

karty w przyszłości.

Algorytmem obiecującym spełnienie powyższych założeń, jest jedynie ten , w którym sposób postępowania w niepowodzeniu, zależy od całości zebranych danych, zarówno tych w toku kodowania, jak i z dotychczas przeprowadzonych powtórek. Zwłaszcza ostatnie z wymienionych, źródło danych, wiele potrafi powiedzieć na temat rozwoju znajomości karty w czasie. Funkcję tę wspiera wyliczany na bieżąco stosunek ilości skutecznych testów do ich całkowitej liczby. Jest on odzwierciedleniem zmiany stopnia trudności słowa w toku powtórek. Wszystkim powyższym założeniom, odpowiada diagram, obrazujący działanie opisanego algorytmu powtórek rozłożonych w czasie (Rys.4.4).



powtÃšrka\_gotowe.png

Rysunek 4.4: Algorytm rozłożonych w czasie powtórek

Kluczowym, w przedstawionym schemacie, jest podejście do słów, uznanych jako problemowe (trudne) już na etapie nauki. Każde niepowodzenie w odzyskaniu takiego hasła z pamięci w trakcie którejkolwiek z powtórek, skutkuje wyzerowaniem jego postępów i skierowaniem go do ponownej nauki w grupie całkowicie nowego materiału. Aby karta taka miała możliwość wzięcia udziału w najbliższej edycji nauki, co chroni przed dalszym osłabieniem jej śladu pamięciowego, otrzymuje najwyższy priorytet wyboru. Wartość ta jest osobną właściwością każdej karty w tabeli w bazie danych. Dodatkowo, zmniejszony zostaje jej współczynnik zwielokrotnienia poprzedniego interwału powtórkowego, co ma w przyszłości zmniejszyć ryzyko jej zapomnienia, poprzez skrócenie przerw pomiędzy odpitywaniami.

#### **4.1.3 Obliczanie odstępów pomiędzy powtórkami**

Ponieważ został tu już poniekąd poruszony problem wyliczania czasu powtórek, warto przyjrzeć się metodzie, która została opracowana na potrzeby niniejszego projektu, a podlega jej działaniu każda testowana w czasie powtórki karta.

Konieczność wyliczania terminów kolejnych terminów przypomnień, pojawiła się już z chwilą, gdy sam H Ebbinghaus [26], odkrył prawidłowości rządzace pamięcią. Wg jego badań, łatwość odzyskania z pamięci zapamiętanej informacji, spada wykładniczo wraz z upływem czasu, co ujęto w postaci wzoru [27]:

$$p = 2^{-\Delta h}$$

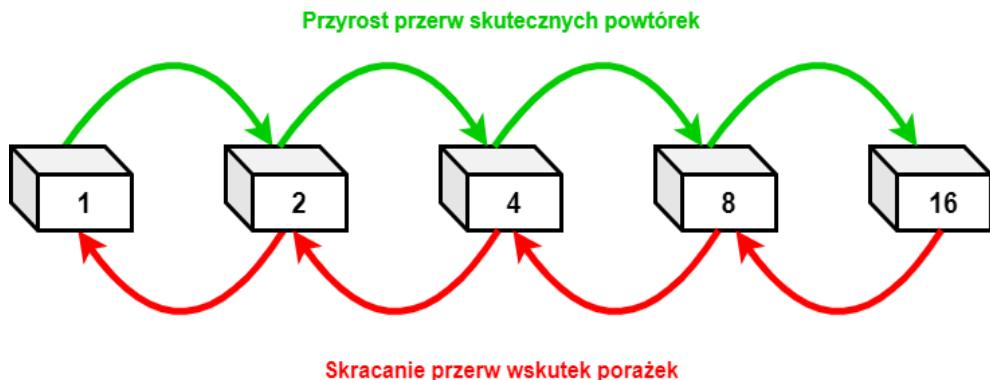
W powyższym równaniu,  $p$  oznacza prawdopodobieństwo sukcesu w przypomnieniu sobie np: słowa. Prawdopodobieństwo to jest funkcją  $\Delta$  - czasu, jaki upłynął od ostatniej powtórki i  $h$ , okresu półtrwania, lub raczej bieżącej siły śladu pamięciowego.

Współcześnie, daje się obserwować liczne modyfikacje tego podstawowego założenia. Potwierdzonym praktycznym użyciem jest metoda Primsleura (1967) [25], opracowana do nauki ze źródeł audio, opierająca wyliczanie odstępów za pomocą zwielokrotniania ich za pomocą stałego mnożnika 5, co dawało ciąg powtórek po: 5 sekundach, 25 sekundach, 2 minutach, 10 minutach, 1 godzinie, 5 godzinach, 1 dniu, 5 dniach, 25 dniach, 4 miesiącach i ostatecznie po 2 latach. Model ten nie uwzględniał jednak indywidualizacji w podejściu do poszczególnych słów, zakładając ich jednakowy stopień trudności, co oczywiście jest niezgodne z rzeczywistością.

Kolejne podejście, dokonane przez Leitnera w 1972 [28], dotyczyło już nauki pamięciowej za pomocą ułożonych w osobnych pudełkach, kartonowych fiszek z hasłami. Zastosowana przez niego metoda, była bardziej elastyczna, pozwalała bowiem, w wypadku niepowodzenia na cofnięcie karty do poprzedniego pudełka, określonego krótszym interwałem powtórkowym, jak na rysunku (Rys.4.5).

Stosowane przez Leitnera mnożniki kolejnych odstępów czasowych, oscylowały wokół liczby 2, co w świetle późniejszych badań, oraz doświadczeń Woźniaka [29], pozwala na uzyskanie przerw, lepiej odpowiadających średniemu stopniowi retencji informacji w pamięci długotrwałej człowieka.

Oba przytoczone przykłady, pomimo starań ich twórców, miały wspólne ograniczenia, wynikające z nakładu pracy, potrzebnego do ich „implementowania” w analogowej rzeczywistości tamtych lat, co powodowało, że stopniem złożoności, nie mogły wykroczyć poza proste funkcje, więc nie mogły uwzględniać wielu istotnych zmiennych.



Rysunek 4.5: Korygowanie niepowodzeń w powtórkach wg systemu Leitnera [28]

Przezwyciężyć te niedogodności, pozwoliło dopiero rozpowszechnienie komputerów. W późniejszych latach mogły dzięki temu powstawać algorytmy, które byłyby w stanie dostosowywać się do użytkownika i materiału z którym on pracuje.

W tej materii, od przeszło dwudziestu pięciu lat, ma wiele osiągnięć polski naukowiec, Piotr Woźniak, twórca programu SuperMemo. Opracowywane przez niego kolejne algorytmy są dostępne open-source. Jedna z jego wczesnych metod, SM-2 z 1994 roku, znalazła chociażby, zastosowanie w konkurencyjnej aplikacji Anki. Sam twórca SuperMemo, zapewniał wtedy [29], że jego sposób obliczania przerw, zapewnia utrzymanie w pamięci długotrwałej 92% uczonych informacji.

Obecne, nowe edycje SuperMemo, bazują już na modyfikacji SM-18, co samo w sobie, nieco dewaluuje zapewnienia składane w 1994 roku, o poziomie skuteczności ówczesnej wersji systemu. Niestety, pomimo udoskonaleń, wzbogacenia algorytmu liczącego o całe matryce danych, czy uwzględnienia subtelnych zmiennych, nie ma nadal dobrego sposobu obliczania odstępów pomiędzy powtórkami. Stoi za tym, wg autora niniejszej pracy, sama natura pamięci, a szczególnie nasz niewielki stan wiedzy o niej, który nie daje żadnych podstaw do traktowania zapamiętywania, jako funkcji matematycznej, a więc przewidywalnej w swym przebiegu i wynikach.

Jak wynika z powyższego, nie udało się dotychczas nikomu opracować systemu, w pełni chroniącego użytkownika przed błędami wynikającymi z zapominania. Pewnym pocieszeniem może być jednak to, że w psychologii istnieje poważna hipoteza, sugerująca pozytywny wreszcie wpływ popełniania błędów na sam proces uczenia[30].

Pozostawało zatem w przedstawionej sytuacji, dostosować projektowany program do nieprzewidzianej zmienności wpływających na zapamiętywanie czynników i zapewnić systemowi mu elastyczność, możliwość samodzielnego wprowadzania modyfikacji, na bazie napływających do niego od strony użytkownika, danych.

Do informacji takich, magazynowanych w bazie danych, jako przypisane pojedynczemu rekordowi karty, a pozwalających na obliczanie terminów jej kolejnych testów, należą:

1. data odbytej fazy nauki
2. łączna ilość słów uczonych w tej edycji nauki
3. mediana liczby prezentacji w tej edycji nauki
4. liczba prezentacji, wymaganych do zapamiętania hasła
5. odbycie fazy podsumowania

6. odbycie fazy przeuczenia
7. odbycie powtórki po upływie 10 - 120 minut
8. odbycie powtórki po 24 godzinach
9. data ostatniej powtórki - dla obliczenia bieżącego odstępu
10. ocena znajomości hasła w czasie powtórki
11. identyfikator trudności hasła
12. ogólna liczba powtórek
13. liczba powtórek, zakończonych sukcesem

Wszystkie powyższe dane, tworzą pewnego rodzaju historię hasła, która pozwala na lepsze dopasowanie działania aplikacji do indywidualnych cech użytkownika.

Podstawę do obliczenia ilości dni, dzielących bieżącą powtórkę od następnej, stanowi jednak wzór, oparty na wspomnianym algorytmie SM-2, opracowanym przez dr Piotra Woźniaka. Jego oryginalna postać, przedstawia się następująco:

dla  $n = 1 I(1) = 1$

dla  $n = 2 I(2) = 6$

dla  $n > 2:$

$$I(n) = I(n - 1) \times EF$$

przy czym:

$$EF = EF' + (0.1 - (5 - q) \times (0.08 + (5 - q) \times 0.02))$$

gdzie:

$n$  - numer powtórki

$I(n)$  - odstęp między powtórką  $n$  i  $n-1$ , w dniach

$EF$  - nowy współczynnik łatwości (mnożnik przerw) *easiness factor*

$EF'$  - dotychczasowy współczynnik łatwości

$q$  - wartość oceny znajomości słowa w 5-cio stopniowej skali

Niemalże jednak stało się wykorzystanie przytoczonego wzoru bez modyfikacji. Powyższy algorytm został opracowany z myślą o 5-cio stopniowej skali oceny znajomości słowa, co nie ma zastosowania w niniejszym projekcie, opartym o skalę dwustopniową. Wprowadzono także, wspomniane wcześniej, a niestosowane w konkurencyjnych aplikacjach, składowe fazy nauki: podsumowanie, przeuczenie i powtórkę po 10-ciu minutach. Obligatoryjna jest również kolejna powtórka po z góry ustalonym czasie, 24 godzin. Poza tym, zdecydowano, aby wyznaczanie mnożnika  $EF$ , odbywało się nie za pomocą podobnego do powyższego, wzoru, ale w wyniku analizy historii słowa.

Wykorzystano tylko ogólną postać równania, wg którego data następnego testowania jest wynikiem przemnożenia długości ostatniego odstępu czasowego przez stałą, zwaną tutaj *mnożnikiem przerw*. Jako wyjściowy, bo wynikający z oznaczeń dokonanych na etapie nauki, mnożnik przerw dla każdego oznaczonego poziomu trudności słowa, przyjęto następujące początkowe wartości:

- 1,5 dla haseł oznaczonych jako trudne
- 2,0 dla haseł o normalnym stopniu trudności
- 2,5 dla haseł oznaczonych jako łatwe

W toku przeprowadzania kolejnych powtórek, zarówno klasyfikacja stopnia trudności karty jak i sam przelicznik (mnożnik), ulegają zmianom. Karta podlega kategoryzacji po każdorazowym ponownym rozpoczęciu fazy nauki, w wyniku pojawiających się niepowodzeń - ich ilość wymagana do wyzerowania postępów słowa, jest zależna od wstępnej klasyfikacji. Przelicznik przerw (mnożnik), z kolei, jest wtedy zmniejszany.

Ulega natomiast powiększeniu zawsze, gdy stosunek skutecznych powtórek do ogólnej ich liczby wzrośnie w wyniku kolejnego przeprowadzonego testowania. Zakres wartości granicznych dla tego współczynnika, zamyka się w przedziale {1.3, 2.8}. Szczegółowe warunki, jakim podlega ten proces, zostały przedstawione na wcześniejszych diagramach, szczególnie dotyczącym fazy nauki (Rys.4.1) i powtórki (Rys.4.4).

## 4.2 Baza danych

## 4.3 Bezpieczeństwo

### 4.3.1 Zabezpieczenie procesu rejestracji i logowania

#### Rejestracja konta w systemie

Kontakt użytkownika z programem zaczyna się w momencie ujrzenia ekranu powitalnego, na którym dokonać musi on wyboru, co do pierwszego, najważniejszego kroku. Jeśli jest już zarejestrowanym i aktywowanym użytkownikiem, powinien wybrać opcję *Logowanie*. Jeśli zaś, nie miał do tej pory styczności z aplikacją, koniecznie musi założyć w niej swoje konto. Aktywowanie tego przycisku, otwiera przed nim okno z formularzem rejestracyjnym, gdzie proszony jest o podanie danych, służących obsługującemu bazę danych na serwerze oprogramowaniu, na trwałe zapisanie go jako nowego rekordu tabeli *Users*.

Dane, jakich podanie jest konieczne to:

- adres e-mail, pełniący później rolę loginu
- zaproponowane przez siebie hasło do przyszłego konta
- ponowne wprowadzenie podanego wyżej hasła

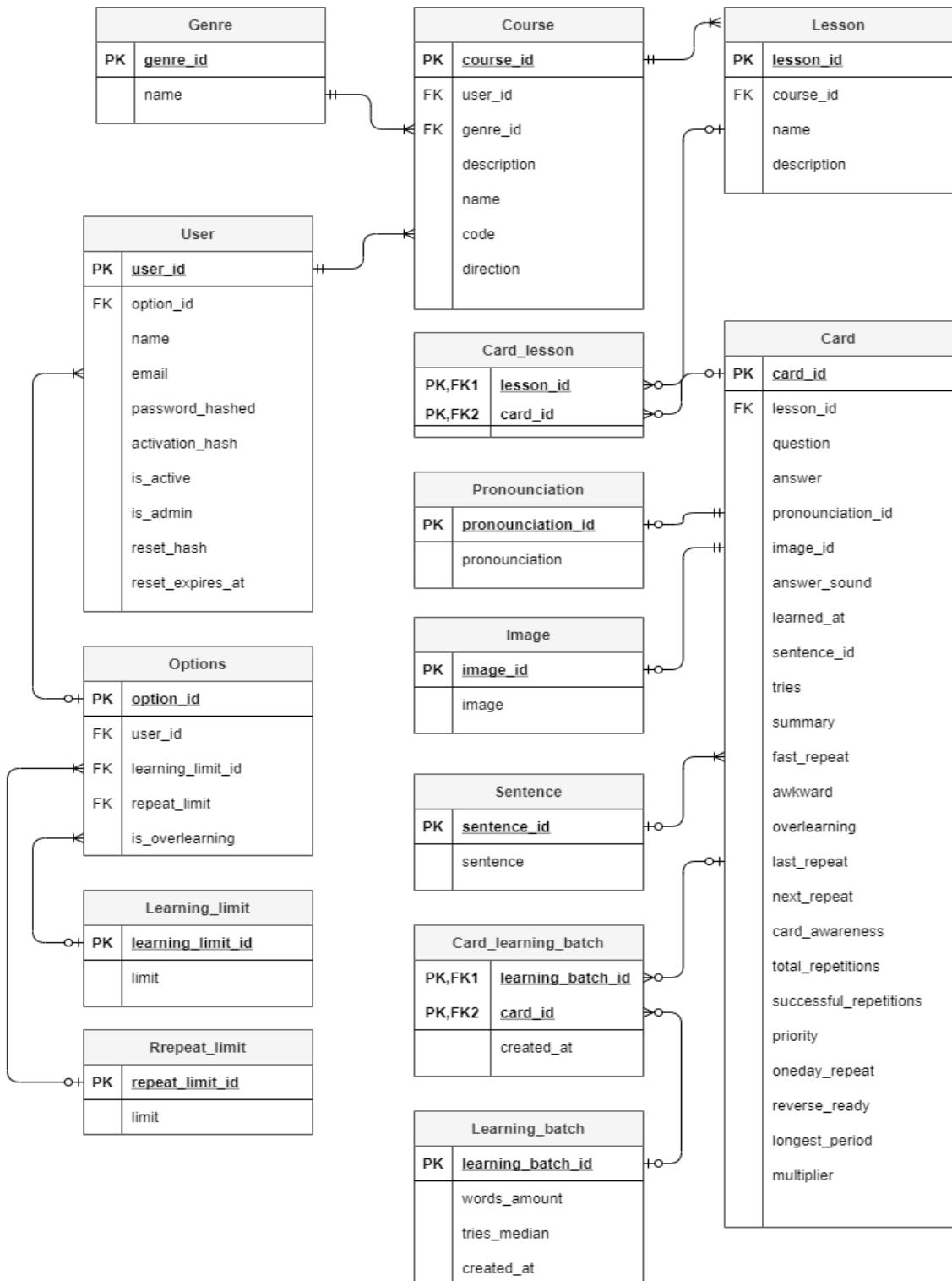
Proces walidacji, czyli sprawdzenia poprawności wprowadzonych danych, rozpoczyna się jeszcze po stronie klienckiej aplikacji. Ocenie poddany jest format wprowadzonego adresu e-mail, który powinien:

- zawierać znak @, przy czym nie jest on ani pierwszym, ani ostatnim znakiem adresu i musi być dokładnie jeden
- nie posiadać szeregu znaków specjalnych: : ; / < > ' oraz ?
- nie posiadać spacji
- nie posiadać polskich znaków diakrytycznych (z ogonkami)
- posiadać przynajmniej jeden znak kropki po znaku @

Wprowadzone dane do pola e-mail, najpierw są pozbawione ewentualnych początkowych i końcowych spacji, przez zastosowanie metody *trim()*. Dopiero tak przygotowany ciąg znaków poddawany jest ocenie zgodności z wzorcem, którym w naszym przypadku jest wyrażenie regularne, przedstawione poniżej:

```
/^[\w\d.-]+[a-z\d]*@[?:[a-z\d]+[a-z\d-]+\.\.][a-z]{2,5}\$/i
```

Zgodnie z powyższym wyrażeniem *regexp*, przed znakiem @ dopuszcza się użycie cyfr i liter - zarówno małych, jak i dużych (umożliwia to końcowa litera "i" w wyrażeniu). Podobnie jest z nazwą domeny, przy czym nie wprowadzano ograniczenia ilości znaków



Rysunek 4.6: Diagram związków encji bazy danych (Entity Relationship Diagram)

ją tworzących, a jedynie ograniczono do sześciu, długość jej końcówki, zakładając, że dłuższe nie występują i mogą być wynikiem błędu wpisującego.

Wprowadzany adres e-mail, podlega jeszcze sprawdzeniu pod kątem zgodności z jesz-

ce jednym, bardzo ważnym kryterium: musi być unikalny, nie może istnieć w bazie danych drugi identyczny ciąg znaków go opisujący. Dlatego też, zatwierdzające kliknięcie przycisku *Submit* w formularzu, uruchamia też wysłanie zapytania do bazy danych w celu stwierdzenia, czy nie zachodzi wspomniany przypadek.

Pole wprowadzania własnego hasła jest zduplikowane, aby można było przez porównanie obu pól, wyszukać wszelkie niezgodności, świadczące o popełnieniu pomyłki przy wpisywaniu

W przypadku pojawiienia się jakichkolwiek błędów, użytkownik jest o tym informowany, poprzez wyświetlenie pod oknem formularza, którego dotyczy, uwagi z krótkim wyjaśnieniem, co powinien poprawić, aby formularz mógł być zaakceptowany przez system. Pozytywne przejście walidacji skutkuje przesłaniem wprowadzonych danych do serwera, zaś użytkownik jest przekierowywany do podstrony z informacją o zatwierdzeniu formularza i konieczności oczekiwania na otrzymanie wiadomości e-mail z linkiem aktywacyjnym.

Skoro dane formularza trafiają do serwera, można zakładać, że powinny być już z pewnością sprawdzone. Autor postanowił jednak zastosować na wszelki wypadek, drugi stopień walidacji, dostarczanej przez klasę *Model*, frameworka Codigniter. Walidacja na tym etapie ma na celu ponowne sprawdzenie istnienia wartości przypisanych do zmiennych odpowiadających za poszczególne pola formularza i ich wymaganej i dopuszczalnej długości. Pod względem szczegółowości, jednak najważniejsze pozostaje i tak sprawdzenie po stronie aplikacji.

Dane na tym etapie mogą zostać wprowadzone do bazy danych, jako nowy rekord w tabeli *Users*. O ile zawartość zmiennej *\$email* nie podlega przed zapisem żadnym zmianom, o tyle wpisanie do bazy treści hasła użytkownika w czystej postaci, jaką wprowadził do formularza rejestracyjnego, byłoby lekkomyślne, gdyż narażone jest, w przypadku przejęcia, na łatwe wykorzystanie.

Zamiast tego zdecydowanie lepiej jest przechowywać hasła użytkowników w postaci zaszyfrowanej, jako hash, czyli wynik działania funkcji skrótu na oryginalnym ciągu znaków.

Klasa *Model* frameworka, służy i tutaj pomocą, zapewniając obsługę zdarzeń mających zajść przed faktem zapisu do bazy danych, do których w tym wypadku należy i nasze działanie. W metodzie *before()*, ze wspomnianej klasy, wywoływana jest funkcja tworząca hash oryginalnego hasła, *hash\_password()*, przyjmująca jako pierwszy argument, oryginalny ciąg znaków, a jako drugi, wykorzystującą, algorytm szyfrujący. Autor skorzystał tu z domyślnego algorytmu *PASSWORD\_DEFAULT*, zalecanego w dokumentacji języka PHP. Tak przygotowane "hasło", może teraz bezpiecznie być przechowywane w bazie danych.

Wyprzedzając nieco kolejność opisu, późniejsze logowanie, jest procesem podobnym do opisanego, w kontekście operacji na haśle. Podane podczas logowania hasło, po przesłaniu na serwer, zostaje zamienione funkcją skrótu na hash, tak samo jak powyżej, i dopiero taka postać podanego hasła jest porównywana z zawartością pola bazy danych, w rekordzie użytkownika. Jeśli zaś chodzi o podawany przy logowaniu adres e-mail, pełniący rolę loginu, z racji niepowtarzalności, to służy on jedynie do wyszukania tego rekordu w tabeli *Users*.

### Aktywacja konta użytkownika

Jednak przed pozwoleniem nowemu użytkownikowi na zalogowanie się nowym loginem i hasłem, weryfikacji podlega prawdziwość podanego przez niego adresu skrzynki mailowej. Dopiero po tym, pole rekordu użytkownika o nazwie *activated*, uzyskuje wartość *true*, zezwalającą na korzystanie z konta.

Odbiera się to za pośrednictwem wysyłanej po udanej rejestracji, wiadomości elektronicznej na podany w formularzu rejestracyjnym, adres. Zawartość tej wiadomości stanowi informacja o potrzebie aktywacji nowego konta poprzez uruchomienie załączonego linku.

Aby jednak móc rozpoznać, który użytkownik akurat odpowiedział na link aktywacyjny, adres Url linku, wzbogacony jest o indywidualny kod, tzw token aktywacyjny.

Powstaje on jednocześnie z zapisaniem w nowym rekordzie, danych rejestracyjnych i wprowadzony jest do osobnego pola tego rekordu.

Wracając do sposobu tworzenia wspomnianego kodu aktywacyjnego, ponieważ odmiennie od hasła, nie pochodzi on od użytkownika, trzeba go wygenerować po stronie serwera. Powinien być możliwie losowy, więc aby tak było, zastosowana jest funkcja PHP, *random\_bytes()*, jako argument, przyjmująca długość oczekiwanej ciągu.

Niestety, ze względów bezpieczeństwa, kodu/tokena w takiej postaci nie można przechowywać, ani używać do identyfikacji osoby aktywującej konto, ponieważ ewentualne wykradzenie, lub skopiowanie bazy danych, umożliwi atakującemu aktywację wszystkich nieczynnych kont.

Zatem, pierwsze co jest robione, to zamiana ciągu binarnego na hexadecymalny. Stosowana jest do tego kolejna funkcja, *bin2hex()* języka PHP. Taki właśnie串 znaków jest dopiero przesyłany w linku do użytkownika. Natomiast w celach bezpieczeństwa,串 ten będzie przechowywany w rekordzie użytkownika jako hash, czyli wynik działania funkcji skrótu, podobnie, jak to miało miejsce przy zapisywaniu hasła. Używa się jednak do tego innej metody z zasobów dokumentacji PHP, *hash\_hmac*.

Po aktywowaniu linku z maila przez użytkownika, jego hexadecymalny串 zostaje na serwerze poddany działaniu tej samej funkcji skrótu i dopiero wynikowy hash wyszukiwany jest w bazie danych w celu identyfikacji osoby, której konto należy aktywować.

W opinii autora, takie właśnie zabezpieczenie, połączone z uwierzytelnieniem adresu e-mail i ochroną osobistego hasła, jest oczekiwane przez współczesnych użytkowników platform internetowych.

### Odzyskiwanie dostępu do konta

Na koniec opisu funkcji zabezpieczenia procesu autoryzacji dostępu do konta, należy wspomnieć także o sposobie rozwiązywania pojawiających się niekiedy problemów z dostępem użytkownika do swojego aktywowanego prawidłowo konta. Powodem takich sytuacji, może być na przykład, zapomnienie hasła użytego do rejestracji.

O ile tylko nie uległ utracie, lub zapomnieniu adres e-mail, odzyskanie dostępu do konta jest jak najbardziej możliwe i wg autora bezpiecznie rozwiązane w niniejszym projekcie.

W przypadku braku możliwości zalogowania się, użytkownik ma do wyboru opcję: *Zapomniałem hasła*, która skutkuje wyświetleniem formularza, przyjmującego tylko jeden wpis: wspomniany adres e-mail, który został użyty w czasie tworzenia konta. jest on potrzebny, ponieważ posłuży on do identyfikacji osoby, ubiegającej się o zresetowanie

hasła, oraz na ten adres zostanie przesłany link do formularza odzyskiwania hasła.

Po wpisaniu adresu i potwierdzeniu go w kolejnym polu, użytkownik zatwierdza formularz i jeśli tylko nie popełnił błędu we wpisywaniu, zawartość formularza zostaje przesłana na serwer. Tam w pierwszym rzędzie, dochodzi do sprawdzenia, czy przesłany adres figuruje w bazie danych. W przypadku nie znalezienia podanego adresu e-mail w bazie, odsyłana jest do aplikacji odpowiedź o kodzie 404, świadczącym o braku użytkownika w systemie. Taka też wiadomość jest wyświetiana użytkownikowi, z gdy ją zaakceptuje, jest przekierowywany do podstrony rejestracji konta.

W przypadku, gdy użytkownik o podanym adresie zostanie znaleziony w bazie, tok postępowania jest podobny do procesu aktywowania konta. Mianowicie, przesyłana jest na wspomniany adres wiadomość z linkiem do podstrony z formularzem resetowania hasła. W linku zawarty jest hexadecymalny ciąg znaków, powstały na tej samej zasadzie, co przy aktywacji. I podobnie jak wtedy, jego hashowana wersja, zapisana jest w rekordzie użytkownika w bazie danych. Użytkownik, po podaniu nowego hasła, jego powtórzeniu i sprawdzeniu po stronie aplikacji zgodności obu, zatwierdza przesłanie formularza na serwer. Tam, jak to już wiemy z poprzednio opisanego procesu, dołączony do żądania token hexadecymalny, zostaje poddany działaniu funkcji skrótu, a jej wynik porównany z przechowywanym hashem. Jeśli tylko zachodzi zgodność obu ciągów, odsyłana jest do użytkownika odpowiedź o możliwości zalogowania się nowymi danymi do naszej aplikacji.

Autor nie zdecydował się na użycie stosowanej niekiedy metody odzyskiwania hasła na podstawie zapisanych przez użytkownika odpowiedzi na pytania z procesu rejestracji, gdyż uważał, że zastosowanie nieodwracalnych w swym działaniu funkcji skrótu, daje większą pewność, że nikt niepowołany nie zdoła podszyć się pod właściciela konta. Metoda pytań i odpowiedzi, jest pod tym względem łatwiejsza znacznie do przełamania, a same odpowiedzi trudniejsze do przechowywania w bezpieczny sposób.

#### 4.3.2 Autoryzacja zapytań do bazy danych

Kwestie autoryzacji działań użytkownika w kontakcie z bazą danych, pozostają nierozerwalnie związane z wybranymi do implementacji obu stron aplikacji, technologiami.

Początkowo planowano dodać do projektu framework Vue.js jedynie jako zainportowany kod z zewnętrznego repozytorium CDN. Pozwalałoby to na zachowanie naturalnej łączności pomiędzy kontrolerami frameworka php Codeigniter, a widokami strony front-end. Model architektoniczny MVC (Model View Controller), jest pod tym względem bardzo przejrzysty, a autoryzacja kontaktu użytkownika z bazą danych na serwerze, z powodzeniem może być obsłużona dzięki stworzeniu sesji. Zawsze bowiem łatwiej jest frameworkowi back-end autoryzować komunikację ze „stworzonymi” przez siebie widokami, niż pochodzącymi z „obcego” źródła.

Dokonana zmiana sposobu korzystania z Vue.js, na korzyść pełnej instalacji Vue CLI, służącej do tworzenia stron typu SPA (Single Page Application), zaskutkowała istotnymi konsekwencjami w sposobie autoryzacji poczyniań użytkownika z częścią aplikacji działającą po stronie serwera.

Głównym następstwem tej decyzji, było czynnościowe, oddzielenie kontrolerów po stronie serwera od dotychczasowych widoków, które teraz stały się dla nich zupełnie obcymi podstronami. Mechanizm sesji przestaje być w takich okolicznościach skuteczny,

ponieważ „stan” użytkownika może się zmienić z żądania na żądanie, bez wiedzy serwera.

Należało tu zastosować mechanizm, który umożliwiłby sprawdzanie każdego docierającego żądania, indywidualnie, aby mieć pewność, że dana prośba pochodzi nadal od tego samego użytkownika, który został wcześniej prawidłowo zalogowany.

Zatem wspomniany stan użytkownika, powinien zostać teraz niejako przypisany do każdego wysyłanego zapytania, a to rodzi konieczność jego sprawdzania po stronie serwera. Rolę taką dobrze spełnia zaszyfrowany ciąg znaków, który stworzony na serwerze, podpinany jest do kolejnych zapytań. Początkowo wydawało się autorowi, że Codeigniter, framework obsługujący stronę back-end powinien taką możliwość zapewnić.

Rzeczywiście, jedną z funkcjonalności frameworka php Codeigniter 4, jest możliwość skorzystania z klasy Security, posiadającej 3 metody generujące token CSRF (Cross Site Request Forgery), uwierzytelniający przesyłanie żądań JSON.

Token taki, można przesyłać zarówno jako jeden z parametrów żądania, lub też umieszczając w specjalnym nagłówku HTTP, co umożliwia funkcja csrf\_field(). Ponieważ jednak opis sposobu stosowania tej techniki był bardzo skąpo wyjaśniony w oficjalnej dokumentacji, można było się jedynie domyślać, że mógłby pomóc w rozwiązaniu powstałego problemu.

Brak dokładnego wyjaśnienie zasady tworzenia tokena, oraz odpowiedzi na pytanie, czy jest odpowiedni do pracy ze stronami SPA, skłoniło jednak autora do szukania rozwiązania alternatywnego, lepiej wyjaśnionego, które dzięki temu można byłoby dostosować do wymogów aplikacji.

Stosunkowo dobrze wy tłumaczony, jeśli chodzi o tworzenie i stosowanie, oraz, co było ważne z punktu widzenia autora, popularny, okazał się token typu JWT – JSON Web Token.

Zasadniczo, token JWT, to podpisany obiekt JSON, zawierający informacje, pozwalające jego odbiorcy na uwierzytelnienie nadawcy zapytania. Podpis ten jest tworzony przy użyciu funkcji skrótu SHA-256, a użyty przy tym algorytm HMAC, pozwala na uzyskanie pewności, zarówno co do autentyczności, jak i nienaruszalności przesłanych danych.

Gotowy JWT, składa się z trzech części, złączonych ze sobą w jeden ciąg za pomocą znaku kropki. Tymi głównymi składowymi są:

1. zakodowany nagłówek
2. zakodowana zawartość – wspomniany obiekt JSON
3. zakodowana sygnatura

Nagłówek zawiera informacje o rodzaju i sposobie kodowania tokena, zawartość, czyli payload, to dane, które można załączyć do tokena, a które pomagają w jego walidacji po stronie serwera. Do tych informacji można zaliczyć czas ważności tokena, czy zakres dopuszczalnych dla użytkownika działań. Ostatnia część, to podpis cyfrowy, potwierdzający autentyczność danych.

Istnieją oczywiście, gotowe biblioteki dla każdego języka back-end, za pomocą których tworzy się JWT. W przypadku PHP, należą do nich:

- Firebase php-jwt
- Luciferous jwt

Tym razem wybór jednej konkretnej, nie był poprzedzony porównaniem ich cech, zamiast tego, zdecydowano się na pierwszą, z którą autor zetknął się podczas poszukiwań, a była to biblioteka firebase php-jwt. Zapewnia ona zarówno wsparcie w procesie two-

rzenia tokena w standardzie RFC 7519, jak i przy jego ekstrahowaniu i odkodowywaniu z przychodzącej na serwer komunikacji.

Jest to właściwie wszystko do czego służy ta biblioteka, ponieważ cała logika postępowania z JWT tokenem zależy już tylko od projektanta systemu i ogónie stosowanej strategii w środowisku deweloperskim.

Standardem, od którego w zasadzie nie można odejść, jest sam proces „nadawania” tokena stronie klienckiej aplikacji internetowej. W niniejszym projekcie zastosowano się do poniżej przedstawionego ogólnego sposobu postępowania.

Pierwsza wiadomość, z którą styka się serwer, a która z naturalnych przyczyn pozbawiona jest tokena, to przesłane ze strony web, dane logowania użytkownika, w naszym wypadku, adres e-mail, pełniący funkcję loginu, oraz hasło. Jako odpowiedź na fakt zalogowania, serwer odsyła stworzony dla tego właśnie użytkownika token, bazujący na jego unikalnym adresie skrzynki pocztowej. Model postępowania przy nadaniu tokena, przedstawia diagram (Rys.4.7).



Rysunek 4.7: Schemat przebiegu procesu nadawania JWT tokena

Problemem, jaki dotyczy momentu odebrania tokena przez stronę kliencką, była kwestia miejsca przechowywania otrzymanego „klucza do serwera”. W grę wchodziły trzy lokalizacje:

- LocaStorage przeglądarki internetowej
- magazyn Store zapewniany przez technologię Vuex dla frameworka Vue.js w wersji CLI
- przechowywanie w ciasteczkach cookies

LocalStorage jest bardzo wygodny i co ważne, stan zapisanych w nim informacji nie zmienia się w wyniku przechodzenia użytkownika pomiędzy zakładkami przeglądarki internetowej, czy odświeżenia strony aplikacji. Wadą jest natomiast, podatność na ataki typu XSS, z tego powodu jego użycie we wspomnianym celu jest odradzane.

Ciasteczka niestety, także nie są odporne na ataki XSS, a także są wrażliwe na inne, typu CSRF, więc ich używanie wymagałoby zastosowania skutecznej strategii anty-CSRF.

Wobec powyższego, autor zdecydował się na zapisywanie otrzymanego tokena w magazynie Store, służącym do przechowywania wartości zmiennych, charakteryzujących

stan działającego systemu. Niestety, ale to wyjście także nie jest doskonałe, gdyż zawarte w nim dane są tracone po odświeżeniu strony.

Dylemat nie został więc, definitywnie rozwiązany, a przyjęte wyjście jest traktowane jako tymczasowe, z możliwością ewentualnej zmiany, jeśli tylko nabywana z czasem wiedza pozwoli na podjęcie lepiej argumentowanej decyzji.

Dalszy scenariusz postępowania, nie jest już jednolity i może podlegać pewnym modyfikacjom, zależnym od zespołu deweloperskiego. Na pewno, powinien zostać wysłany razem z żądaniem ze strony klienckiej aplikacji. W przypadku bieżącego projektu, zdecydowano się na załączanie go w tworzonym do tego celu w każdym żądaniu wykonywanym przy użyciu Axios, specjalnym nagłówku autoryzacyjnym.

Na serwerze, pierwsze, na co napotyka takie żądanie, to filtr, stworzony w Codigniter 4, mający za zadanie wydobycie z nagłówka zaszyfrowanego tokena i jego autoryzację po odszyfrowaniu. Odszyfrowanie zawartości tokena, podobnie jak jego wcześniejsze stworzenie, zapewnia odpowiednia metoda, dostarczana z zainstalowaną biblioteką Firebase jwt-token. Walidacja, która odbywa się w filtrze ma na celu trzy zasadnicze sprawdzenia: - potwierdzenie, że token został stworzony przez nasz serwer - potwierdzenie, że przesłany e-mail należy do zarejestrowanego i aktywowanego użytkownika - potwierdzenie, że nie upłynął termin ważności tokena

To, czy to ten sam serwer przygotował odszyfrowywaną wiadomość, widać od razu, ponieważ proces szyfrowania, przy użyciu Firebase, odbywa się co prawda, przy zastosowaniu algorytmu HMSC SHA-256, ale na bazie osobistego klucza szyfrowego, przechowywanego w pliku .env na serwerze. Po prostu, token przygotowany przez kogoś innego nie zostanie prawidłowo odszyfrowany.

Czyli mamy sytuację, jeszcze przed uruchomieniem docelowej dla zapytania, metody konkretnego kontrolera, rozpoczęta została walidacja tokena.

Odszyfrowany payload tokena, o którym wspomniano przy opisie jego budowy, zawiera między innymi dane potrzebne do jego walidacji po odesłaniu z żądaniem na serwer, takie jak termin ważności, czy e-mail użytkownika. Termin przydatności jest kryterium określającym, czy dany token może jeszcze być uznany przez serwer, czy też jest już przestarzały i użytkownik musi się zalogować ponownie, żeby otrzymać nowy token.

Krótki czas życia, spowoduje zatem, niespodziewane przerwanie pracy użytkownika z aplikacją i konieczność ponownego logowania. Kuszące wydawało się przyznanie tokenowi odpowiednio długiego czasu trwania, aby użytkownikowi pozostawić komfort nieprzerwanej pracy.

Trudność polegała na określeniu jak długo użytkownik zamierzał będzie korzystać z aplikacji po jej otwarciu. Poza tym, jednak, nadanie tokenowi długiego terminu ważności może mieć niekorzystne następstwa w przypadku jego wykradzenia, ponieważ zapewniłoby to również napastnikowi długi dostęp do serwera i jego danych. Z tego głównie powodu, w przypadku naszego projektu, nie zmieniono ogólnej strategii nadawania ważności tokenowi i poprzestano na wyznaczeniu mu 10-cio minutowego okresu życia.

Ale problem nagłego przerywania pracy użytkownika pozostał. W tym momencie doszły do głosu konsekwencje wyboru biblioteki przygotowywania JWT, bez dogłębnego rozpoznania i porównania różnych rozwiązań. Otóż zazwyczaj, biblioteka taka, powinna zapewnić, oprócz możliwości tworzenia tokena, również jednoczesne tworzenie specjalnego tokena odświeżającego. Rola tego ostatniego polega na tym, że jest on przesyłany

do aplikacji jednocześnie z tym pierwotnym, ale nieco innym sposobem, bo za pomocą `http_cookie` – ciasteczka lepiej zabezpieczonego przed atakiem CSRF niż zwykłe.

Taki `refresh_token`, przechowywany w aplikacji, jest przesyłany na serwer tuż przed upływem ważności pierwotnego tokena, niejako w tle, nie towarzysząc żadnemu zapytaniu do bazy danych.

Na jego podstawie, serwer jest w stanie wydać nowy token, zanim stary stanie się przestarzały. Jest to tzw ciche odświeżanie, przedłużające autoryzowaną pracę użytkownika. Niestety, Firebase jwt-token, zastosowany w niniejszym projekcie, nie zapewnia tej funkcjonalności. Zmusiło to autora do przygotowania innej strategii, która nie naruszałaby zasad bezpieczeństwa, a jednocześnie poprawiłaby komfort używania systemu.

Cele do osiągnięcia, przedstawały się następująco: - nadany token nie powinien pozwalać na autoryzację pracy z bazą danych dłużej niż 10 minut - ewentualne przechwycenie tokena, nie powinno pozwalać atakującemu na dostęp dłużej niż 10 minut - użytkownik nie powinien być zmuszany do ponawiania logowania, aż do samodzielnej decyzji o zakończeniu pracy, lub do momentu odświeżenia strony aplikacji

Pozornie sprzeczne ze sobą oczekiwania, udało się pogodzić dzięki tworzeniu tokena na nowo na serwerze, po każdym otrzymaniu żądania i dołączaniu tej jego nowej edycji do każdej odpowiedzi. W aplikacji, nowy token zastępował starą wersję w magazynie Store, aby zostać użyty w kolejnym zapytaniu.

Asynchroniczność żądań, nie była tu przeszkodą, która poprzez przesłanie z aplikacji starszego, niż już wystawiony, tokena, mogłaby zakłócić proces uwierzytelniania. Powodem jest to, że produkowane tokeny, nie są przechowywane po stronie serwera, więc jeśli tylko token nie uległ przeterminowaniu i zgadzały się pozostałe punkty validacji, był akceptowany.

Niemniej, w ramach dalszego rozwoju projektu, należy zaplanować działania mające na celu zmianę biblioteki obsługującej JWT, na taką, która zapewni pełną obsługę mechanizmu JSON Web Token.

#### 4.3.3 Inne zastosowane zabezpieczenia

Na koniec niniejszego rozdziału, warto jeszcze wspomnieć o takich rodzajach zastosowanych zabezpieczeń, których użytkownik raczej nie będzie w stanie zauważać, gdyż ich działanie nie wymaga od niego żadnej aktywności. Mowa tu przede wszystkim o zapobieganiu wprowadzaniu obcego kodu, głównie html (ataki XSS), oraz sql (SQL injection).

Wiele takich zabezpieczeń jest niewidocznych, a raczej nie wymagają aktywnego uruchamiania przez developera, głównie dzięki temu, że framework Codeigniter 4 reprezentuje pod tym względem wysoki poziom. Zaliczyć tu można wspomnianą ochronę przed atakami XSS. Wszystkie wprowadzane do systemu dane są sanityzowane pod kątem obecności kodu html. Próby wprowadzania jako zawartości kart do nauki, ciągów zawierających tagi html, kończą się wyświetleniem ich użytkownikowi w nieinterpretowanej postaci.

Co do ochrony przed *SQL injection*, to i tutaj framework zadba sam o oczyszczanie otrzymywanych zapytań z możliwych szkodliwych wstępów, o ile korzysta się przy obsłudze bazy danych z klasy *Query Builder*, której metody są wyposażone we wspomniane zabezpieczenia.

Autor przy tworzeniu zapytań do bazy MySQL, korzystał jednak na równi z ułatwień

dostarczanych przez klasę *Query Builder*, jak i konstruując je samodzielnie w postaci string.

W tym drugim przypadku, Codeigniter też wyręcza w oczyszczaniu wprowadzanych do zapytania zmiennych z potencjalnie niebezpiecznych dodatków, pod warunkiem stosowania funkcjonalności pod nazwą *Query Biding*, polegającej na wprowadzaniu tychże zmiennych do zapytania nie bezpośrednio, lecz poprzez metodę *query()*, w której tablica owych zmiennych stanowi jeden z przyjmowanych argumentów.

Podsumowując, trzeba przyznać, że praca z frameworkm Codeigniter w kontekście spraw zabezpieczeń, dała duże poczucie komfortu autorowi.

#### 4.4 Wymagania funkcjonalne i niefunkcjonalne

W niniejszym rozdziale zostały przedstawione ogólne założenia funkcjonalne programu *Repeater*, oraz sposoby w jakich system będzie reagował na aktywności pochodzące ze środowiska zewnętrznego. Pełna specyfikacja wymagań, sporządzona zgodnie z wzorcem IEEE, została przedstawiona w załączniku A, dołączonym na końcu dokumentacji projektowej.

System *Repeater*, został zaprojektowany jako aplikacja działająca w przeglądarkach internetowych, z koniecznością stałego dostępu do sieci internetowej. Wszystkie funkcjonalności systemu są pochodną celu, dla którego realizacji powstał. Repeater jest platformą do nauki pamięciowej, poprzez prezentowanie słów obcojęzycznych, w celu zapamiętania ich przez użytkownika, a także pełni rolę organizatora powtórek uczonego materiału, wyznaczając użytkownikowi rozłożone w czasie kolejne prezentacje zapamiętanych danych.

Z tego względu, potencjalnym użytkownikiem programu powinna być osoba zaznajomiona z obsługą przeglądarek internetowych, umiejąca w razie potrzeby je instalować i uaktualniać. Dodatkowa umiejętność, jakiej wymagać może od użytkownika obsługa systemu, to umiejętność operowania plikami csv, z uwagi na format wprowadzania własnych danych do systemu. Oczywiście, regularnie częste uruchamianie aplikacji, jest zalecane, aby nie dochodziło do utraty postępów w nauce pamięciowej, zgodnie z wytycznymi opisanyymi we wcześniejszych rozdziałach.

##### Środowisko pracy systemu

System zaprojektowany został do uruchamiania w przeglądarkach internetowych Chrome (od wersji 63+), Firefox (wersja 40+), Safari (7+), oraz Opera, działających na urządzeniach z systemami operacyjnymi: Windows (Vista lub nowszy), Linux, macOS, oraz Android.

Biorąc pod uwagę ogólny podział środowiska pracy, działanie programu odbywa się zarówno po stronie klienta, udostępniając użytkownikowi i odbierając od niego dane na ekranie urządzenia, jak i po stronie serwerowej, gdzie umieszczono usługę bazy danych.

Interakcja z końcowym użytkownikiem, odbywa się przy użyciu peryferiów takich jak myszka i klawiatura, za pomocą których korzysta się w obsłudze z formularzy, przycisków, oraz rozwijanych menu. Możliwe jest też przesyłanie i zapisywanie w bazie danych na serwerze, danych, przygotowanych w formacie CSV. Jednak przesyłanie danych wewnętrz aplikacji, pomiędzy stroną klienta a serwerem, odbywa się już po przekonwertowaniu ich na format JSON, w celu lepszej kompatybilności.

Do komunikacji pomiędzy tymi stronami, zastosowano technologię AJAX, realizowaną przy pomocy biblioteki Axios.js. Wynikająca z tego konieczność posiadania przez aplikację stałego połączenia z internetem o minimalnej szybkości przesyłu danych rzędu 0,5 Mb/s, nie powinna być istotnym ograniczeniem dla użytkownika w obecnym stanie rozwoju infrastruktury komunikacyjnej.

Od strony programistycznej, wspomnieć tylko tutaj można o zastosowaniu framework'a Vue.js w przygotowaniu aplikacji od strony front-end, oraz framework'a Codeigniter 4 do realizacji back-end, a sama baza danych działa na silniku MySQL.

#### Funkcjonalności systemu

Pełny opis funkcjonalności, wraz z diagramami przypadków użycia, został umieszczony w dodatku A, w tym miejscu zatem, ograniczono się do przedstawienia ogólnej zasady ich podziału.

Wymagania funkcjonalne systemu Repeater, można umieścić w kilku zasadniczych grupach organizacyjnych. Poniżej, zamieszczono gotowy podział, uzupełniony o składowe funkcjonalne każdego działu:

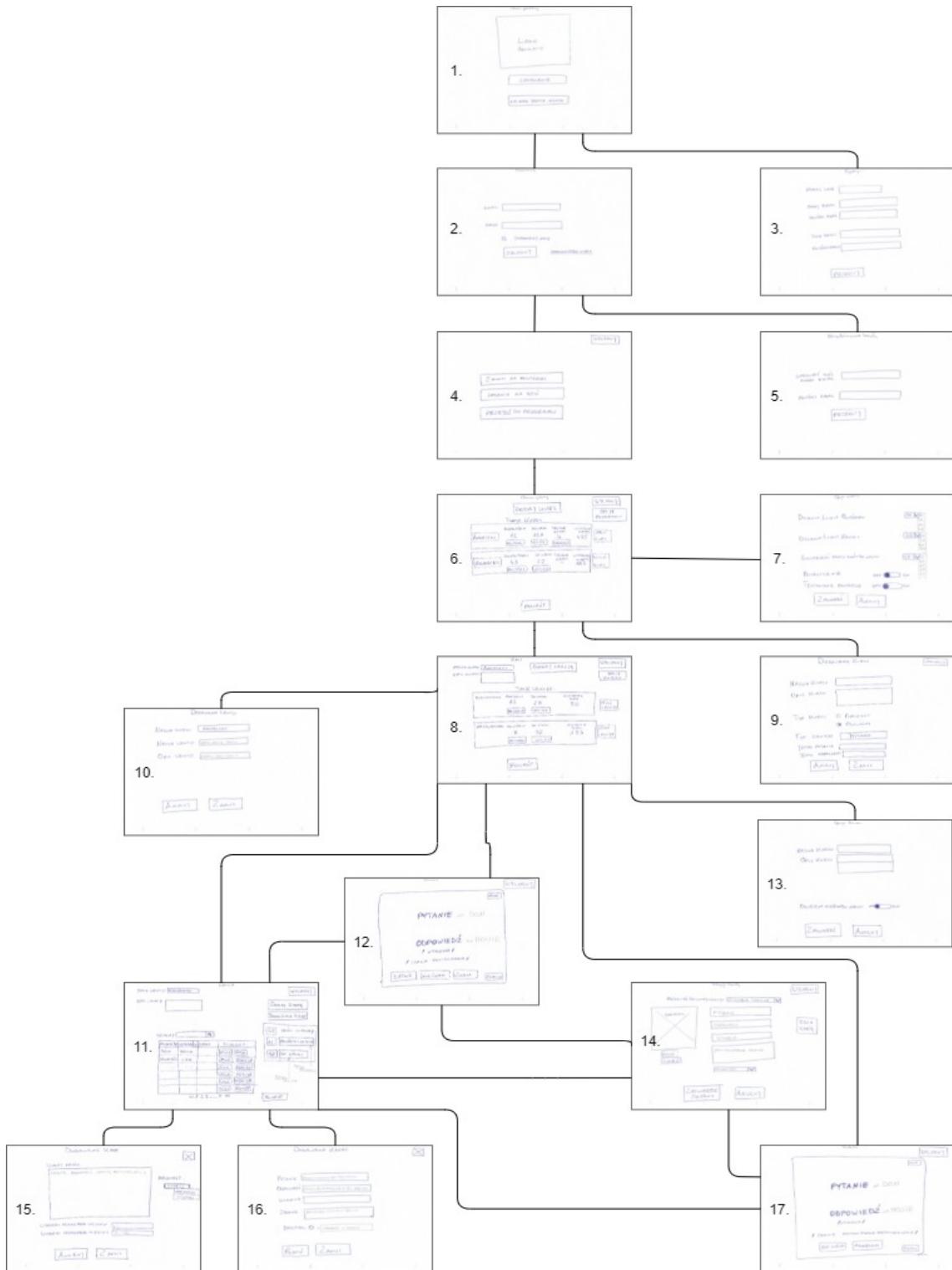
1. Procedura zakładania konta użytkownika
  - tworzenie konta użytkownika
  - aktywacja konta przy użyciu maila aktywacyjnego
  - likwidacja konta przez użytkownika
  - automatyczne zamknięcie konta nieużywanego
2. Proces logowania do aplikacji
  - zmiana hasła użytkownika po jego zapomnieniu
  - zalogowanie się użytkownika po podaniu przez niego danych rejestracyjnych
  - wylogowanie się użytkownika w dowolnym momencie jego pracy z programem
3. Proces nauki nowych kart
  - przeprowadzenie fazy nauki partii kart
  - przeprowadzenie etapu przeuczenia
  - przeprowadzenie podsumowania po zakończeniu każdej fazy nauki
4. Przeprowadzanie powtórek materiału
  - wyznaczanie terminu kolejnej powtórki materiału
  - przeprowadzanie powtórek zapamiętanego materiału
  - przeprowadzanie krótkich szybko dostępnych sesji powtórkowych, wprost z ekranu głównego
  - przeprowadzanie pierwszej powtórki już po 10-ciu minutach od zakończenia nauki
5. Edytowanie danych
  - tworzenie nowego kursu w obrębie konta użytkownika
  - tworzenie nowej lekcji w obrębie bieżącego kursu
  - usuwanie kursu przez użytkownika wraz z przynależnymi do niego lekcjami i kartami
  - usuwanie lekcji przez użytkownika wraz z przynależnymi do niej kartami
  - edycję karty w oknie lekcji
  - edycję karty w trakcie nauki i powtórek
  - usuwanie karty z programu, zarówno w czasie pracy z nią, jak i w lekcji
  - wyszukiwanie słowa/karty w tabeli w zakładce lekcji
6. Ustawienia opcji programu
  - wyświetlenie opcji ustawień ogólnych programu
  - edycja opcji bieżącego kursu
  - zmiana limitu dziennego nauki w programie
  - zmiana limitu dziennego powtórek w programie

- dodanie/usunięcie obrazka karty
- usunięcie dziennego limitu kart do nauki i powtórki

#### 7. Eksport i import danych

- dodawanie kart pojedynczo
- dodawanie wielu kart jednocześnie, jako kopia zawartości pliku csv
- eksportowanie zbioru kart danej lekcji, lub kursu do pliku tekstowego

## 4.5 Diagramy LOFI

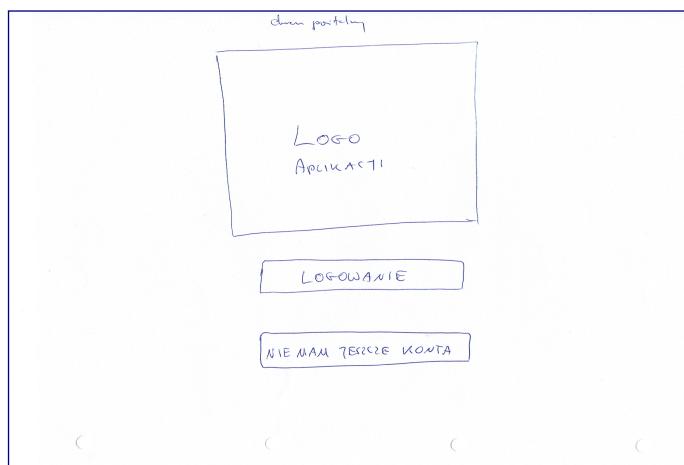


Rysunek 4.8: Diagram możliwych przejść pomiędzy ekranami aplikacji

Załączony (Rys.4.25) całościowy diagram możliwych przejść pomiędzy ekranami pro-

gramu, został na kolejnych stronach rozłożony na poszczególne ekrany, o numeracji porządkowej, zgodnej z poniższą listą:

1. Ekran powitalny
2. Formularz logowania
3. Formularz rejestracyjny
4. Ekran szybkiego wyboru działań
5. Formularz odzyskiwania hasła
6. Okno główne z listą kursów użytkownika
7. Ekran edycji opcji ogólnych programu
8. Okno wnętrza kursu z listą lekcji danego kursu
9. Formularz tworzenia nowego kursu
10. Formularz tworzenia nowej lekcji
11. Okno wnętrza lekcji z tabelą kart danej lekcji
12. Widok z fazy nauki/zapamiętywania
13. Ekran edycji opcji kursu
14. Ekran edycji karty/słowa
15. Formularz dodawania pojedynczej karty/słowa
16. Formularz dodawania wielu kart jednocześnie
17. Widok z fazy powtórki



Rysunek 4.9: **Ekran nr 1. diagramu** - ekran powitalny, umożliwiający przejście do logowania użytkownika, lub otwarcie formularza zakładania konta

A hand-drawn sketch of a login screen titled "Logowanie". It features two input fields: "EMAIL" and "HASŁO", each with a rectangular input box. Below these fields is a checkbox labeled "ZAPAMIĘTAJ MIE". At the bottom left is a blue-outlined button labeled "ZALOGUJ", and at the bottom right is a blue-outlined link labeled "ZAPOMNIĘTE HASŁA".

Rysunek 4.10: **Ekran nr 2. diagramu** - ekran logowania, to formularz wprowadzania danych autoryzacyjnych użytkownika

A hand-drawn sketch of a registration screen titled "Rejestracja". It contains five input fields: "PODAJ IMIĘ" (with one input box), "PODAJ EMAIL" (with two input boxes stacked vertically), "POSIĘRZ EMAIL" (with one input box), "TWÓJ HASŁO" (with one input box), and "POSIĘRZ HASŁO" (with one input box). At the bottom center is a blue-outlined button labeled "PRZESŁIJ".

Rysunek 4.11: **Ekran nr 3. diagramu** - ekran rejestracji, przedstawia formularz zakładania konta użytkownika

A hand-drawn sketch of a quick action selection screen. In the top right corner is a blue-outlined button labeled "WYLOGUJ". The main area contains three blue-outlined buttons: "2 MINUTY NA POWTÓRKI", "ZADANIA NA DZIS", and "PRZEJDŹ DO PROGRAMU".

Rysunek 4.12: **Ekran nr 4. diagramu** - ekran szybkiego wyboru działań, pozwala na rozpoczęcie wykonywania najpilniejszych zadań, np powtórek, bez konieczności przeszukiwania kursów i lekcji

Odzyskiwanie hasła

WPROWADŹ SWÓJ ADRES EMAIL

PONOWIĘ EMAIL

**PRZESŁIJ**

Rysunek 4.13: **Ekran nr 5. diagramu**- formularz odzyskiwania zapomnianego hasła, po podaniu adresu mailowego zarejestrowanego użytkownika, na jego skrzynkę przesyłany jest link do formularza zmiany hasła

Edukator

DODAJ KURS

WYLOGUJ

OPCJE PROGRAMU

Tworzone KURSY

ANGIELSKI	DOPAŁÓWEKI 12	DOKAŃKI 128	TRĄBNE KARTY 4	WYSZCZEGÓLNIENIA KARTY 435	USUN KURS
NIEMIECKI	DOPAŁÓWEKI 43	DOKAŃKI 22	TRĄBNE KARTY -	WYSZCZEGÓLNIENIA KARTY 183	USUN KURS

POWRÓT

Rysunek 4.14: **Ekran nr 6. diagramu** - okno główne programu z widokiem listy kursów użytkownika i dostępem do opcji głównych całej aplikacji. W paskach kursów wyszczególnione są najważniejsze dane dotyczące kart/słów należących do danego kursu

Opcje główne

DZIENNY LIMIT PÓŁDZIEK

DZIENNY LIMIT NAUKI

LICZENIEŚĆ PARTII SLOW DO NAUKI

PRZEUCZENIE  OFF  ON

TESTOWANIE DOKIĘDZIE  OFF  ON

ZATWIERDZ ANULUJ

Rysunek 4.15: **Ekran nr 7. diagramu** - ekran przedstawiający formularz edycji opcji głównych programu, mających zastosowanie we wszystkich aktywnościach użytkownika

Kurs

NAZWA KURSU: ANGIELSKI  
OPIS KURSU:

DODAJ LEKCJĘ

WYLOGUJ

OPCJE KURSU

TUOŁE LEKCJE:

RZECZOWNIKI	PÓŁTÓRKI	DO NAUKI	WYSZCZEGÓL. SŁÓW
12	28	50	50
<input type="button" value="POWТОРЗ"/>	<input type="button" value="UCZ SIĘ"/>		<input type="button" value="USUN LEKCJĘ"/>

PRZYKŁADNIKI	PÓŁTÓRKI	DO NAUKI	WYSZCZEGÓL. SŁÓW
8	32	133	133
<input type="button" value="POWТОРЗ"/>	<input type="button" value="UCZ SIĘ"/>		<input type="button" value="USUN LEKCJĘ"/>

POWRÓT

Rysunek 4.16: Ekran nr 8. diagramu - ekran wnętrza kursu z listą przynależnych do niego lekcji; w paskach lekcji zawarte są najistotniejsze informacje dotyczące kart/słów z danej lekcji

DODAWANIE KURSU

WYLOGUJ

NAZWA KURSU:

OPIS KURSU:

Typ KURSU:  PUBLICZNY  
 PRYWATNY

Typ DANYCH: TEŻYCIOWY

Pytań:

Odpowiedzi:

Rysunek 4.17: Ekran nr 9. diagramu - widok formularza tworzenia nowego kursu; utworzenie kursu umożliwia dodanie do niego lekcji i kart/słów

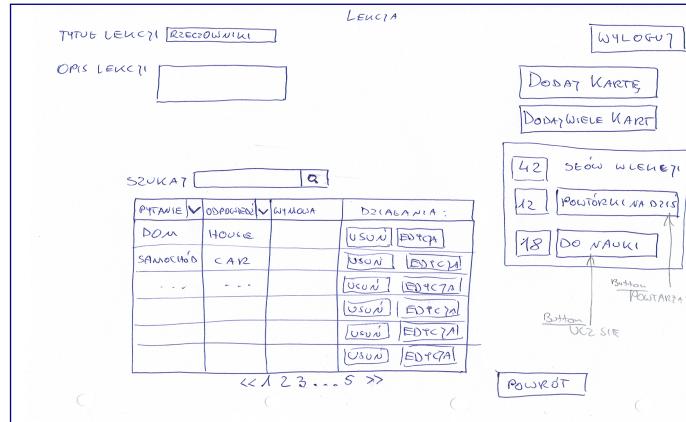
DODAWANIE LEKCJI

NAZWA KURSU: ANGIELSKI

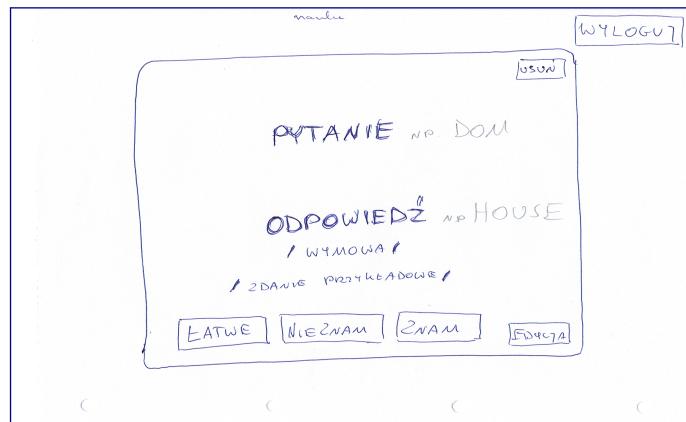
NAZWA LEKCJI: Wpisz nazwę lekcji

OPIS LEKCJI: Krótki opis lekcji

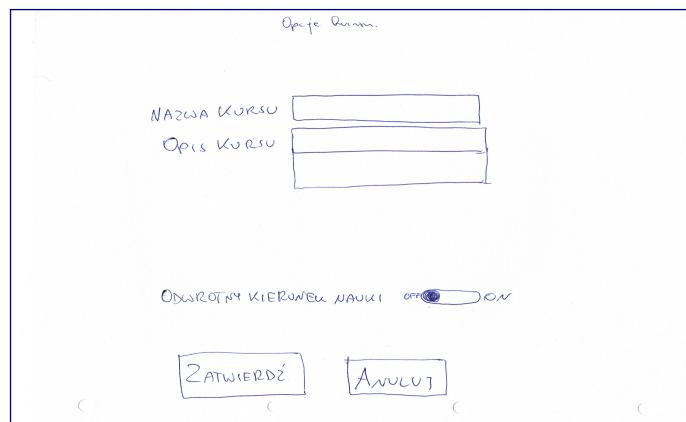
Rysunek 4.18: Ekran nr 10. diagramu - widok formularza tworzenia nowej lekcji w obrębie bieżącego kursu użytkownika; utworzenie lekcji umożliwia dodawanie kart/słów do niej



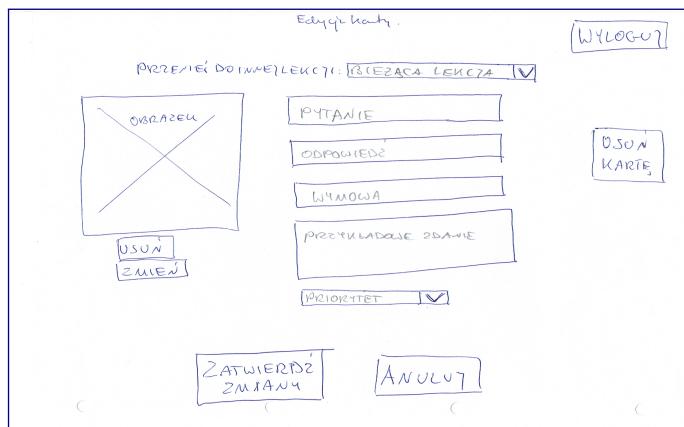
Rysunek 4.19: **Ekran nr 11. diagramu** - ekran przedstawiający widok wewnętrza lekcji wraz z tabelą sortowaną kart/słów należących do tej lekcji



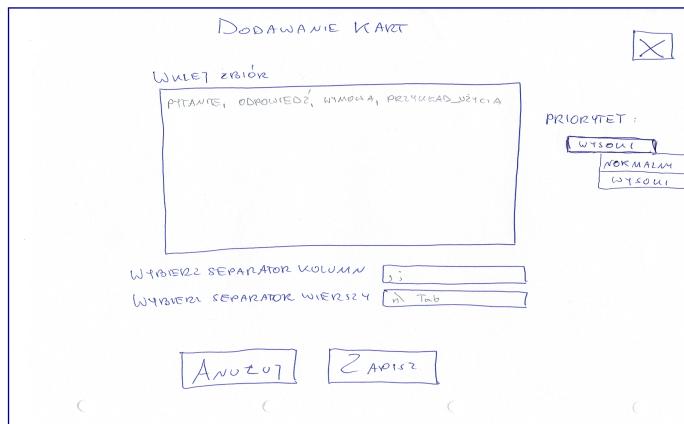
Rysunek 4.20: **Ekran nr 12. diagramu** - widok z fazy nauki/zapamiętywania - wygląd ekranu po zatwierdzeniu przez użytkownika gotowości do odpowiedzi; w takim stanie program oczekuje na wprowadzenie oceny stopnia znajomości hasła



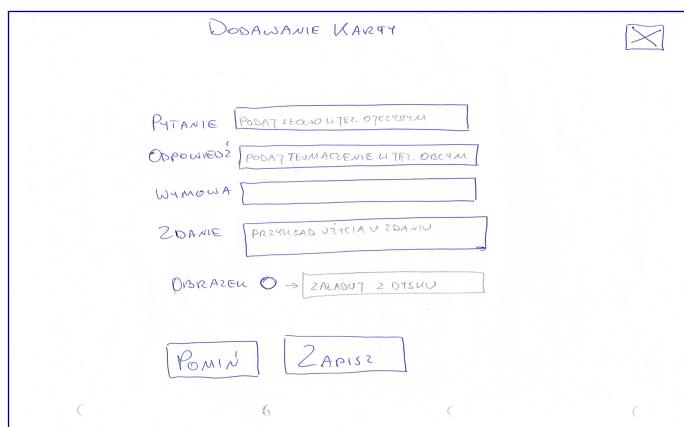
Rysunek 4.21: **Ekran nr 13. diagramu** - ekran przedstawiający formularz edycji opcji właściwych dla bieżącego kursu; nie mają one wpływu na ustawienia innych kursów



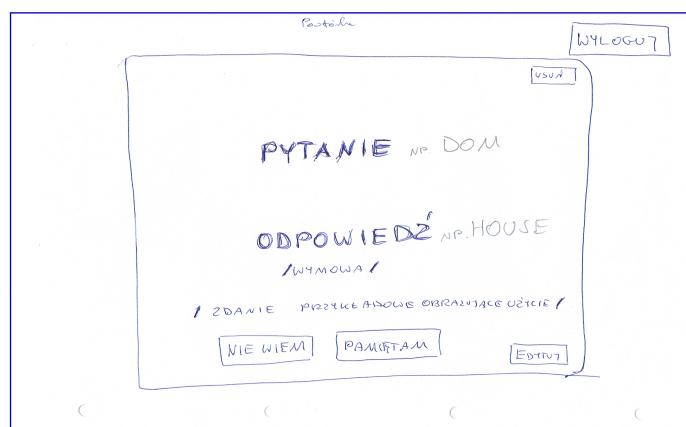
Rysunek 4.22: **Ekran nr 14. diagramu** - ekran edycji pojedynczej karty/słowa - dostęp do niej użytkownik posiada zarówno z poziomu lekcji danej karty, jak i każdej aktywności, w której widoczna jest jej zawartość



Rysunek 4.23: **Ekran nr 15. diagramu** - ekran przedstawia formularz umożliwiający dodanie jednorazowo wielu kart, poprzez wklejenie zawartości pliku csv do pola tekstowego formularza



Rysunek 4.24: **Ekran nr 16. diagramu** - ekran formularza, umożliwiającego dodanie pojedynczej karty/słowa; ekran pozostaje widoczny po zatwierdzeniu wpisanego słowa, umożliwiając użytkownikowi dalsze wprowadzanie słów bez konieczności ponownego otwierania okna



Rysunek 4.25: **Ekran nr 17. diagramu** - widok jaki ukazuje się użytkownikowi, podczas przeprowadzania fazy powtórki, po potwierdzeniu przez niego gotowości oceny znajomości słowa; w tym stanie, program oczekuje na wybór przez użytkownika przycisku z oceną

# 5. Wybór technologii

## 5.1 Kliencka strona aplikacji

Przy podejmowaniu decyzji o sposobie implementacji projektu po stronie wizualnej, front-endowej, do wyboru, w grę wchodziło kilka możliwości, począwszy od pozornie najprostszej, czyli Vanilla JS, bez wsparcia frameworka.

Doświadczenie autora wyciągnięte z projektów przygotowywanych w toku 2 ostatnich lat studiów, sugerowało jednak, że w miarę rozrostu aplikacji i wzbogacania jej o nowe funkcjonalności, możliwości porządkowania kodu i odnajdywania w nim konkretnych miejsc, drastycznie maleją. Przyjęto założenie, że skoro rozmiar projektu wykracza poza wszystko, co dotychczas implementowano, ryzyko utraty spójności kodu i idące z nim w parze straty czasu, mogą postawić realizację projektu pod znakiem zapytania. Należało również, brać pod uwagę to, że jakość i ilość wsparcia ze strony społeczności programistów, dotycząca czystego kodu JavaScript, tzw VanillaJS, będzie zbyt mała do rozwiązania pojawiających się problemów. Z tych powodów, autor podjął decyzję o realizacji części klienckiej w oparciu o gotowy framework JS, a z powodu braku doświadczenia z którymkolwiek, należało dokonać selekcji spośród rozwiązań obecnych na rynku.

Wybór konkretnego frameworka jest trudnym przedsięwzięciem przy braku osobistego doświadczenia z każdym z nich. Z konieczności oprzeć się trzeba o charakterystyki i opinie innych użytkowników. Obecność porównań, z których każde oparte jest o odmienne cechy, sprawiła, że konieczne stało się opracowanie przez autora własnych kryteriów, na podstawie których można byłoby dokonać wyboru.

Porównaniu poddano zatem, następujące frameworki:

- Angular
- React
- Vue
- Ember
- Svelte

Wyodrębniono również jednolite kryteria porównania wymienionych technologii:

- Dopasowanie do charakteru projektu
- Elastyczność
- Dokumentacja frameworka
- Łatwość nauki i pracy z frameworkiem – krzywa uczenia
- Popularność i wsparcie społeczności

Bez wątpienia, można wymienić wiele innych cech, różniących wymienione technolo-

logie, jak na przykład: skalowalność, szybkość, progresywność, czy rodzaj kreowanego DOM(wirtualny lub nie). Niemniej, wobec niewielkiego doświadczenia autora w pracy z frameworkami front-end w ogóle, nie miały one tak dużego znaczenia przy wyborze technologii, jak wymienione wcześniej cechy.

### Dopasowanie do charakteru projektu

Angular, z racji stopnia trudności i złożoności, jest powszechnie rekomendowany dla tworzenia dużych projektów komercyjnych, tworzonych zarówno przez korporacje jak i mniejsze wieloosobowe zespoły. W małych niedoświadczonych zespołach, rekomendowane jest nie stosowanie tego framework'a. Z kolei Vue jest mały i lekki pod względem zajętości miejsca, i który dzięki temu może zarówno pomóc w stworzeniu dużej witryny, jak i niewielkiej strony. Można go z powodzeniem zastosować przy tworzeniu rozrastających się z czasem witryn, bądź projektów, których koncepcje w toku tworzenia są zmieniane. Nie jest natomiast rekomendowany dla aplikacji, których główną cechą ma być niezawodność działania, ponieważ zgłaszane są problemy ze stabilnością jego komponentów.

Ember, z kolei, chociaż poleca się go w tworzeniu nowoczesnych aplikacji z bogatym interfejsem użytkownika, to jednak brak doświadczenia deweloperskiego, może znacznie utrudnić korzystanie z jego pomocy.

Na koniec pozostał Svelte, dla którego zalecenia stosowania były zgodne, jeśli chodzi o wielkość mojego projektu.

### Elastyczność/wszechstronność

Z punktu widzenia autora pracy, w podejściu do wyboru oprogramowania strony wizualnej projektu, istotne było zagwarantowanie przez wybrany framework, elastyczności w tworzeniu kodu programowego. Innymi słowy, możliwość wyboru stopnia złożoności rozwiązań, zależnie od aktualnych potrzeb i stopnia rozwoju aplikacji.

Spośród branych pod uwagę frameworków front-end, największą możliwość dopasowania programu do sytuacji i preferencji programisty, daje Vue.js. Począwszy od samej instalacji, użytkownik ma wybór pomiędzy prostym zainstalowaniem Vue ze zdalnego repozytorium, lub może zdecydować się na instalację CLI, która wzbogaca środowisko projektu o moduł testowania, routingu, czy Vuex, służący do obsługi stanu aplikacji. Także na późniejszym etapie rozwoju programu, można zdecydować, czy poprzestać na wstawieniu do kodu html tylko dynamicznych wstawek JavaScript, czy też stosować całe komponenty.

Na nieco mniejszą swobodę zmiany decyzji w tworzeniu kodu, możemy pozwolić sobie w przypadku frameworku React, który wymaga już instalacji, a później również umiejętności dostosowania opcji konfiguracyjnych.

Jednak najbardziej rygorystyczny pod względem przestrzegania reguł stylu programowania, jest z kolei, Angular, co zapewne wynika z jego przeznaczenia do tworzenia dużych aplikacji korporacyjnych, w licznych zespołach programistów, co z kolei wymusza w takich okolicznościach konieczność pełnej formalizacji i standaryzowania.

Wspomniana zaleta elastyczności, przejawiająca się w indywidualnym podejściu do poziomu skomplikowania struktury kodu, rodzi jednak też trudności, zwłaszcza w pracy

nad projektem zespołowym – powodując powstanie rozbieżności w stylu programowania pomiędzy uczestnikami projektu i brak jednolitego stylu pracy. W naszym jednak przypadku, gdy jest jeden uczestnik projektu, pozostaje nadal znaczącą zaletą, pozwalającą autorowi, na dopasowanie tworzonego programu do swoich preferencji i aktualnych umiejętności.

### Jakość dokumentacji

Oficjalnie udostępniana dokumentacja, jest najważniejszym źródłem wiedzy o danej technologii w pracy nad aplikacją, gdyż to od niej zależy szybkość i łatwość tworzenia projektu. Porównywane frameworki, na ogół nie wyróżniają się in minus pod tym względem.

Jedynie React, jako rozwijający się bardzo szybko framework, pozostaje nieco w tyle, głównie z powodu nie nadążania oficjalnych opracowań za zachodzącymi zmianami. Stawia to użytkowników, zwłaszcza niedoświadczonych, przed licznymi problemami implementacyjnymi.

Co ciekawe, trudno było uzyskać informacje o jakości dokumentacji Svelte i jej powszechnym odbiorze, a to z powodu niewielkiej społeczności i stosowności tegoż – co w sumie, jest też samo w sobie jakąś informacją o stanie dokumentacji tej technologii.

Natomiast dokumentacja dotycząca Vue.js oraz Angular, jest według powszechnej opinii, bardzo dobrze przygotowana i uaktualniana na bieżąco.

### Krzywa nauki

Rozpoczęcie pracy z nowymi technologiami wytwarzania oprogramowania, rodzi dla tworzonego projektu oczywiste zagrożenia. Głównym z nich jest ryzyko utraty zbyt dużej ilości czasu na naukę danej technologii. Należy zawsze założyć jako pewne, pojawienie się licznych trudności w trakcie pracy z nowym frameworkm. Mając to na uwadze, celowe było więc, oparcie kryteriów wyboru w przeważającej mierze, na tzw spodziewanej krzywej nauki porównywanych technologii.

Największego wysiłku poznawczego, zarówno na etapie początkowym, jak i w późniejszym czasie pracy nad aplikacją, wymagałby tu Angular, według powszechnej opinii społeczności programistów. Dodatkowym utrudnieniem w nauce Angulara jest konieczność opanowania dodatkowo TypeScript'a, z którym jest on nierozerwalnie związany. Ponadto, paradoksalnie, pracy z Angularem wcale nie ułatwia niewielki rozmiar niniejszego projektu, ponieważ to nie zwalnia programisty z konieczności stosowania złożonych aspektów wspomnianego frameworka.

Porównywalny stopniem trudności jest także Ember, na którego nieelastyczność struktur skarżą się programiści. Dla odmiany, stosunkowo łatwym i nie sprawiającym trudności w pracy na początku, jest React i Svelte. Na późniejszych etapach pracy, zarówno z React'em, jak i Svelte, pojawiają się jednak trudności, na co skarży się wielu użytkowników obu tych frameworków. Takim utrudnieniem w przypadku tego ostatniego, jest potrzeba uzupełniania brakujących funkcjonalności, własnym kodem, zaś w pracy z Reactem, trudność sprawia składnia JSX (format włączania kodu html i css w kod JS).

Vue zaś, uznany za najłatwiejszy w nauce, sprawiłby najmniej trudności, zwłaszcza, że można w jego przypadku dostosować używane funkcjonalności do swojego bieżącego zaangażowania.

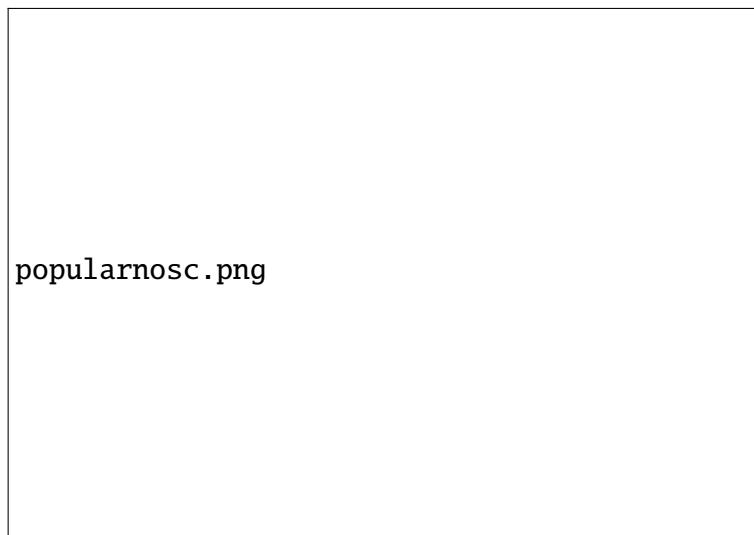
### Popularność

Kryterium to, mimo że posługiwanie się nim sugeruje chęć łatwego dostosowania się do preferencji większości, miało jednak niebagatelną wagę, z punktu widzenia autora pracy. Ryzyko braku wsparcia w rozwiązywaniu powstających problemów, jest realnym zagrożeniem dla realizacji każdego projektu, zwłaszcza gdy nie pracuje się w zespole. Popularność stosowania danej technologii, jest gwarancją istnienia wielu źródeł rozwiązań rozmaitych trudności.

W ocenie stopnia popularności, bądź powszechności danego rozwiązania, można oprzeć się na kilku kryteriach. Do wyboru jest tzw liczba gwiazdek GitHub, ilość pobrań projektów wykonanych w danej technologii, liczba projektów wykonywanych na świecie przy wsparciu danej technologii, czy ilość użytkowników serwisu stack overflow, wypowiadających się w danym temacie.

Napotkane rozbieżności w ilości gwiazdek GitHuba, oraz podejrzanie wysoka pozycja Vue.js w tym rankingu wyższa niż React), nie mająca potwierdzenia w innych źródłach zestawień, nie pozwala jednak stosować tej skali jako jedynej. Również rankingi dostępne na stronie <https://insights.stackoverflow.com>, nie odzwierciedlały rzeczywistego stanu stosownalności danego framework'a, ponieważ oceniały jedynie preferencje, nie zaś realne użycie w projektach.

Z uwagi na rzetelność i wiarygodność, a przede wszystkim na bezpośrednie przełożenie na skalę używalności technologii, ocenę w tym wypadku oparto na liczbie pobrań poprzez NPM. I tak, dla porównywanych framework'ów, na koniec czerwca 2020 roku, wyniki przedstawiały się jak poniżej(Rys.5.1):

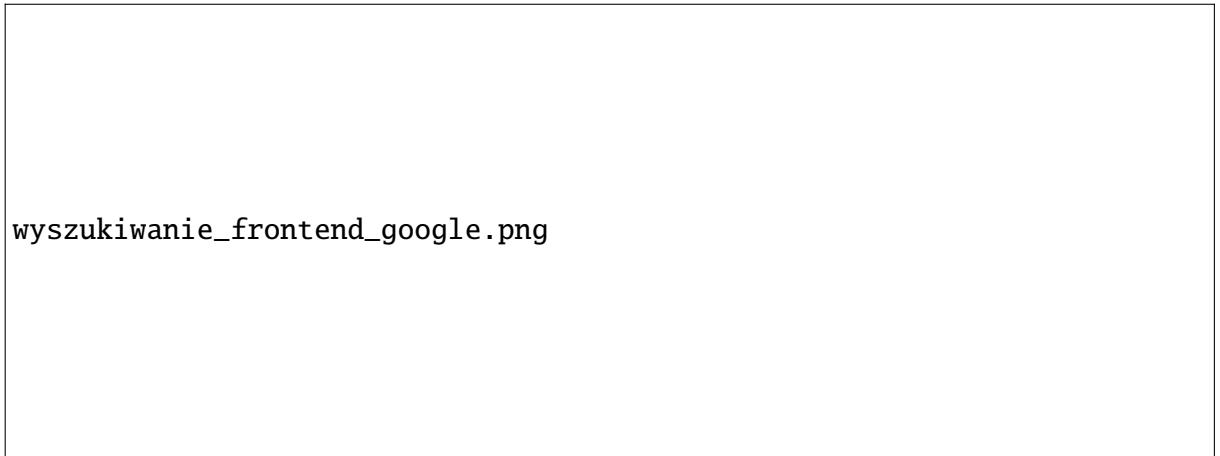


Rysunek 5.1: Stosowanie poszczególnych frameworków front-end w projektach

Tutaj, React wysuwa się na czoło, gdyż cieszy się obecnie największą ilością stworzonych z jego pomocą projektów. Spośród porównywanych framework'ów, zaraz za nim plasuje się Angular i nieco dalej Vue. Svelte, oraz Ember, z racji bardzo niewielkiej używalności, nie zostały ujęte w tym rankingu.

Na koniec, jeśli chodzi o podsumowanie kwestii popularności każdego z rozwiązań, można posłużyć się statystykami wyszukiwania w google.com, za okres ostatnich dwunastu miesięcy.

stu miesięcy, widocznymi na wykresie(Rys.5.2).



Rysunek 5.2: Częstość wyszukiwania poszczególnych frameworków front-end w wyszukiwarce google - 12 miesięcy

W podjęciu ostatecznej decyzji o wyborze docelowego frameworka front-endowego, zdecydowano się posłużyć wielokryterialnym modelem decyzyjnym, w postaci aplikacji Visual Promethee. Po wprowadzeniu do tego programu danych, w postaci kryteriów oceny, dostępnych alternatyw, oraz wag tych kryteriów i wyników przedstawionego powyżej porównania, uzyskano jasną sugestię – Vue.js (Rys.5.3).

Diagram obrazuje nie tylko końcową kolejność opcji, ale także uzasadnia, które z ocenianych cech miały na to wpływ. Ostateczny wybór autora był zgodny z wynikiem uzyskanym z Visual Promethee.

Wynik tego samego porównania, ale w innej interpretacji, przedstawia kolejny wykres, (Rys.5.4).

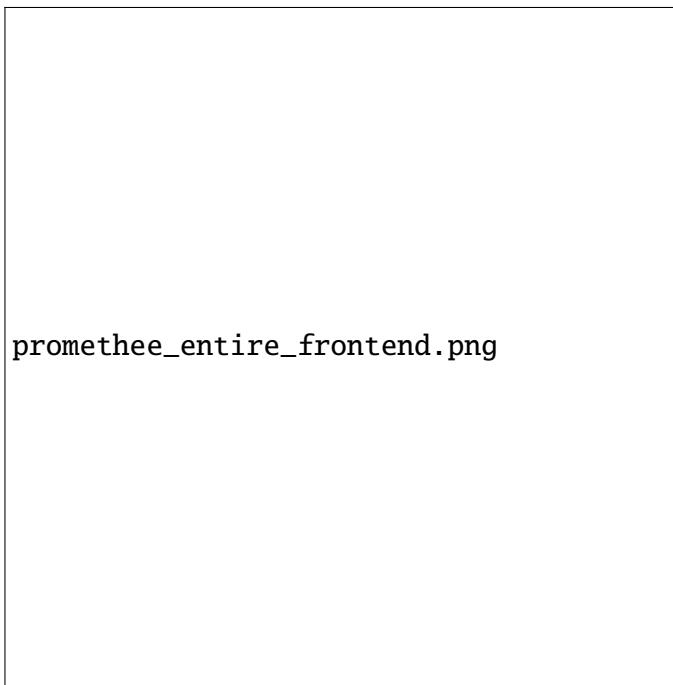
## 5.2 Back-end

Wybór sposobu realizacji części back-end projektu odbył się w odmienny sposób od przedstawionego wyżej procesu, dotyczącego strony wizualnej. Po pierwsze, pomimo bogatego wyboru samych języków programowania, które służą do obsługi części serwerowej aplikacji, takich jak JavaScript, PHP, Ruby, Java, czy Python, autor z góry był zdecydowany zastosować technologię wywodzącą się z języka PHP. Powodem była chęć dalszego rozwoju osobistych umiejętności w tym właśnie kierunku. Dodatkowo, od razu zdecydowano się na zastosowanie gotowego frameworka, a nie bazowanie na czystym PHP, również z podanego wcześniej powodu, jednak tym razem, z braku doświadczenia z tego typu rozwiązaniami, konieczne stało się porównanie i wybór frameworka najbardziej dostosowanego do wymagań projektu i dostępnych ram czasowych jego realizacji.



promethee\_gaia\_frontend.png

Rysunek 5.3: Wynik analizy problemu przez *Visual Promethee* - wykres Gaia



promethee\_entire\_frontend.png

Rysunek 5.4: Wynik analizy problemu przez *Visual Promethee* - wykres typu Promethee Complete Ranking

Do porównania przeznaczono następujące frameworki PHP:

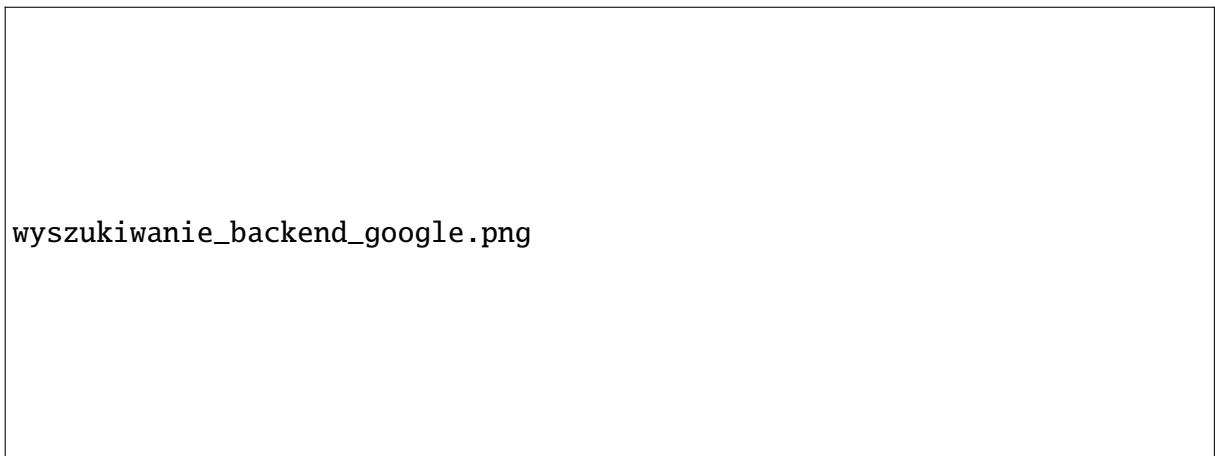
- Laravel
- Symfony
- Symfony
- Codeigniter 4
- Cake

Kryteriami, według których oceniano przedstawione opcje programowe, były:

- popularność i wsparcie społeczności
- szybkość pracy frameworka
- łatwość nauki
- jakość dokumentacji

### **Popularność rozwiązania**

W ocenie popularności, podobnie, jak wyżej, wzięto pod uwagę statystyki wyszukiwań w wyszukiwarce google, za okres ostatnich dwunastu miesięcy(Rys.5.7).



Rysunek 5.5: Częstość wyszukiwania poszczególnych frameworków back-end w wyszukiwarce google - 12 miesięcy

Ponadto kolejnym źródłem statystyk rozpowszechnienia danego rozwiązania wśród programistów, był portal <https://stackshare.io/stackups/codeigniter-vs-laravel-vs-symfony>, kolekcjonujący deklaracje użytkowników, dotyczące stosowanych technologii. Według właśnie tego ostatniego, najwięcej, bo 3,4 tys głosów oddano za Laravelem, ok 1tys głosów za Symfony i tylko 433 użytkowników przyznało się do stosowania Codeignitera. Wynik dla Cake'a wyniósł poniżej 100.

Wyniki te pozostają w rozbieżności w stosunku do liczby współtworzących repozytoria GitHub poszczególnych frameworków, dla przykładu repo Laravel ma jedynie ok 530-tu uczestników, mniej więc niż Symfony, które posiada ok 2200 współtwórców. Wyniki z GitHub-a przedstawia rysunek (Rys.5.6).



Rysunek 5.6: Statystyki portalu GitHub.com, dot. frameworków back-end

Jak już wspomniano w poprzednim dziale, w ślad za popularnością danego rozwiązania idzie wsparcie jakie może zaoferować jego społeczność. Z tego właśnie powodu, właściwość ta jest tak istotna dla nowych użytkowników danej technologii.

### Łatwość instalacji i nauki

Kryterium to, nabrąło w opisywanym projekcie, szczególnego znaczenia, głównie z uwagi na z góry określone, nie zmienialne ramy czasowe jego realizacji. Dlatego właśnie, zarówno w opisany wcześniej procesie wyboru technologii po stronie klienckiej, jak i tu, przy selekcji frameworka do obsługi strony serwerowej, stanowiło ono kluczowe ograniczenie przy wyborze. Autorowi zależało tym samym na ograniczeniu wstępnych czynników ryzyka niepowodzenia projektu.

Jeśli chodzi o prostotę procesu instalowania frameworka i jego konfiguracji, to Codeigniter jest pod tym względem na pierwszym miejscu w niniejszym rankingu. Obok niego plasuje się Cake, zgodnie z opiniami programistów.

Z kolei Symfony, to wymagający framework, zarówno pod względem samej instalacji, jak i pierwszej konfiguracji środowiska. Duża ilość opcji konfigurowalnych stanowi wyraźną przeszkodę w szybkim rozpoczęciu pracy. Niestety, Symfony uznaje się także, w środowisku deweloperów, za wyraźnie trudniejszy do nauczenia niż pozostałe frameworki PHP.

Podobna sytuacja dotyczy Cake'a, również uważanego za wymagającego wysiłku w jego opanowaniu.

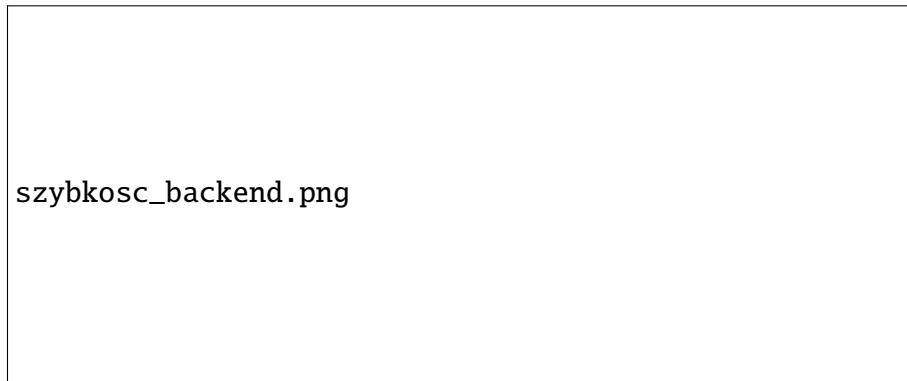
Laravel, podobnie do Symfony, charakteryzuje się stromą krzywą uczenia, co przy opinii technologii opartej na dużej ilości tzw „magicznych” metod, a więc o mocno ukrytej logice działania, nie sprzyja jego łatwemu zrozumieniu przez początkującego użytkownika.

### Szybkość i wydajność

W przypadku tego kryterium, postanowiono nie opierać się na niepoliczalnych odczuciach użytkowników, ale sięgnąć po dane przedstawiane przez TechEmpower Framework Benchmark na portalu <https://www.techempower.com>, będące wynikiem testów wydajnościowych przeprowadzanych na maszynach fizycznych, jak i rozwiązańach chmurowych.

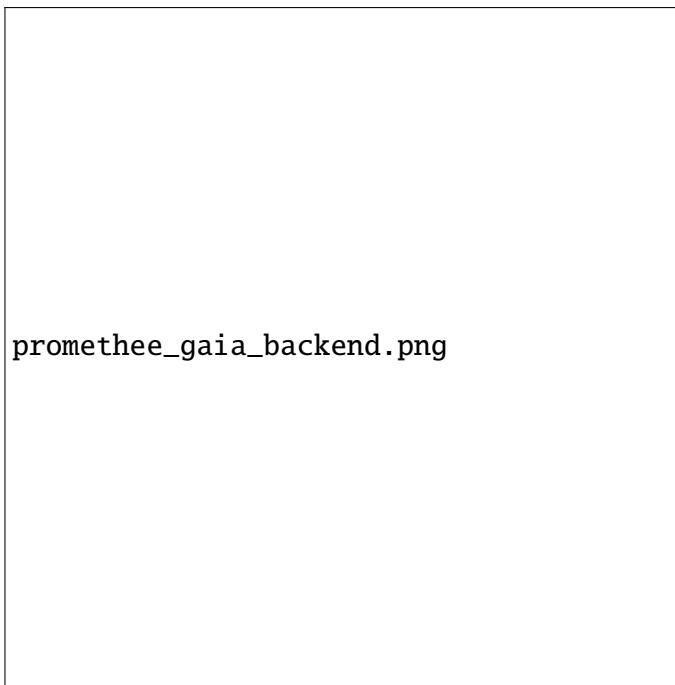
W tabeli (Rys. 5.7) przedstawiono wyniki testów, przeprowadzonych na fizycznych

komputerach, gdzie oceniano typowe dla pracy z bazami danych, operacje, jak przesyłanie zapytań i odpowiedzi, zarówno dla pojedynczych zapytań, jak i seryjnych (po 20 zapytań w jednej serii). Według przedstawionych danych, pierwsze miejsce w rankingu wydajności spośród ocenianych frameworków, zajął Codeigniter, potem kolejno: Symfony, Laravel i wyraźnie odstając od reszty, Cake.

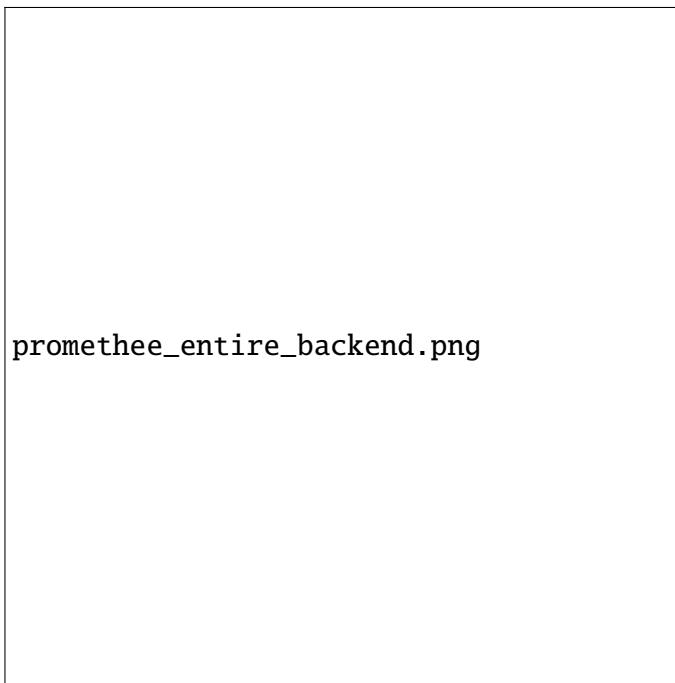


Rysunek 5.7: Wyniki testów wydajnościowych, źródło danych: <https://www.techempower.com>

Ostateczną decyzję, dotyczącą wyboru technologii back-endowej, podjęto po przeanalizowaniu zebranych kryteriów w programie Visual Promethee, będącym jak już wspomniano, implementacją wielokryterialnego modelu decyzyjnego. Wykres (Rys.5.8) przedstawia hierarchię ocenianych frameworków wraz z kryteriami wspierającymi ich wybór. Ewentualne wątpliwości, rozwiewa wykres *Promethee Complete Ranking* (Rys.5.9), na którym przewaga Codeignitera pod względem wziętych pod uwagę cech, jest wyraźnie zobrazowana.



Rysunek 5.8: Wynik analizy problemu przez *Visual Promethee*



Rysunek 5.9: Wynik analizy problemu przez *Visual Promethee* - wykres typu Promethee Complete Ranking

# 6. Implementacja projektu

## 6.1 Opis procesu tworzenia produktu

Implementacji założeń projektowych podjęto się z chwilą decyzji w sprawie wyboru technologii back-end. Wcześniej uległo też zakończeniu przygotowanie wymagań funkcjonalnych całego systemu, które były najważniejszym zbiorem warunków, jakim powinien sprostać gotowy program.

Poznawanie tej technologii odbywało się równolegle do pracy programistycznej nad procesem autoryzacji użytkownika docelowego systemu. Na tym etapie strona kliencka aplikacji istniała tylko jako widoki tworzone w języku html, wywoływanie bezpośrednio z odpowiednich kontrolerów framework. Zgodnie z początkowymi planami projektowymi, obsługa dynamicznych zachowań gotowej strony internetowej, miała opierać się o zastosowanie czystego języka JavaScript, tzw Vanilla JS, co w zamyśle autora, miało pozwolić na łatwiejsze, zatem szybsze prowadzenie prac. Takie założenie, zdecydowało o zastosowaniu mechanizmu sesji do autoryzacji poczyniań użytkownika w aplikacji internetowej Repeater.

Z chwilą zakończenia prac nad systemem rejestracji, logowania i odzyskiwania hasła dostępowego użytkownika, rozpoczęto przygotowanie do projektowania elementów dynamicznych poszczególnych widoków. Pierwsze rozważania teoretyczne, postawiły jednak pod znakiem zapytania możliwość panowania nad organizacją kodu, przy zachowaniu pierwotnego założenia stosowania czystego JavaScript. Stosowane dotychczas w małych projektach laboratoryjnych, podobne rozwiązanie, już wtedy rodziło szereg trudności, głównie polegających na nadmiernym wzroście objętości programów.

Autor w związku z powyższym podjął decyzję o zmianie podejścia projektowego do strony klienckiej aplikacji i oparcia jej tworzenia o gotowy framework, którego wyboru dokonano zgodnie z przyjętymi priorytetami i pomocą platformy wielokryterialnego wspomagania decyzyjnego, *Promethee*. Wybór technologii Vue.js 3, dawał możliwość elastycznego stosowania go w projekcie, począwszy od prostego importowania framework'a z zewnętrznego repozytorium, aż po zastosowanie wzbogaconej o dodatkowe funkcjonalności, wersji instalacyjnej (CLI).

Pomimo atrakcyjnej łatwości pracy z wersją uproszczoną Vue, przewidywania autora, dotyczące konieczności posiadania kontrolera stanu zmiennych po stronie front-end, skłoniły go do użycia pełnej wersji framework'a, posiadającej możliwość użycia techniki Vuex, managera stanu aplikacji.

Konsekwencją dokonanego wyboru, była zmiana sposobu autoryzacji użytkownika po stronie serwera z pierwotnie przygotowanej, sesjnej, na taką, która pozwoliłaby uwierzyć gadniąć stronę klienta, opartą na bezstanowym protokole HTTP, podczas obsługi wielo-

krotnych żądań wysyłanych do bazy danych.

Do tego celu, wybrano metodę JWT (JSON Web Token), która polega na każdorazowym dołączaniu do wysyłanego żądania na serwer, niepowtarzalnego, przygotowanego przez tenże serwer (od razu po zalogowaniu użytkownika) tokena. Na jego podstawie, każde działanie oparte o interakcję z bazą danych, jest uwierzytelnianie osobno. Szczegółowo sposób implementacji tegoż, został opisany w rozdziale poświęconym aspektom bezpieczeństwa projektowanego systemu.

Post factum, można przyznać, że obie główne decyzje dotyczące oparcia implementacji back-end o framework Codeigniter, oraz front-end o Vue CLI, okazały się z perspektywy czasu bardzo dobre, choć związane były z podjęciem dużego wysiłku w celu poznania tych nowych technik.

Pierwsza z nich, ponieważ stopień trudności opanowania Codeigniter'a nie okazał się ryzykownie duży, a funkcjonalności framework'a w pełni pokryły wymagania stawiane mu przez projekt.

Druga, dotycząca wyboru Vue.js do obsługi strony wizualnej, ponieważ tylko dzięki użyciu funkcji oferowanych przez jego pełną wersję CLI, udało się stworzyć uporządkowany kod, realizujący pierwotne założenia reaktywności strony internetowej Repeater.

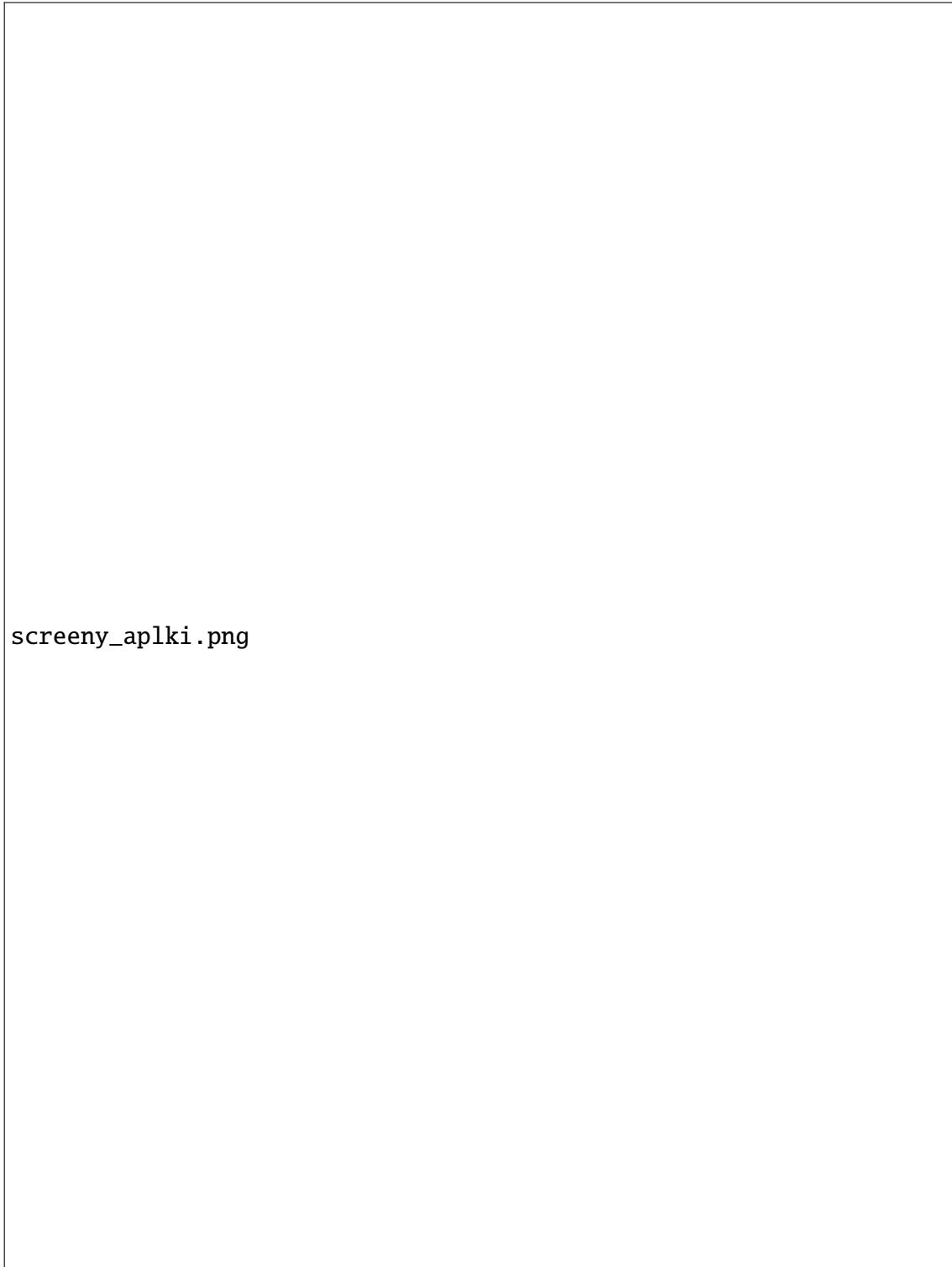
Ostatnim etapem przed zainstalowaniem gotowej platformy na serwerze, było zastosowanie poprawy strony wizualnej elementów poszczególnych widoków – podstron, za pomocą CSS – kaskadowych arkuszy stylów, w celu uatrakcyjnienia pracy z aplikacją i podkreślenia ważności poszczególnych elementów podstron.

Ostatecznie, uzyskano produkt, którego funkcjonalności odpowiadają założeniom przedstawionym w dokumentacji wymagań, zawartej w załączniku A do niniejszej pracy.

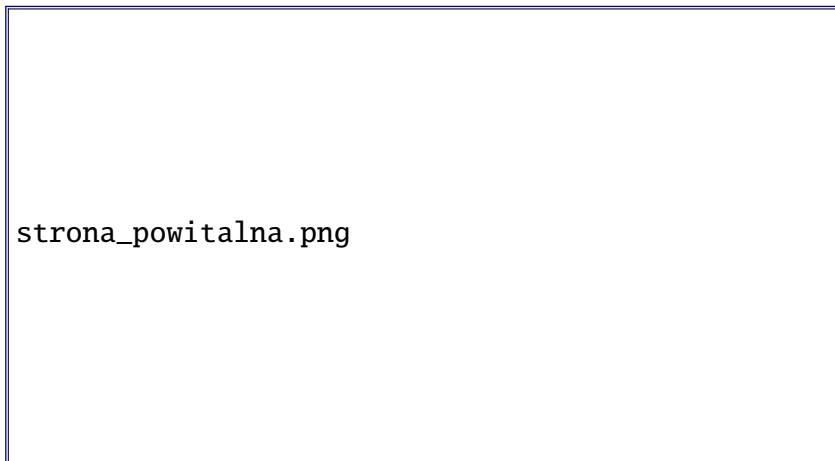
Sumaryczny czas realizacji projektu od strony praktycznej, czyli jego implementacja, wliczając w to również czas poświęcony na poznanie zastosowanych technologii, wyniósł blisko 2,5 miesiąca.

## 6.2 Ekrany gotowej aplikacji

W niniejszym rozdziale, zostaną przedstawione zrzuty ekranów gotowego systemu, osobno dla każdej aktywności użytkownika, której podjęcie umożliwia mu aplikacja. Poniżej, (Rys. 6.17), zamieszczono diagram przejść pomiędzy poszczególnymi ekranami gotowego projektu.



Rysunek 6.1: Schemat przejść pomiędzy ekranami gotowego systemu

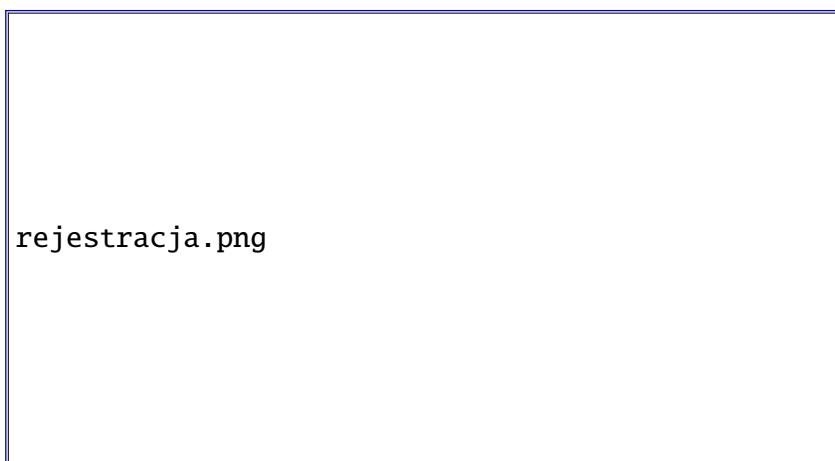


strona\_powitalna.png

Rysunek 6.2: **Ekran nr 1. diagramu** - strona powitalna z opcjami dla zarejestrowanych użytkowników i nie posiadających jeszcze konta

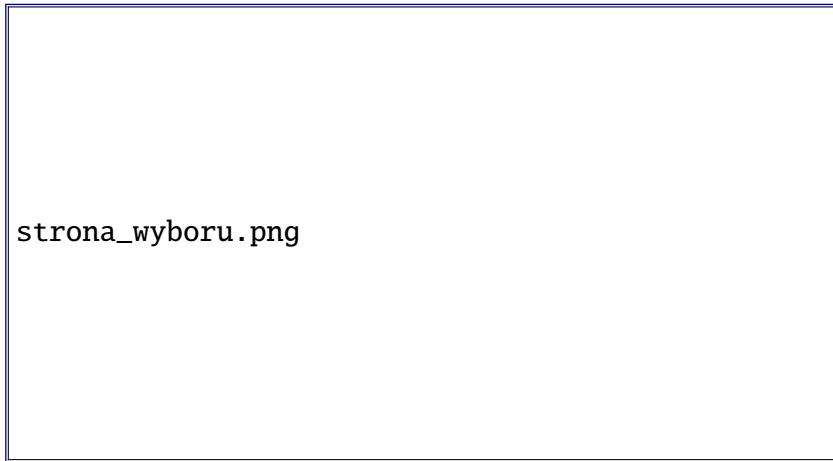
A screenshot of a login form titled "Logowanie". It contains two input fields: "E-mail" and "Hasło", both with placeholder text. Below the fields is an orange "Zaloguj" button. At the bottom are two blue buttons: "Cofnij" on the left and "Zapomniane hasło..." on the right.

Rysunek 6.3: **Ekran nr 2. diagramu** - formularz logowania dla aktywowanych użytkowników

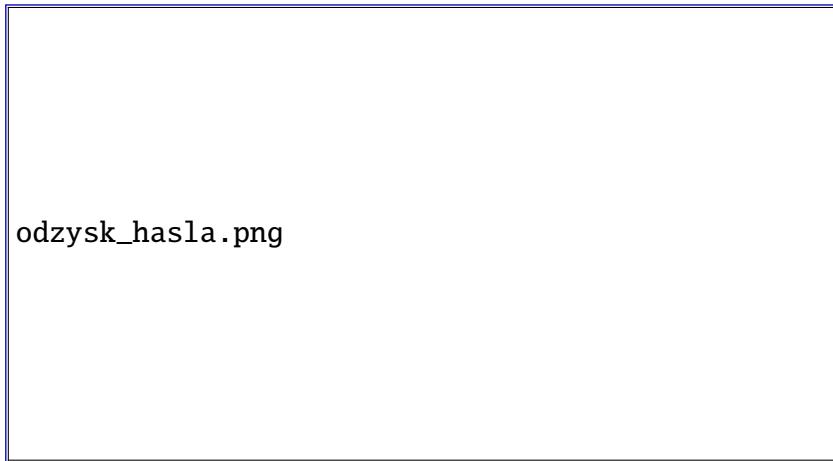


rejestracja.png

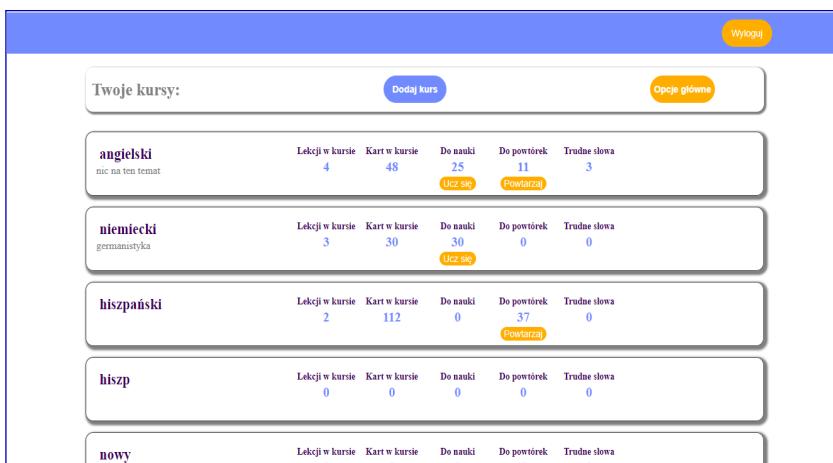
Rysunek 6.4: **Ekran nr 3. diagramu** - formularz rejestracyjny, użytkownik podaje w nim dane konieczne do stworzenia nowego konta



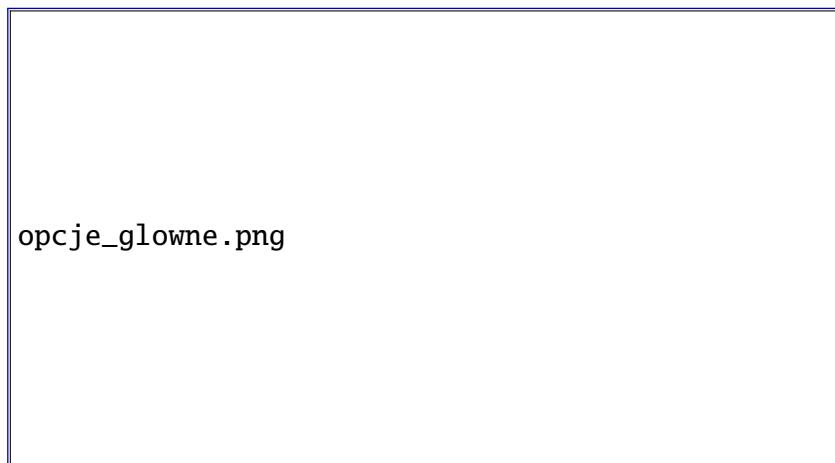
Rysunek 6.5: **Ekran nr 4. diagramu** - możliwość wyboru głównych opcji uruchamiania systemu



Rysunek 6.6: **Ekran nr 5. diagramu** - formularz resetowania hasła przez użytkownika, w przypadku jego zapomnienia

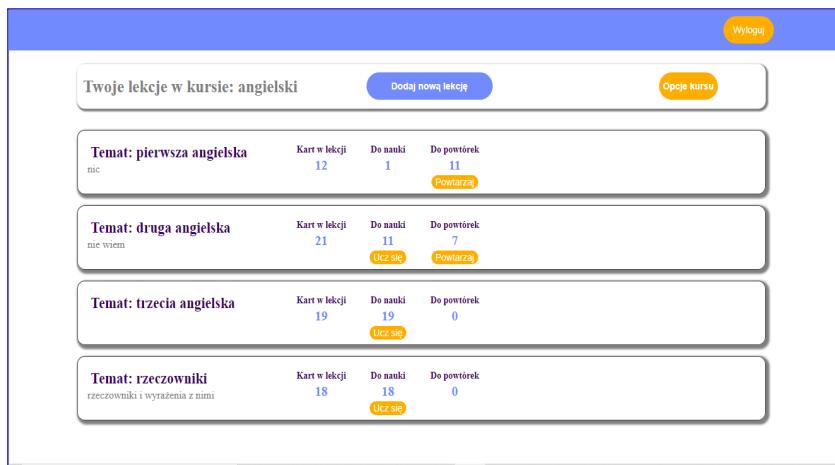


Rysunek 6.7: **Ekran nr 6. diagramu** - główne okno systemu z dostępem do listy kursów założonych przez użytkownika i do edycji opcji aplikacji

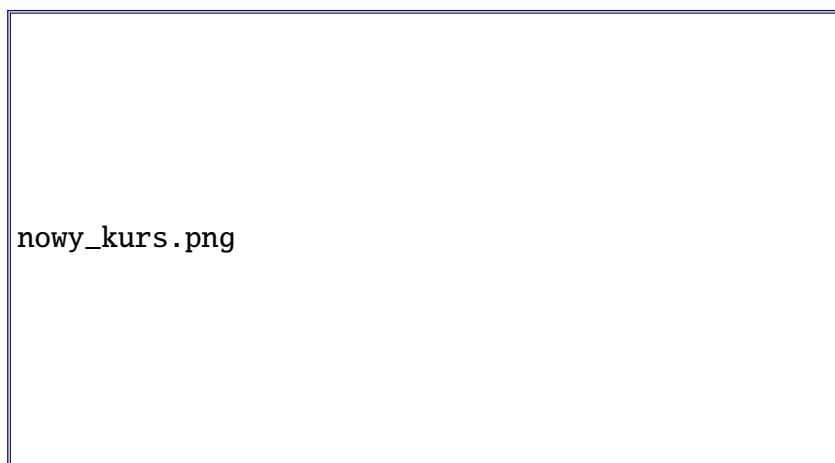


opcje\_glowne.png

Rysunek 6.8: **Ekran nr 7. diagramu** - podstrona umożliwiająca edytowanie ustawień systemu

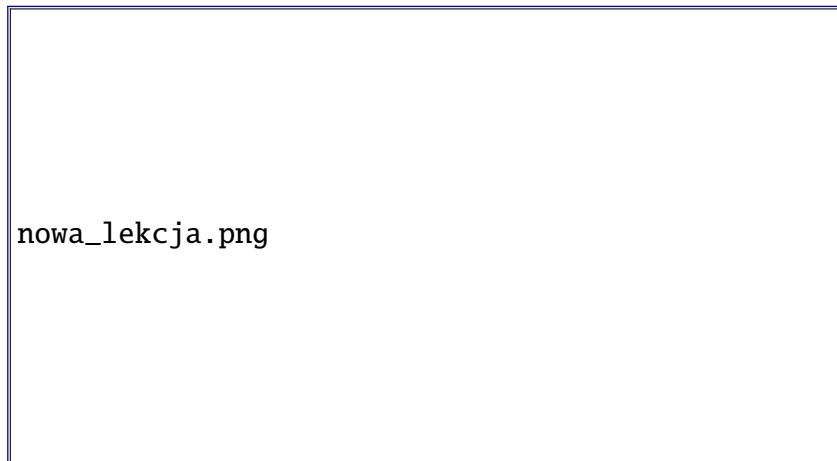


Rysunek 6.9: **Ekran nr 8. diagramu** - widok wnętrza pojedynczego kursu, widoczna jest lista dostępnych lekcji

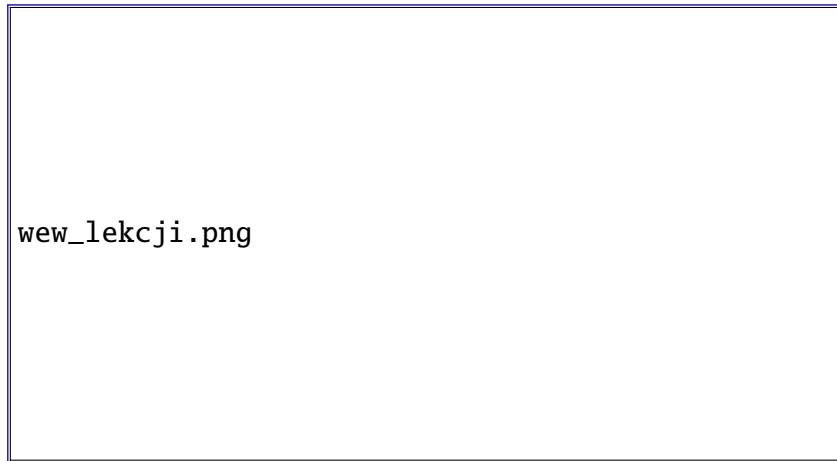


nowy\_kurs.png

Rysunek 6.10: **Ekran nr 9. diagramu** - formularz umożliwiający utworzenie nowego kursu przez użytkownika



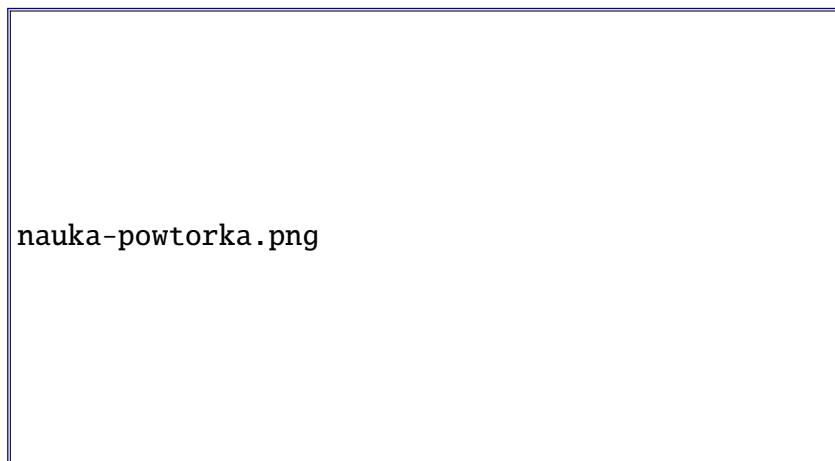
Rysunek 6.11: **Ekran nr 10. diagramu** - formularz tworzenia nowej lekcji w obrębie kursu



Rysunek 6.12: **Ekran nr 11. diagramu** - widok wnętrza pojedynczej lekcji wraz z tabelą przecho-wywanych słów - kart słownikowych

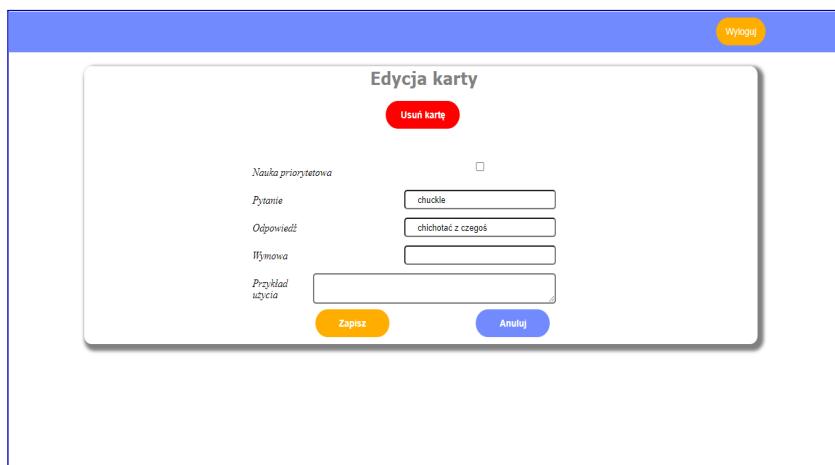


Rysunek 6.13: **Ekran nr 12. diagramu** - pierwszy etap nauki lub powtórki, polegający na przypo-mnieniu przez użytkownika tłumaczenia ukazanego słowa

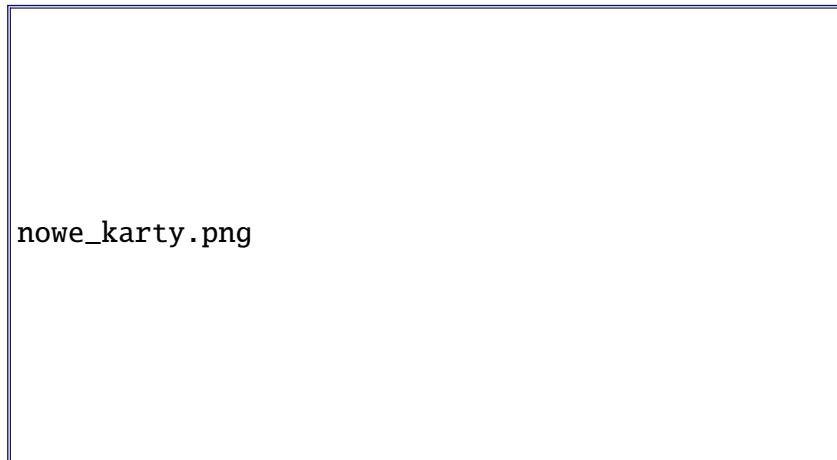


nauka-powtorka.png

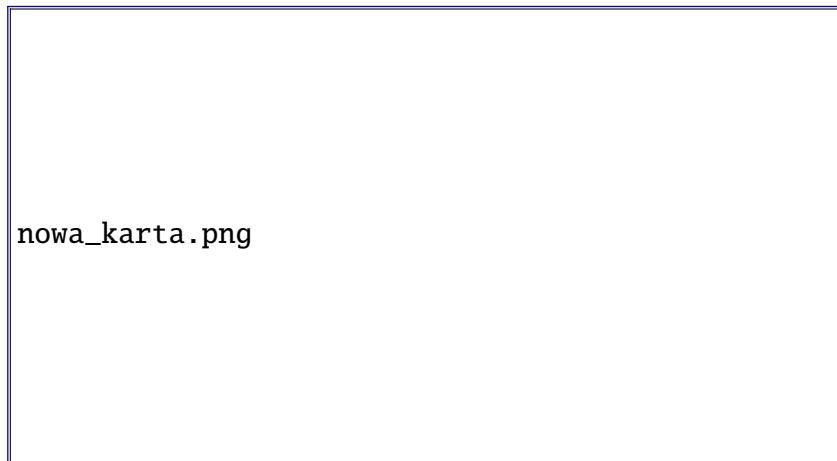
Rysunek 6.14: **Ekran nr 13. diagramu** - sprawdzenie stopnia zapamiętania słowa i dokonanie samooceny - tutaj dodatkowo możliwość oceny słowa jako łatwe - możliwość dostępna tylko w czasie nauki



Rysunek 6.15: **Ekran nr 14. diagramu** - widok formularza edycyjnego karty/słowa, dostępny zarówno z poziomu lekcji, jak i w czasie nauki i powtórek



Rysunek 6.16: **Ekran nr 15. diagramu** - formularz do wprowadzania wielu nowych słów jednocześnie z pliku csv



Rysunek 6.17: **Ekran nr 16. diagramu** - formularz do wprowadzania pojedynczej karty do bazy danych

### 6.3 Sprawdzenie działania systemu w kontakcie z użytkownikiem

W chwili otwarcia aplikacji po raz pierwszy, użytkownik widzi odnośnik do stworzenia osobistego konta. Rejestracja nie wymaga oczekiwania na aktywację konta przez administratora, ponieważ od razu po przesłaniu danych, użytkownika otrzymuje wiadomość mailową z linkiem aktywacyjnym. W momencie jego uruchomienia, konto staje się aktywne i można się logować na nie.

Po zalogowaniu, uczniów wita ekran z opcjami działania, w tym z możliwością natychmiastowego wykonania zaległych powtórek, o ile korzystano już z systemu (Rys. 6.18).

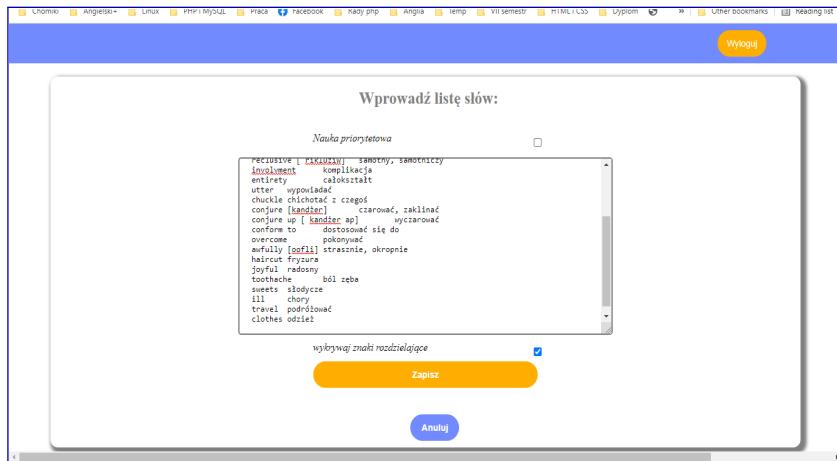


Rysunek 6.18: ekran wyboru pierwszych aktywności

Nowym użytkownikom, pozostaje przejście do głównego okna aplikacji, gdzie powinni od razu stworzyć co najmniej jeden magazyn na swoje karty ze słowami, czyli kurs. Kurs jest zwykle poświęcony jednemu odrębnemu rodzajowi zasobów, np przeznaczony powinien być dla jednego tylko języka obcego. Przy okazji, w oknie głównym, uczeń ma możliwość dostosowania do siebie, warunków nauki, jak limitu powtórek, które ma możliwość wykonać w ciągu jednego dnia, określić ilość słów, których jest w stanie się uczyć jednorazowo, czy z bardziej zaawansowanych ustawień, może zmienić kierunek odpytywanie przez aplikację.

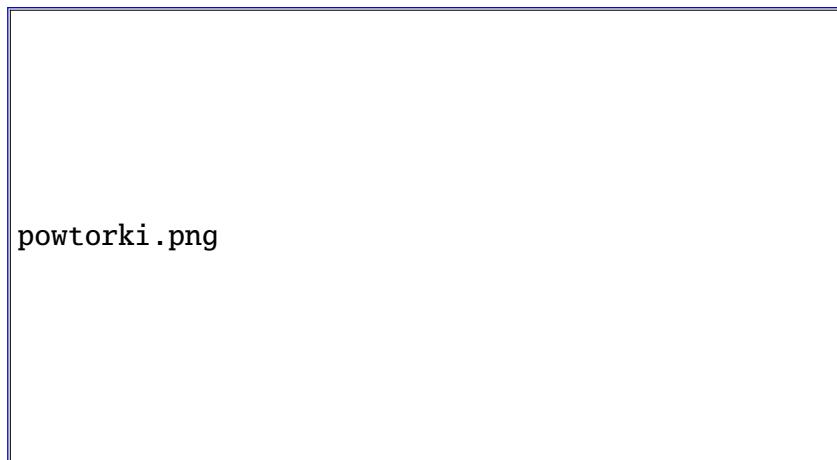
Po założeniu chociaż jednego kursu, użytkownik otwiera go aby wewnątrz magazynować swoje fiszki. Aby zachować porządek w przechowywanych informacjach, powinien on zapisywać je w lekcjach, których tematy sam określa, w ramach głównego przedmioty jakim poświęcił dany kurs. Lekcja może przechowywać praktycznie nieograniczoną ilość kart - fiszek. Ich wprowadzanie do systemu odbywać się może dwójako: albo można je wpisywać pojedynczo, albo od razu przesyłać systemowi wiele kart, w postaci listy sformatowanej w każdym popularnym arkuszu kalkulacyjnym (Rys. 6.19).

Do każdego słowa, wraz z jego tłumaczeniem na obcy język, aplikacja umożliwia dołączenie dodatkowych przydatnych informacji, np zapisu wymowy, lub przykład użycia w zdaniu. Ewentualne błędy w pisowni, a o takie nietrudno, gdy się przygotowuje samemu swoje karty, można natychmiast, nie przerywając nauki, naprawić, włączając edycję bieżącej karty/fiszki.



Rysunek 6.19: ekran wprowadzania wielu kart

Mając materiał do nauki, można ją rozpocząć z każdego prawie okna aplikacji. Można to zrobić z okna głównego z listą kursów, okna kursu z listą lekcji, lub z konkretnej lekcji. Wszędzie tam, system udostępnia użytkownikowi informację o ilości słów oczekujących na nauczenie, powtórzenie, lub liczbie wszystkich posiadanych kart (Rys. 6.20).



Rysunek 6.20: ekran wprowadzania wielu kart

Proces nauki, polega na wybraniu przez system partii słów wg określonych w opcjach aplikacji limitów i pokazywaniu ich na ekranie w kilku cyklach, aż uczeń potwierdzi opanowanie wszystkich z nich. Uczeń, pytany o znajomość danego słowa, widzi początkowo tylko jego odpowiednik w ojczystym języku (Rys. 6.21) i dopiero po potwierdzeniu gotowości do konfrontacji, pokazywane jest mu tłumaczenie hasła (Rys. ??).

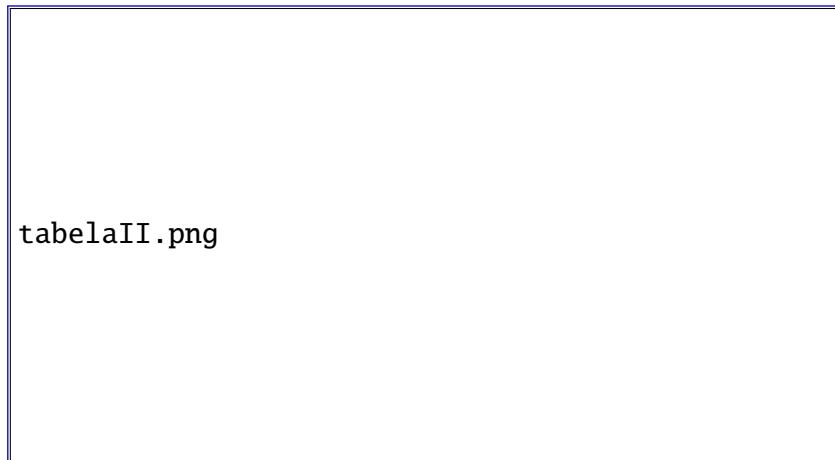
Po zapamiętaniu wszystkich słów w partii, system powtarza prezentację tych kart, których opanowanie trwało najdłużej. Aplikacja Repeater wymaga od ucznia, w krótkim czasie po tej nauce, poddania się odpytaniu ze świeżo nauczonego materiału. Poprawia to w znacznym stopniu efekty nauki. Samo odpytanie, czy raczej powtórka, nie różni się zasadniczo od etapu nauki, bo podobnie jak przedtem, sam ocenia swoją znajomość słowa. Według tej oceny, wyznaczany jest kolejny termin powtórki, lub słowo wraca do



Rysunek 6.21: ekran początkowego etapu nauki

puli kart przeznaczonych do nauki. Terminy kolejnych powtórek są coraz dłuższe dla danego słowa, przez co coraz dłużej także, pozostają w pamięci ucznia.

Uczeń na każdym etapie nauki, ma możliwość edycji posiadanych fiszek, jak również ich archiwizacji poza systemem Repeater. Ma również możliwość przeglądania ich w oknie lekcji, gdzie widoczne są w postaci tabeli, w której można je wyszukiwać i sortować, a także edytować i usuwać(Rys. 6.22).



Rysunek 6.22: ekran lekcji z widokiem tabeli słów/fiszek

System Repeater projektowany był z myślą o komforcie nauki, oraz łatwości rozpozęcia z nim pracy. Wszystkie opcje są łatwo dostępne a obsługa od samego początku nie nastręcza trudności nowym użytkownikom. Samo działanie systemu oparte jest o wielokrotnie udokumentowaną teorię rozłożonych w czasie powtórek przy nauce pamięciowej. Budowa aplikacji nie wyklucza również możliwości zastosowania jej do wspomagania nauki pamięciowej również innych, niż słowa obcojęzyczne, treści. Wystarczy, aby spełniały odpowiadały schematowi pytanie - odpowiedź. Możliwość wyboru kierunku nauki i testowania, jest w tym wypadku bardzo pomocna.

# 7. Podsumowanie

## 7.1 Ocena stanu końcowego

Praca niniejsza, jest dowodem na zasadność podejmowania dalszych prób tworzenia narzędzi, wspomagających naukę pamięciową, opartą o wykorzystanie zalet powtórek rozłożonych w czasie. Pomimo istnienia szeregu podobnych, gdy chodzi o ogólną zasadę działania, rozwiązań, nowe badania nad pamięcią dostarczają kolejnych argumentów do podejmowania nowych eksperymentów, mogących poprawić skuteczność wysiłków ucznia.

Tutaj akurat, zdecydowano się na przesunięcie nacisku z dalszego zagłębiania się w rozwój algorytmu obliczania terminów powtórek, na korzyść lepszego początkowego zakotwiczenia informacji w pamięci użytkownika. Zastosowano do tego celu podsumowywanie każdej porcji nauki, opcjonalne przeuczenie, oraz dwie kolejne, obligatoryjne fazy powtórkowe, w krótkim odstępie czasu. Dodatkowo, wprowadzanie możliwości odpytywania ucznia w odwrotnym układzie karty, uzależniono od stopnia opanowania tejże, a nie jak to miało miejsce w innym programie, bezwarunkowo.

Wszystkie te funkcjonalności, stanowią wyróżnik projektu, gdyż w większości, żadna z nich nie znalazła zastosowania w rynkowych programach. Obiecujący wynik pierwszego testu, przeprowadzonego na niezaangażowanym w projekt użytkowniku, pozwala sądzić, że obrana strategia nie powinna ulec zasadniczym zmianom w toku dalszych prac rozwojowych.

W toku projektowania i implementacji, osiągnięto więc zasadniczy cel, jakim było przygotowanie aplikacji wspierającej zapamiętywanie słów obcego języka. Udało się zrealizować zaplanowane w projekcie funkcjonalności, a sam produkt gotowy jest do używania przez końcowego użytkownika.

Przy okazji, niejako, prace nad projektem dały możliwość przetestowania użyteczności dla podobnych zastosowań, kilku nowych dla autora, technologii programistycznych. Mowa tu o dwóch frameworkach, Vue.js, oraz Codeigniter, oraz pracy z systemem uwierzytelniania, opartym o JWT.

Z uwagi jednak na to, że efekty pracy z programem w dużej mierze zależą od jego interakcji z użytkownikiem, należy liczyć się z koniecznością około 3-miesięcznego testowania aplikacji na różnorodnej grupie 6-8 użytkowników. Miałoby to na celu zidentyfikowanie możliwych utrudnień w sferze UX, czego nie można wykluczyć, zważywszy jednoosobowe autorstwo projektu, skutkujące zazwyczaj, dopasowaniem wyglądu aplikacji do osobistych upodobań projektanta.

Komfort pracy z programem, jest więc w pewnym stopniu niewiadomą na ten moment życia aplikacji, należy zatem dołożyć starań, aby możliwie szybko rozpoznać ewentualne

braki w tej sferze, aby nie stały na przeszkodzie do skutecznej nauki. Brak przyjemności pracy z dowolnym systemem, skłania do unikania zaangażowania i wpływa na negatywną jego ocenę przez użytkownika.

## 7.2 Możliwości dalszego rozwoju

Zaprojektowany program, daje duże możliwości rozwoju w przyszłości, ponieważ już na etapie projektowania, zwłaszcza bazy danych, dostosowano go do zbierania wielu dodatkowych danych w czasie interakcji z użytkownikiem. Będą one potem źródłem informacji dla takich planowanych funkcjonalności, jak możliwość tworzenia ról dla użytkowników, co pozwoli tworzyć całe klasy szkolne, którym materiał lekcyjny będą zapewniać ich nauczyciele.

Również ewentualne poprawki dotyczące sposobu wyliczania powtórek, będą mogły bazować na statystykach, przygotowanych z tych nadmiarowych w tej chwili danych.

przydatną funkcjonalnością z punktu widzenia autora projektu, byłoby również dołączanie do każdej karty/słowa, lektora, dysponującego poprawną wymową danego związku frazeologicznego. Ułatwi to pracę z trudnymi do wymówienia słowami.

## Bibliografia

- [1] S. Hunston, G.Francis, E.Manning *Grammar and vocabulary: showing the connections* ELT Journal, Volume 51, Issue 3, July 1997, Pages 208–216
- [2] Wilkins ,D.A. *Linguistics in Language Teaching* Australia: Edward Arnold.(1972)
- [3] E.H.hiebert, M. L. Kamil *Teaching and learning vocabulary* Lawrence Erlbaum Associates Publishers, Mahwah New Jersey 2005
- [4] E. T. Rolls *Memory systems in the brain* Uniwersity of oxford, Department of Experimental Psychology, Annual Reviews Psychol. 2000.
- [5] JMJ Murre, J. Dros *Replication and Analysis of Ebbinghaus' Forgetting Curve* Published online 2015 Jul 6. doi: 10.1371/journal.pone.0120644
- [6] Maria Jagodzińska *Psychologia pamięci: Badania, terie, zastosowania* Sensus (2012)
- [7] Górska, T., Grabowska, A., Zagrodzka, J.(red.) *Mózg a zachowania*.495 Warszawa:Wydawnictwo Naukowe PWN (1997)
- [8] Squire, L.R., Kandel, E.R. *From mind to molecules*New York: Scientific American Library.(2000)
- [9] Balochowicz, C., Fisher, P. *Teaching Vocabulary* Manhwah, NJ: Erlbaum (2000)
- [10] Richard J. Gerrig, Philip G. Zimbardo *Psychologia i życie* Wydawnictwo Naukowe PWN, Warszawa 2009, s. 8
- [11] Kintsch, W. *Pamięć i poznanie* Gedächtnis und Kognition. Berlin: Springer-Verlag (1982)
- [12] Dempster, F.N., *Distributing and managing the conditions of encoding and practice* W: E.L. Bjork, R.A. Bjork (red.), Memory (s. 318 – 344). San Diego: Academic Press (1996).
- [13] Włodarski, Z. *Z tajemnic ludzkiej pamięci*Wyd. II. Warszawa: WSiP, s.309, (1990)
- [14] Bahrick, H.P. *Semantic memory content in permastore: 50 years of memory for Spanish learned in school* Journal of Experimental Psychology: General, 113, 1 – 29 (1984).

- [15] Hazenberg, S., Hulstijn J. H. *Defining a minimal receptive second-language vocabulary for non-native university students: An empirical investigation* Applied Linguistics, 17, 145–163 (1996)
- [16] Laufer, B. *What percentage of text-lexis is essential for comprehension?* In C. Laurén M. Nordman (Eds.), *Special language: From humans thinking to thinking machines* (pp. 316-323) Clevedon: Multilingual Matters.
- [17] Darrell Wilkinson, *EFL Vocabulary Acquisition through Word Cards: Student Perceptions and Strategies*, Teaching english as a second or foreign language, nr3, vol21(2017)
- [18] Joe Barcroft, *Can writing a new word detract from learning it? More negative effects of forced output during vocabulary learning* Second Language Research 2006
- [19] Yihsiang Kuo, *Effects of Word Card Strategy versus Word List Strategy on Taiwanese EFL Junior High School Students' Vocabulary Retention* Electronic Journal of Foreign Language Teaching 2012, Vol. 9, No. 1, pp. 26–45
- [20] Tatsuya Nakata, *English vocabulary learning with word lists, word cards and computers: implications from cognitive psychology research for optimal spaced learning* European Association for Computer Assisted Language Learning 2008
- [21] Mondria J-A, Mondria de Vries S., *Efficiently memorizing words with the help of word cards and “hand computer”: Theory and applications* System Elsevier 1994
- [22] H. Pashler, D.Rohrer *Enhancing learning and retarding forgetting: Choices and consequences* Psychonomic Bulletin Review 2008
- [23] *The Effects of Overlearning and Distributed Practise on the Retention of Mathematics Knowledge* Applied Cognitive Psychology 20: 1209–1224 (2006)
- [24] *Effect of Overlearning on Retention* Journal of Applied Psychology Vol.77 nr 5, 1992
- [25] P. Pimsleur *A memory schedule* Modern Language Journal 51(2):73–75 1967
- [26] H. Ebbinghaus *Memory - A Contribution to Experimental Psychology* Teachers College, Columbia University, New York, NY, USA 1885
- [27] B. Settles, B. Meeder *A Trainable Spaced Repetition Model for Language Learning* Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1848–1858, Berlin, Germany, August 7-12, 2016
- [28] S. Leitner *So lernt man lernen. Angewandte Lernpsychologie – ein Weg zum Erfolg* Verlag Herder, Freiburg im Breisgau, Germany 1972
- [29] P.Wozniak, E.J. Gorzelanczyk *Optimization of repetition spacing in the practice of learning* Acta Neurobiologiae Experimentalis, 54:59-62, 1994

- [30] Trabasso and Bower *Attention in learning: Theory and research* New York: Wiley, s.46 (1968)
- [31] Hermann von Ebbinghaus (1912)
- [32] Niewiadomska, G. (1997) *W poszukiwaniu molekularnych mechanizmów pamięci* W: T. Górska, A. Grabowska, J. Zagrodzka (red.), Mózg a zachowanie(s. 269 – 297). Warszawa: Wydawnictwo Naukowe PWN.(1997)

## A. Dodatek

# Specyfikacja wymagań



# Spis treści

<b>1. Wprowadzenie</b>	5
1.1. Cel dokumentu	5
1.2. Zakres produktu	5
1.3. Definicje i synonimy	5
1.4. Zawartość dokumentu	6
<b>2. Opis ogólny</b>	7
2.1. Środowisko programistyczne	7
2.1.1. Interfejsy systemowe	7
2.1.2. Interfejsy użytkownika	7
2.1.3. Interfejsy programowe	7
2.1.4. Interfejsy komunikacyjne	8
2.2. Funkcjonalności produktu	8
2.2.1. Tworzenie konta użytkownika	9
2.2.2. Logowanie użytkownika do konta	10
2.2.3. Nauka kart	11
2.2.4. Przeprowadzanie powtórek	12
2.2.5. Edycja danych	13
2.2.6. Dostosowywanie ustawień programu	14
2.2.7. Eksport i import danych	15
2.3. Charakterystyka użytkownika	15
2.4. Ograniczenia	16
<b>3. Wymagania szczegółowe</b>	17
3.1. Wymagania funkcjonalne	17
3.1.1. Tworzenie konta użytkownika	17
3.1.2. Logowanie użytkownika do konta	19
3.1.3. Nauka kart	20
3.1.4. Przeprowadzanie powtórek	22
3.1.5. Edycja danych	23
3.1.6. Dostosowywanie ustawień programu	27
3.1.7. Eksport i import danych	29



# Rozdział 1

## Wprowadzenie

### 1.1. Cel dokumentu

Celem przygotowania niniejszego dokumentu, jest dostarczenie dokładnego opisu aplikacji internetowej **Repeater**, oraz przedstawienie jej funkcjonalności, oraz interfejsów, zarówno użytkownika, jak i systemowych, oraz sposobów w jakich system będzie reagował na aktywności pochodzące ze środowiska zewnętrznego. Dokument ten, przygotowano z myślą powrotu do prac nad systemem przez jego autora w późniejszym czasie, oraz innych programistach, którzy w toku prac rozwojowych nad niniejszym projektem, będą potrzebowali specyfikacji jego założeń.

### 1.2. Zakres produktu

System o nazwie **Repeater**, będzie aplikacją uruchamianą w przeglądarkach internetowych z aktywnym połączeniem z siecią, do wspomagania i organizowania nauki i powtórek obcojęzycznego słownictwa.

Założeniem jego powstania była chęć zminimalizowania ilości porażek przy nauce pamięciowej, wynikających z niemożności doboru odpowiedniego terminu powtórek zapamiętywanego materiału.

System Repeater będzie pełnił rolę platformy do nauki pamięciowej, poprzez prezentowanie słów obcojęzycznych, w celu zapamiętania ich przez użytkownika.

Program ten, poprzez wykorzystanie algorytmów obliczeniowych, będzie także miał za zadanie przygotowywać i przedstawiać użytkownikowi w formie graficznej na ekranie, partie materiału, którego przypomnienie jest wymagane w danym momencie.

Wśród funkcji tego systemu, będzie zawarta także możliwość wprowadzania i korzystania z zasobów własnych użytkownika, oraz ich edycja.

Dzięki swoim funkcjonalnościom, aplikacja **Repeater**, umożliwi intensyfikację nauki obcego języka, oraz przyczyni się do poprawy umiejętności komunikacyjnych użytkownika w tym języku, czemu wydatnie sprzyja poszerzanie zasobu słownictwa operatywnego.

### 1.3. Definicje i synonimy

**Definicje i synonimy** – synonimy pojęcia produktu, będącego tematem niniejszego opracowania

**Użytkownik, uczeń** – osoba wykorzystująca system do nauki pamięciowej słownictwa

**Baza danych** – wszelkie dane, wykorzystywane w pracy systemu, przechowywane w postaci binarnej na serwerze

**Pytanie** – słowo, lub wyrażenie w języku ojczystym użytkownika, którego tłumaczenie na język obcy powinno być zapamiętane **Odpowiedź** – tłumaczenie na język obcy słowa, lub wyrażenia w języku ojczystym użytkownika

**Rekord słownika** – wiersz tabeli zawierającej dane dotyczące uczonego słowa: między innymi pytanie i odpowiedź

**Karta** – rekord słownika, prezentowany użytkownikowi na ekranie w zakresie połowicznym – jedynie pytanie, lub całościowym – pytanie i odpowiedź jednocześnie

**Nauka** - pierwsze zapoznanie się użytkownika z kartą, mające na celu osiągnięcie stanu zapamiętania tej karty w toku kilkukrotnie powtarzanej prezentacji jej użytkownikowi

**Powtórka** - prezentacja kart, które w czasie fazy nauki, zostały oznaczone przez użytkownika, jako zapamiętane; potwierdzenie znajomości karty w tej fazie, kończy się wyznaczeniem nowej daty powtórki, w odstępie dłuższym niż poprzedni

**Przeuczenie** - dodatkowy, opcjonalny etap utrwalenia w pamięci karty zapamiętanej w fazie nauki, polegający na kilkukrotnym powtórzeniu prezentacji karty, już po fakcie stwierdzenia jej zapamiętania przez użytkownika

**Podsumowanie** - dodatkowy etap utrwalenia w pamięci kart zapamiętanych w bieżącej fazie nauki, polegający na jednokrotnej prezentacji każdej z nich po zakończeniu fazy nauki

## 1.4. Zawartość dokumentu

W kolejnym rozdziale, zatytułowanym *Opis ogólny*, przedstawione zostały główne funkcjonalności systemu, oraz wymagania techniczne niezbędne do prawidłowej pracy produktu. Część ta zawiera poza tym opis interfejsów systemu.

Dokładne zapoznanie się ze wszystkimi funkcjonalnościami, a także odpowiednimi dla nich diagramami, ukazującymi logiczne zależności, umożliwia kolejny rozdział, *Specyfikacja wymagań*.

## Rozdział 2

# Opis ogólny

### 2.1. Środowisko programistyczne

#### 2.1.1. Interfejsy systemowe

System działa zarówno po stronie użytkownika, przekazując mu i odbierając od niego informacje, wyświetlane na ekranie urządzenia, jak i po stronie serwerowej, gdzie znajduje się źródło kodu, obsługującego komunikację z klientem/przeglądarką, oraz baza danych.

Aplikacja może być uruchamiana w przeglądarkach internetowych Chrome (od wersji 63+), Firefox(wersja 40+), Safari (7+), oraz Opera.

Praca z systemem może się odbywać na urządzeniach stacjonarnych typu desktop, posiadających system operacyjny Windows (Vista lub nowszy), Linux, macOS, oraz Android.

#### 2.1.2. Interfejsy użytkownika

System umożliwia interakcję z użytkownikiem poprzez udostępnienie pól wprowadzania danych oraz łączenia plików, przycisków, rozwijanych menu, opcji umieszczonych w przyciskach w panelu głównym, a korzystanie z nich odbywać się może za pomocą klawiatury lub myszki.

#### 2.1.3. Interfejsy programowe

- Oprogramowanie front-end: Vue.js3
- Oprogramowanie back-end: PHP Codeigniter 4
- Baza danych: MySQL
- Architektura systemu - model MVC – Model, Widok, Kontroler

System umożliwia przesyłanie do bazy danych na serwerze, list danych, przygotowanych w formacie CSV.

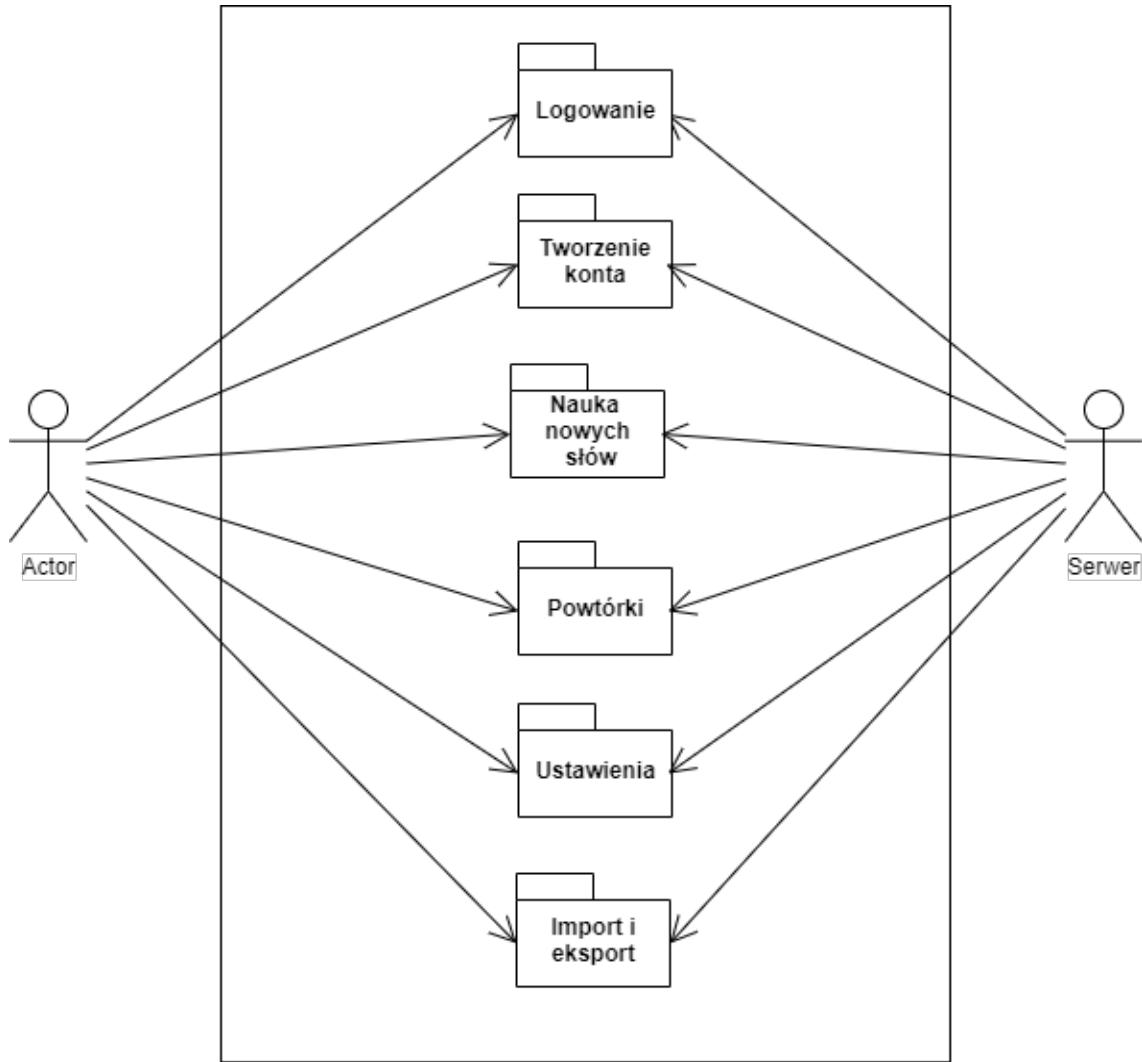
System pozwala na przesyłanie do użytkownika danych w formacie TXT.

Aplikacja operuje na danych, przechowywanych w formacie JSON, dla sprawnego przesyłania ich i odbierania w komunikacji klient – serwer.

### 2.1.4. Interfejsy komunikacyjne

System komunikuje ze sobą stronę klienta oraz stronę serwera, za pomocą technologii AJAX, komunikacji asynchronicznej, realizowanej przy pomocy biblioteki Axios.js.

## 2.2. Funkcjonalności produktu

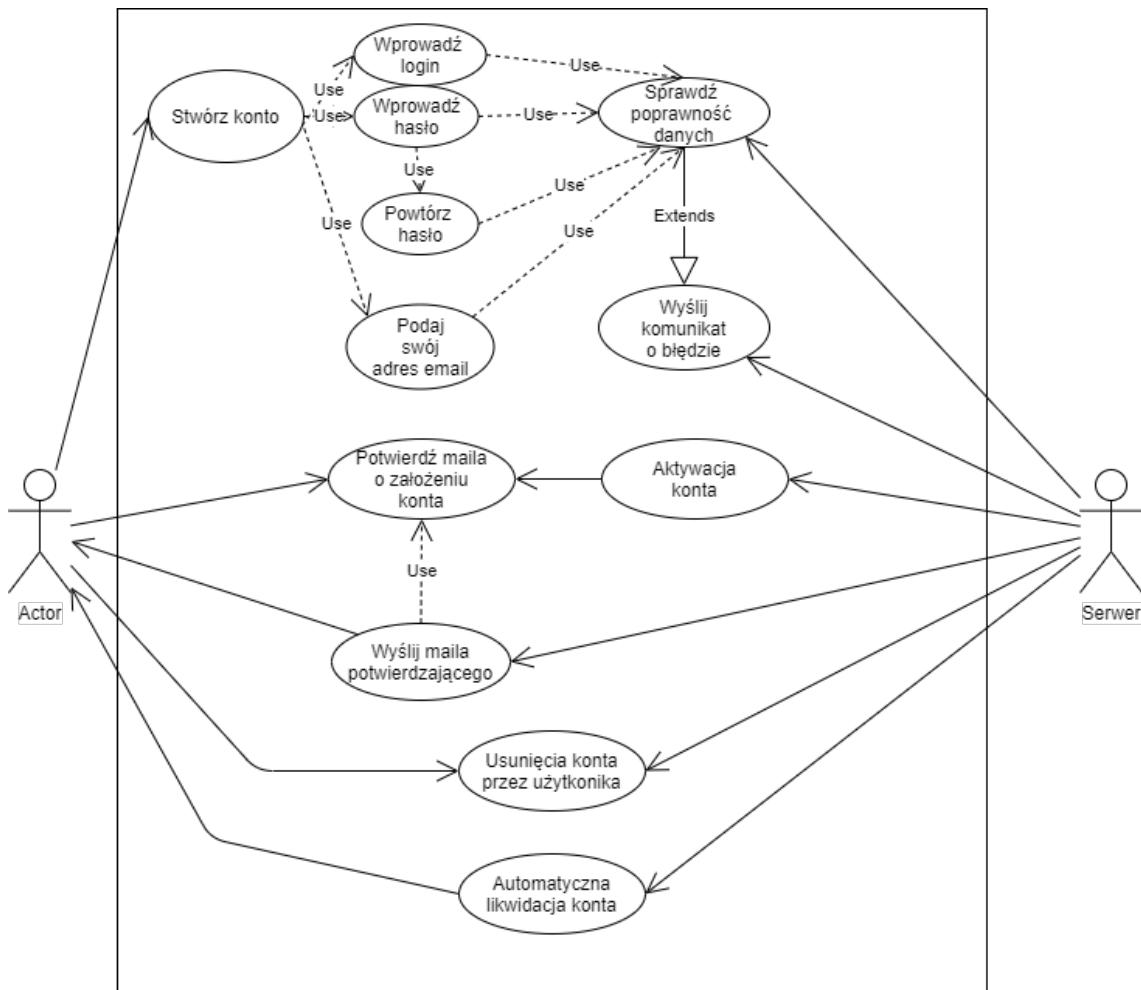


Rysunek 2.1: Diagram ogólnego podziału funkcjonalności produktu

System Repeater, posiada jednego aktora, którym jest użytkownik programu, zwany też uczniem, oraz obsługujący jego aktywności serwer z zainstalowaną na nim bazą danych.

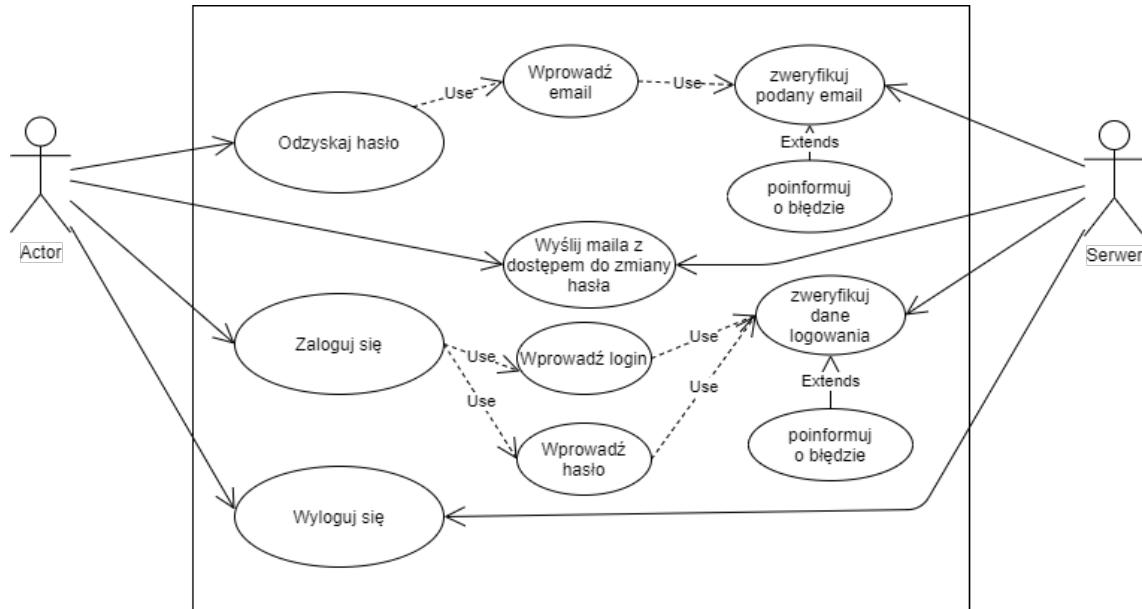
Przedstawione na diagramie (Rys.2.1) główne grupy funkcji programu, zostały rozdzielone i opisane poniżej. Dokładne specyfikacje poszczególnych funkcjonalności, są umieszczone w rozdziale 3.1, w porządku zgodnym z nadanymi im tutaj numerami identyfikującymi.

## 2.2.1. Tworzenie konta użytkownika



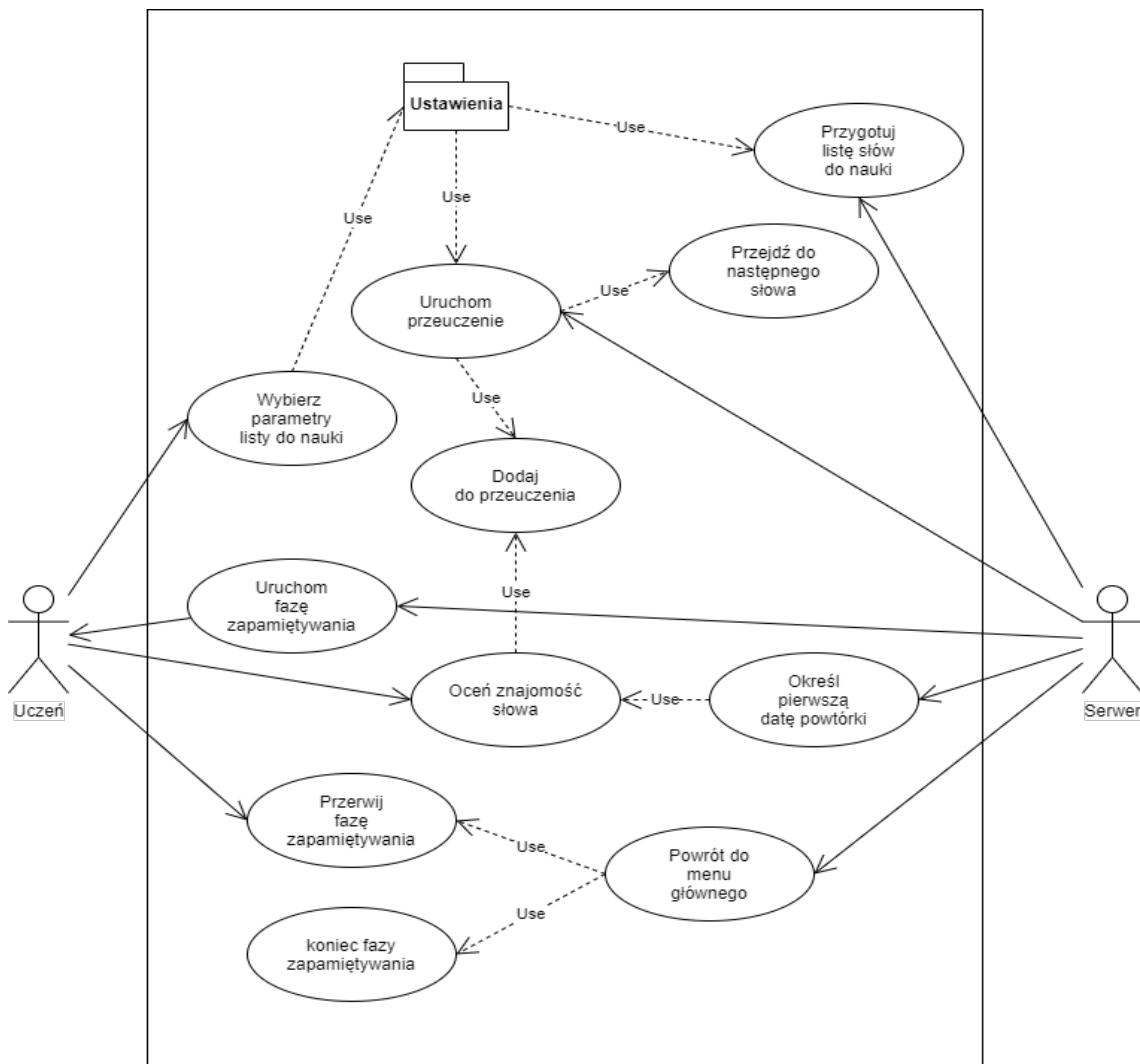
1. System powinien umożliwiać użytkownikowi stworzenie przez niego konta w systemie poprzez podanie danych rejestracyjnych w formularzu - szczegółowy opis w sekcji: 3.1.1.1
2. System powinien umożliwiać użytkownikowi aktywacją dostępu do podanego w formularzu rejestracyjnym konta mailowego poprzez odpowiedź na przesłany email z systemu - szczegółowy opis w sekcji: 3.1.1.2
3. Likwidacja konta przez użytkownika System powinien umożliwiać likwidację konta przez użytkownika - szczegółowy opis w sekcji: 3.1.1.3
4. System powinien umożliwiać zamknięcie konta użytkownika i usuwanie jego zasobów po upływie 6-ciu miesięcy od ostatniego logowania, jednocześnie przesyłając na dowiązany do konta adres mailowy zbiór kart użytkownika w formie pliku txt - szczegółowy opis w sekcji: 3.1.1.4

## 2.2.2. Logowanie użytkownika do konta



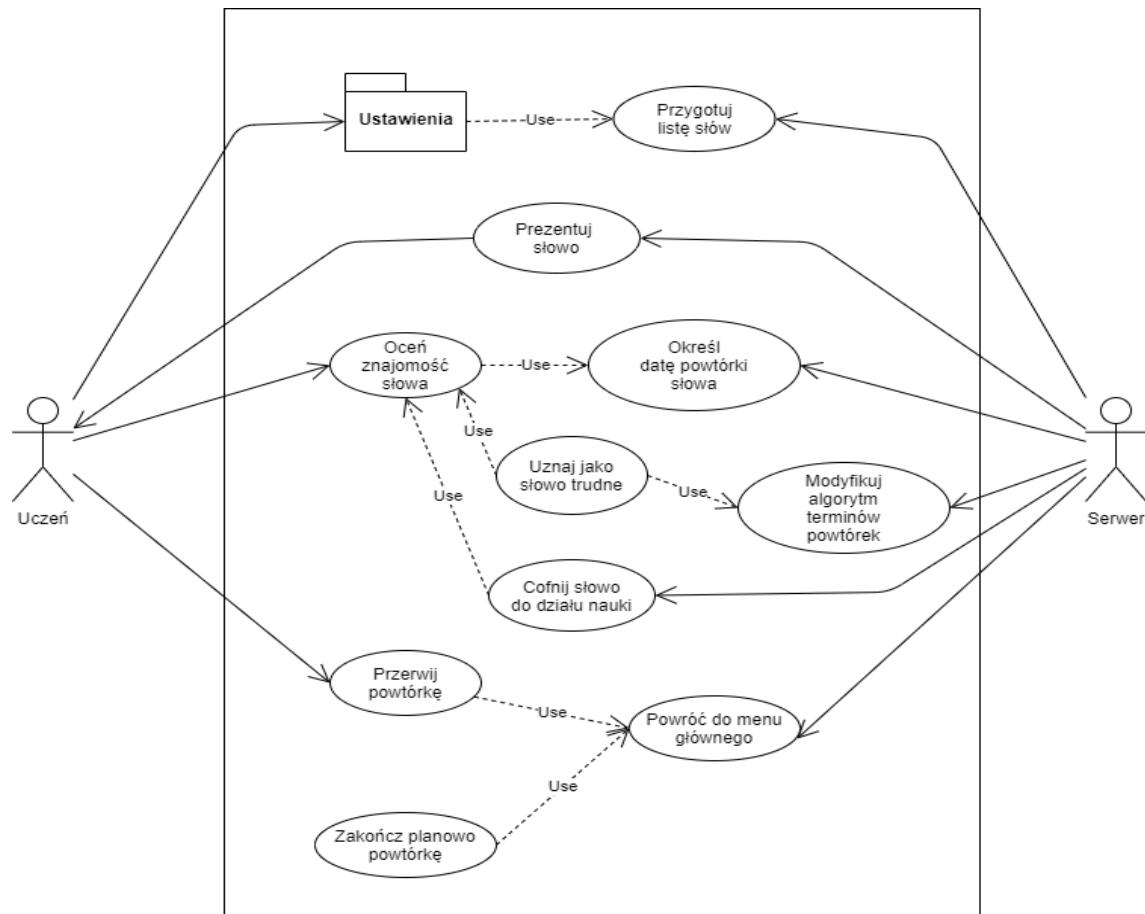
1. System powinien umożliwiać zmianę hasła użytkownika po jego zapomnieniu - dostępne jedynie dla zarejestrowanych użytkowników - szczegółowy opis w sekcji: 3.1.2.1
2. System powinien umożliwiać zalogowanie się użytkownika po podaniu przez niego danych z formularza rejestracyjnego - szczegółowy opis w sekcji: 3.1.2.2
3. System powinien umożliwiać wylogowanie się użytkownika w dowolnym momencie jego pracy z programem - szczegółowy opis w sekcji: 3.1.2.3

### 2.2.3. Nauka kart



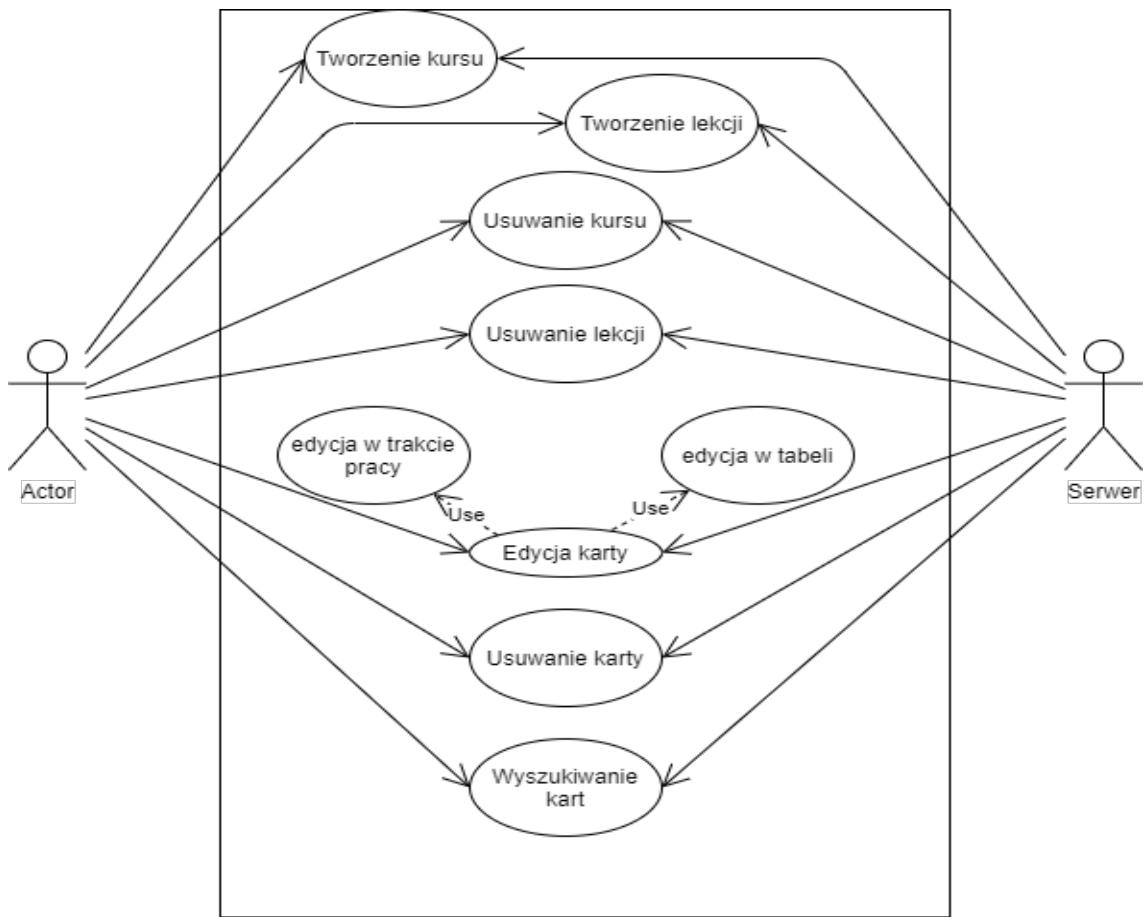
1. System powinien umożliwiać przeprowadzenie fazy nauki kart - szczegółowy opis w sekcji: 3.1.3.1
  2. System powinien umożliwiać przeprowadzenie etapu przeuczenia, po zakończeniu każdej fazy nauki kart - szczegółowy opis w sekcji: 3.1.3.2
  3. System powinien umożliwiać przeprowadzenie podsumowania po zakończeniu każdej fazy nauki, lub etapu przeuczenia - szczegółowy opis w sekcji: 3.1.3.3

## 2.2.4. Przeprowadzanie powtórek



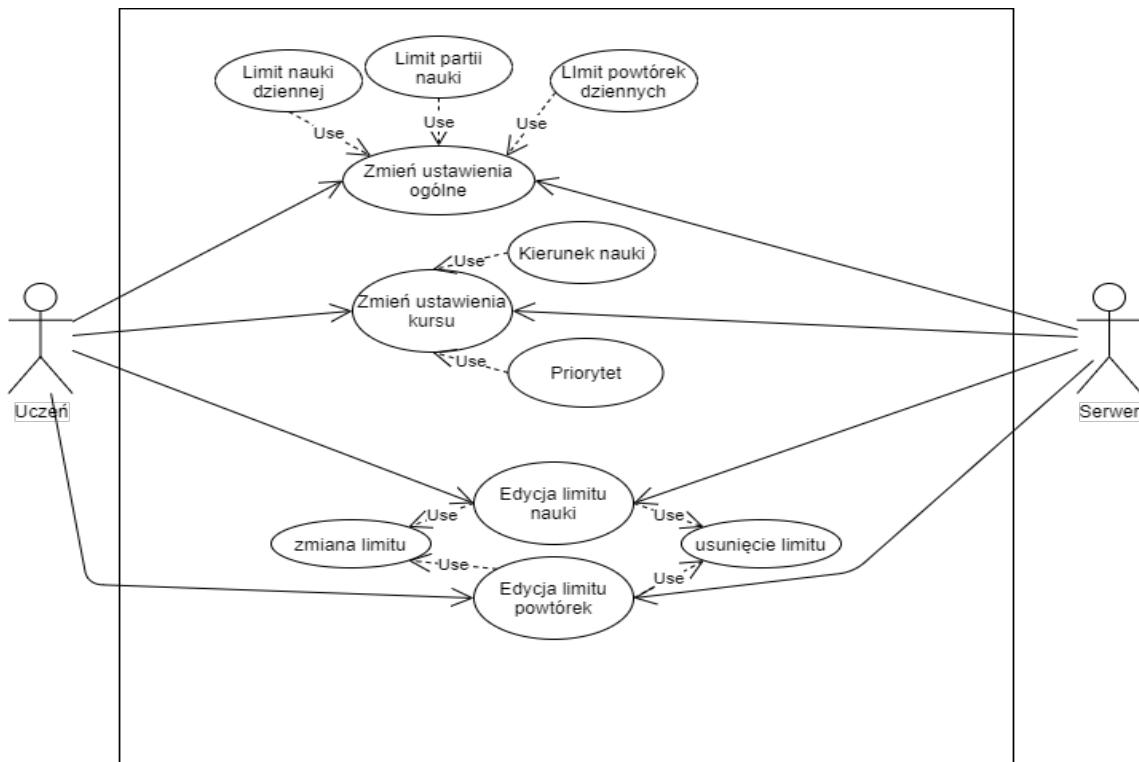
1. System powinien umożliwiać przeprowadzanie powtórek zapamiętanego materiału słownego - szczegółowy opis w sekcji: 3.1.5.1
2. System powinien umożliwiać przeprowadzanie krótkich sesji powtórkowych, wprost z ekranu głównego, po zalogowaniu, bez wchodzenia do żadnego z kursów- szczegółowy opis w sekcji: 3.1.5.2
3. System powinien umożliwiać przeprowadzanie pierwszej powtórki już po 10-ciu minutach od zakończenia nauki - szczegółowy opis w sekcji: 3.1.5.3

## 2.2.5. Edycja danych



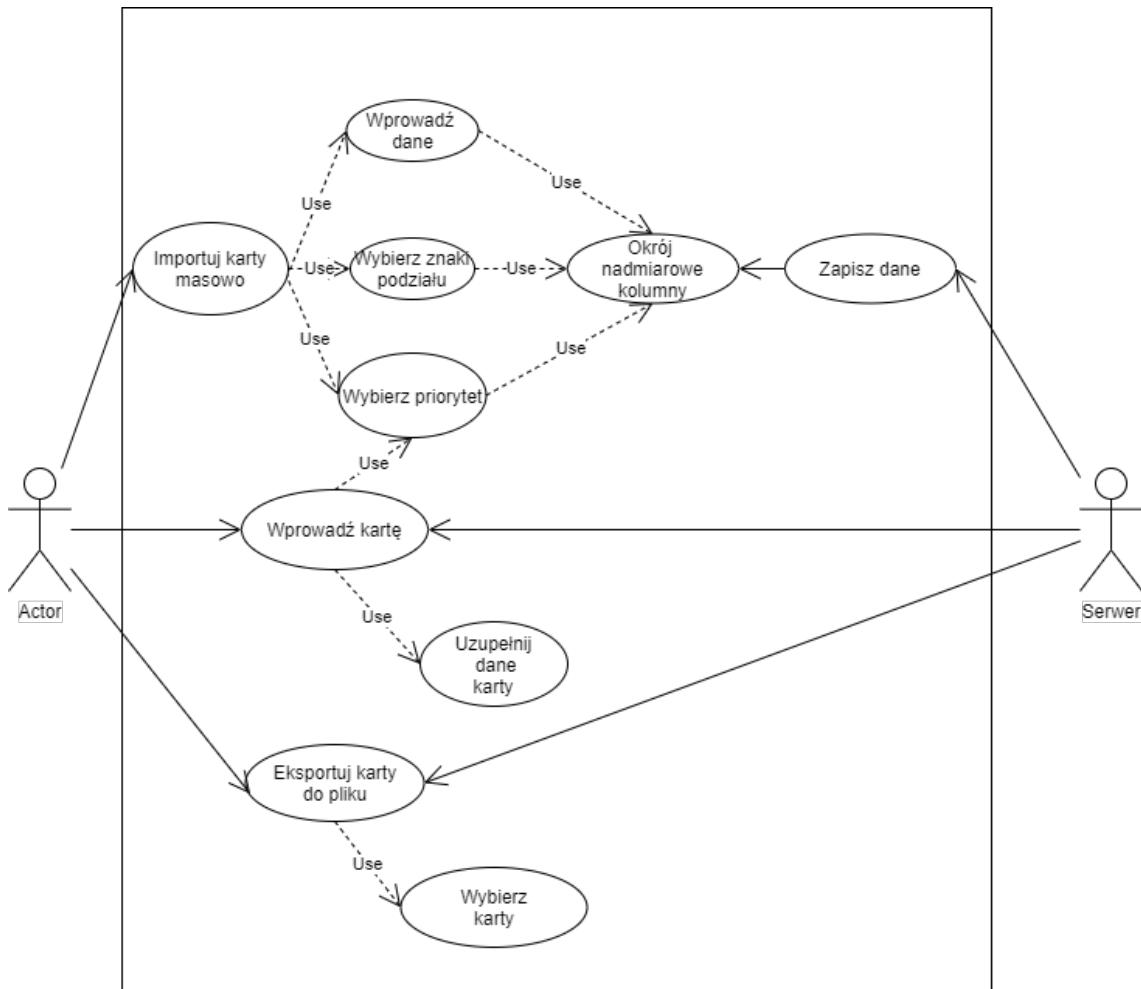
1. System powinien umożliwiać tworzenie nowego kursu w obrębie konta użytkownika - szczegółowy opis w sekcji: 3.1.5.1
2. System powinien umożliwiać tworzenie nowej lekcji w obrębie bieżącego kursu - szczegółowy opis w sekcji: 3.1.5.2
3. System powinien umożliwiać usuwanie kursu przez użytkownika wraz z przynależnymi do niego lekcjami i kartami - szczegółowy opis w sekcji: 3.1.5.3
4. System powinien umożliwiać usuwanie lekcji przez użytkownika wraz z przynależnymi do niej kartami - szczegółowy opis w sekcji: 3.1.5.4
5. System powinien umożliwiać edycję karty, po wybraniu jej z tabeli umieszczonej w zakładce lekcji - szczegółowy opis w sekcji: 3.1.5.5
6. System powinien umożliwiać usuwanie karty z programu, zarówno w czasie pracy z nią, jak i korzystając z tabeli w zakładce lekcji - szczegółowy opis w sekcji: 3.1.5.6
7. System powinien dawać użytkownikowi możliwość aby wyszukiwać słowa/karty w tabeli w zakładce lekcji- szczegółowy opis w sekcji: 3.1.5.7

## 2.2.6. Dostosowywanie ustawień programu



1. System powinien umożliwiać użytkownikowi wyświetlenie i edycję na ekranie listy opcji ustawień ogólnych programu - szczegółowy opis w sekcji: 3.1.6.1
2. System powinien umożliwiać wyświetlenie i edycja opcji bieżącego kursu - szczegółowy opis w sekcji: 3.1.6.2
3. System powinien umożliwiać użytkownikowi zmianę limitu dziennego nauki w programie - szczegółowy opis w sekcji: 3.1.6.3
4. System powinien umożliwiać użytkownikowi zmianę limitu dziennego powtórek w programie - szczegółowy opis w sekcji: 3.1.6.4
5. System powinien umożliwiać użytkownikowi usunięcie dziennego limitu kart do nauki - szczegółowy opis w sekcji: 3.1.6.5
6. System powinien umożliwiać użytkownikowi usunięcie dziennego limitu kart do powtórki - szczegółowy opis w sekcji: 3.1.6.6 zmianę limitu dziennego nauki w programie

## 2.2.7. Eksport i import danych



1. System powinien umożliwiać dodawanie kart pojedynczo, na zasadzie wypełniania pól formularza dla danego rekordu - szczegółowy opis w sekcji: 3.1.7.x
2. System powinien umożliwiać dodawanie wielu kart jednocześnie, jako kopia zawartości pliku csv - szczegółowy opis w sekcji: 3.1.7.x
3. System powinien umożliwiać wyeksportowanie zbioru kart danej lekcji, lub kursu do pliku tekstowego - szczegółowy opis w sekcji: 3.1.7.x

## 2.3. Charakterystyka użytkownika

Użytkownik programu **Repeater** powinien być zaznajomiony z obsługą przeglądarek internetowych i umieć w razie potrzeby je instalować i aktualizować.

Powinien także regularnie, optymalnie codziennie, logować się do systemu w celu prowadzenia zaplanowanej partii powtórek, zby nie dochodziło do utraty postępów nauki.

Użytkownik powinien posiadać umiejętność operowania plikami csv, modyfikacji ich elementów rozdzielających komórki i wiersze, a także posiadać umiejętność korzystania z plików z rozszerzeniem .txt .

## 2.4. Ograniczenia

Do pracy systemu konieczne jest aktywne połączenie z siecią web.  
Minimalna szybkość przesyłu danych powinna wynosić 0,5 MB/s.

# Rozdział 3

## Wymagania szczegółowe

### 3.1. Wymagania funkcjonalne

#### 3.1.1. Tworzenie konta użytkownika

##### 3.1.1.1 Tworzenie konta użytkownika – rejestracja

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie powitalnym programu

1. uczeń wybiera opcję Zarejestruj się
2. system otwiera okno z formularzem rejestracyjnym
3. uczeń uzupełnia wymagane pola
  - Imię
  - adres e-mail, pełniący funkcję loginu
  - proponowane hasło do konta
  - potwierdzenie wpisanej propozycji hasła
4. uczeń zatwierdza zmiany poprzez kliknięcie przycisku „zatwierdź”
5. system analizuje wprowadzone dane
6. system zapisuje dane użytkownika jako nowy rekord tabeli użytkowników w bazie danych

**Rozszerzenia:**

- 5.a podane przez użytkownika dane są nieprawidłowe
  - 5.a.1 system wyświetla informację o nieprawidłowościach
  - 5.a.2 użytkownik wprowadza poprawione dane do formularza rejestracyjnego
  - 5.a.3 użytkownik zatwierdza zmiany poprzez kliknięcie przycisku „zatwierdź”
  - 5.a.4 koniec przypadku użycia

##### 3.1.1.2 Potwierdzenie dostępu do podanego w formularzu rejestracyjnym konta mailowego

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik przesłał do systemu wypełniony formularz rejestracyjny

1. system tworzy rekord w tabeli użytkowników w bazie danych na serwerze

- 
2. system wysyła na podany w formularzu rejestracyjnym adres e-mail, wiadomość z linkiem, zawierającym token aktywacyjny
  3. użytkownik otwiera link aktywacyjny w wiadomości na swoim koncie pocztowym
  4. system analizuje zgodność przesłanego do niego tokena z tym, który został przesłany użytkownikowi
  5. system aktywuje konto użytkownika i wyświetla mu informację o tym fakcie
  6. użytkownik może zalogować się na swoje konto

**Rozszerzenia:**

- 4.a Odebrany od użytkownika token jest nieważny, lub nieprawidłowy
  - 4.a.1 system nie aktywuje konta użytkownika i informuje go o tym fakcie na ekranie
  - 4.a.2 użytkownik nie może zalogować się do nowo utworzonego konta
  - 4.a.3 użytkownik może skorzystać z opcji odtworzenia hasła po jego zapomnieniu, w celu aktywacji konta
  - 4.a.4 koniec przypadku użycia

### 3.1.1.3 Likwidacja konta przez użytkownika

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest zalogowany

1. uczeń wybiera opcję likwidacja konta
2. system wyświetla okno z wyborem opcji eksportu zasobów użytkownika
3. uczeń zaznacza na liście kursy, których materiał chce eksportować
4. uczeń wybiera format pliku docelowego i adresu mailowego do przesyłania pliku
5. uczeń wybiera opcję zatwierdź
6. system wyświetla okno confirm z pytaniem o ostateczną decyzję
7. uczeń wybiera opcję potwierdzam
8. system dokonuje przesyłania pliku
9. system w ciągu 7-mu dni od tego momentu, kasuje zbiory użytkownika i jego dane logowania.

**Rozszerzenia:**

- 4.a uczeń chce zmienić domyślny adres mailowy
  - 4.a.1 uczeń wybiera opcję zmień adres
  - 4.a.2 system otwiera formularz wpisywania adresu mailowego
  - 4.a.3 uczeń wprowadza adres mailowy i ponownie jego potwierdzenie
  - 4.a.4 uczeń wybiera opcję zatwierdź
  - 4.a.5 system zamyka okno wprowadzania adresu i wyświetla okno likwidacji konta
  - 4.a.6 koniec przypadku użycia
    - 5.a rezygnacja z likwidacji konta
      - 5.a.1 uczeń kliką w przycisk „anuluj”
      - 5.a.2 uczeń wybiera opcję anuluj
      - 5.a.3 system zamyka okno i wyświetla okno główne konta użytkownika
      - 5.a.3 koniec przypadku użycia

### **3.1.1.4 Automatyczne zamknięcie konta użytkownika**

**Aktorzy:** system

**Warunki wstępne:** brak aktywności przez okres kolejnych 6-ciu miesięcy

1. System raz w miesiącu podsumowuje aktywność wszystkich użytkowników w okresie ostatnich 6-ciu miesięcy
2. Gdy zostanie stwierdzony brak logowania w tym okresie, system przesyła na adres mailowy użytkownika, informację o planowanym zamknięciu konta w ciągu 30-tu dni, jeśli nie nastąpi logowanie.
3. Gdy w czasie kolejnego sprawdzenia aktywności, okaże się, że dane konto nie było używane przez ostatnie 7 miesięcy, system przystąpi do jego likwidacji
4. System eksportuje zasoby przypisane użytkownikowi do pliku w formacie xls
5. System przesyła plik z eksportowanymi danymi na adres mailowy użytkownika
6. system usuwa zasoby użytkownika z bazy danych i kasuje jego dane logowania

## **3.1.2. Logowanie użytkownika do konta**

### **3.1.2.1 Zmiana hasła po jego zapomnieniu**

**Aktorzy:** użytkownik

**Warunki wstępne:** użytkownik jest zarejestrowany w systemie, a hasło uległo przez niego zapomnieniu

1. użytkownik wybiera opcję zapomniłem hasła na ekranie powitalnym
2. system otwiera okno formularza do podania adresu email
3. użytkownik wprowadza adres email, który podał przy rejestracji konta w systemie
4. użytkownik zatwierdza kliknięcie przycisku „zatwierdź”
5. system weryfikuje użytkownika na podstawie podanego adresu email
6. system wysyła do użytkownika wiadomość mailową z linkiem do formularza zmiany hasła
7. użytkownik przechodzi do linkowanego formularza i wprowadza nowe hasło i jego potwierdzenie
8. użytkownik zatwierdza kliknięcie przycisku „zatwierdź”

#### **Rozszerzenia:**

- 8.a użytkownik podał nieprawidłowe potwierdzenie nowego hasła
- 8.a.1 system wyświetla informację o błędzi i ponownie udostępnia formularz
- 8.a.2 koniec przypadku użycia

### **3.1.2.2 Logowanie użytkownika do konta**

**Aktorzy:** użytkownik

**Warunki wstępne:** Użytkownik jest nie zalogowany ale posiada konto w systemie

1. użytkownik wybiera opcję zaloguj na ekranie powitalnym
2. system otwiera okno formularza, w którym wyświetla wszystkie potrzebne dane logowania
3. użytkownik uzupełnia dane i zatwierdza je przyciskiem

- 
4. system analizuje prawidłowość wprowadzonego hasła i loginu
  5. system akceptuje dane i otwiera okno główne programu właściwe dla konta użytkownika

**Rozszerzenia:**

- 5.a użytkownik nie podał właściwych dnych logowania
- 5.a.1 system informuje użytkownika o niemożności zalogowania i wyświetla ponownie formularz logowania
- 5.a.2 system zamyka okno edycji karty i wyświetla kartę jak na wejściu, bez zmian
- 5.a.3 koniec przypadku użycia

### 3.1.2.3 Wylogowanie się

**Aktorzy:** użytkownik

**Warunki wstępne:** Dostęp z każdego okna programu

1. użytkownik wybiera ikonę konta użytkownika
2. system rozwija listę wyboru akcji
3. użytkownik wybiera opcję Wyloguj
4. system zamyka bieżące okno programu i wyświetla okno logowania
5. koniec przypadku użycia

### 3.1.3. Nauka kart

#### 3.1.3.1 Nauka nowych słów

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** zakończona jest faza prezentacji nowego materiału

1. system wyświetla pytanie L1 z losowo wybranej karty z fazy prezentacji
2. uczeń potwierdza gotowość odpowiedzi
3. system dodaje do wyświetlenia odpowiedź L2
4. uczeń ocenia znajomość słowa
5. system powtarza sekwencję 1-4, pomijając za każdym razem słowa, które już otrzymały ocenę „znam”
6. gdy brak słów do wyświetlania, system wyświetla podsumowanie z informacją o następnym kroku
7. uczeń zatwierdza przekazaną informację
8. system zamyka okno

**Rozszerzenia:**

- 4.a rezygnacja z nauki
- 4.a.1 uczeń w dowolnym momencie tej fazy może zamknąć aplikację, wszystkie spełnione aktywności do tej chwili, powinny być zapisane
- 4.a.2 koniec przypadku użycia

### 3.1.3.2 Zastosowanie przeuczenia

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** nastąpiło zakończenie fazy zapamiętywania

1. system wyświetla pytanie z losowo wybranej karty z fazy zapamiętywania
2. użytkownik potwierdza gotowość do odpowiedzi
3. system dodaje do już wyświetlonej karty, odpowiedź
4. użytkownik zatwierdza kartę
5. system powtarza sekwencję 1-4 dla pozostałych kart z zakończonej fazy zapamiętywania
6. system powtarza sekwencję 1 – 5, tak aby każda karta pojawiła się 50% więcej razy, niż była wyświetlana w fazie zapamiętywania, zanim uczeń potwierdził jej opanowanie
7. system wyświetla okno z informacją podsumowującą
8. użytkownik zatwierdza powrót do strony głównej

**Rozszerzenia:**

- 6.a rezygnacja z nauki
  - 6.a.1 uczeń w dowolnym momencie tej fazy może zamknąć aplikację, wszystkie spełnione aktywności do tej chwili, powinny być zapisane
  - 6.a.2 koniec przypadku użycia

### 3.1.3.3 Podsumowanie fazy nauki kart

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik potwierdził zapamiętanie każdego ze słów z danej partii nauki

1. system wyświetla w nowym oknie informację o zakończonej fazie nauki
2. uczeń klik na przycisk „dalej”
3. system w kolejności dowolnej wyświetla kolejno stronę pytania każdej nauczonej karty
4. uczeń potwierdza za każdym razem gotowość odpowiedzi
5. system dodaje odpowiedź L2
6. uczeń ocenia znajomość słowa
7. system powtarza sekwencję 3-6 dla pozostałych kart które wzięły udział w fazie zapamiętywania
8. system wyświetla w nowym oknie informację o zakończeniu podsumowania
9. uczeń zatwierdza powrót do strony głównej

**Rozszerzenia:**

- 6.a uczeń nie rozpoznaje słowa
  - 6.a.1 system kontynuuje wyświetlanie kolejnych kart
  - 6.a.2 po zakończeniu talii kart, system wyświetla w nowym oknie informację o zakończeniu podsumowania
  - 6.a.3 uczeń może przejść do fazy powtórek
  - 6.a.4 koniec przypadku użycia

### **3.1.4. Przeprowadzanie powtórek**

#### **3.1.4.1 Powtarzanie zapamiętanego materiału**

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie kursu, lub pasek kursu w oknie głównym programu

1. uczeń wybiera opcję „Powtórki” w pasku kursu lub w oknie kursu
2. system otwiera nowe okno z pytaniem L1 wylosowanej z listy karty
3. uczeń potwierdza gotowość do odpowiedzi
4. system dodaje do wyświetlenia odpowiedź L2
5. uczeń dokonuje wyboru stopnia znajomości słowa
6. system powtarza sekwencję 2-5
7. system wyświetla wynik bieżącej tury
8. uczeń dokonuje wyboru opcji kontynuacja
9. system rozpoczyna kolejną turę od punktu 1

**Rozszerzenia:**

- 5.a uczeń usuwa kartę z toku utrwalania
  - 5.a.1 uczeń wybiera opcję usuń kartę
  - 5.a.2 system zamknie bieżącą kartę i wyświetli kolejną na stronie L1
  - 5.a.3 koniec przypadku użycia
- 8.a uczeń kończy powtarzanie materiału
  - 8.a.1 uczeń wybiera opcję wyjście
  - 8.a.2 system zamknie okno powtórek i wyświetli poprzednie okno
  - 8.a.3 koniec przypadku użycia

#### **3.1.4.2 Przeprowadzanie krótkich sesji powtórkowych**

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie okno powitalnym po zalogowaniu

1. uczeń wybiera opcję edytuj w postaci ikony na karcie
2. użytkownik wybiera opcję krótkich powtórek
3. system przeprowadza fazę powtórek, ograniczając ich ilość do 10 kart.
4. system wyświetla confirm z pytaniem o kontynuację powtórek
5. użytkownik wybiera opcję „wyjście”
6. system zamknie confirm i zamknie okno powtórek, wyświetlając w jego miejsce okno powitalne po zalogowaniu
7. Koniec przypadku użycia

**Rozszerzenia:**

- 5.a użytkownik kontynuuje sesję krótkich powtórek
  - 5.a.1 użytkownik wybiera opcję jeszcze raz”
  - 5.a.2 system ponownie przeprowadza sekwencję 3-4
  - 5.a.3 koniec przypadku użycia

### 3.1.4.3 Organizowanie powtórki po 10 minutach od nauki

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie kursu, lub pasek kursu w oknie głównym programu

1. uczeń wybiera opcję „Powtórki” w pasku kursu lub w oknie kursu
2. jeśli wcześniej użytkownik przeprowadzał fazę nauki, po upływie 10 minut od tego czasu system wyznacza słowa z tej partii nauki do obligatoryjnej powtórki
3. system przeprowadza powtórkę w identyczny sposób, jak powtórki regularne, późniejsze
4. po potwierdzeniu przez ucznia znajomości karty, system wyznacza jej termin kolejnej powtórki na q dobę po bieżącej
  - 8.a uczeń kończy powtarzanie materiału
  - 8.a.1 uczeń wybiera opcję wyjście
  - 8.a.2 system zamyka okno powtórek i wyświetla poprzednie okno
  - 8.a.3 koniec przypadku użycia

### 3.1.5. Edycja danych

#### 3.1.5.1 Tworzenie kursu

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie strony głównej

1. uczeń wybiera opcję „stwórz kurs”
2. system wyświetla okno tworzenia zestawu
3. uczeń uzupełnia dane:
  - nazwa zestawu
  - typ danych
  - język pytania i odpowiedzi
4. uczeń wybiera opcję „twórz kurs”
5. system zamyka bieżące okno i wyświetla okno główne
6. uczeń widzi na liście zestawów, utworzony przed chwilą kurs

#### Rozszerzenia:

- 3.a rezygnacja z tworzenia zestawu
  - 3.a.1 uczeń kliką w przycisk „anuluj”
  - 3.a.2 system zamyka bieżące okno i wyświetla okno główne bez zmian
  - 3.a.3 koniec przypadku użycia

#### 3.1.5.2 Tworzenie nowej lekcji

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie kursu

1. użytkownik wybiera opcję „dodaj lekcję”
2. system wyświetla okno dodawania lekcji

- 
3. użytkownik wpisuje nazwę lekcji
  4. system sprawdza, czy nazwa jest unikalna w obrębie danego kursu
  5. użytkownik wpisuje opis lekcji – opcjonalne, niewymagane
  6. użytkownik wybiera opcję „dodaj lekcję”
  7. system wyświetla informację o skutecznym dodaniu lekcji
  8. użytkownik potwierdza zapoznanie się z informacją
  9. system zamyka okno informacji i okno dodawania lekcji i wyświetla okno kursu

**Rozszerzenia:**

- 4.a nazwa nie jest unikalna
  - 4.a.1 system wyświetla alert o konieczności zmiany nazwy
  - 4.a.2 użytkownik wprowadza nową nazwę jak na wejściu, bez zmian
  - 4.a.3 koniec przypadku użycia
    - 6.a użytkownik wybiera opcję „anuluj”
    - 6.a.1 system zamyka okno dodawania lekcji i wyświetla okno kursu
    - 6.a.2 koniec przypadku użycia

### 3.1.5.3 Usuwanie kursu

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie głównym programu

1. użytkownik wybiera opcję „usuń kurs” umieszczoną z prawej strony paska kursu
2. system wyświetla okno confirm z pytaniem „czy jesteś pewien”
3. użytkownik wybiera opcję tak
4. system w formie alertu informuje, że przesyła na adres mailowy użytkownika plik z kartami kursu i nastąpi jego usunięcie
5. użytkownik zatwierdza alert poprzez kliknięcie ok.
6. system odświeża stronę główną konta użytkownika, bez paska usuniętego kursu
7. użytkownik kontynuuje pracę z programem
8. koniec przypadku użycia

**Rozszerzenia:**

- 4.a użytkownik rezygnuje z usunięcia kursu
  - 4.a.1 użytkownik wybiera opcję Nie
  - 4.a.2 system zamyka okno confirm
  - 4.a.3 koniec przypadku użycia

### 3.1.5.4 Usuwanie lekcji

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie kursu

1. użytkownik wybiera opcję usuń lekcję na pasku lekcji
2. system wyświetla confirm z pytaniem „czy jesteś pewien”
3. użytkownik wybiera opcję tak
4. system w formie alertu informuje, że przesyła na adres mailowy użytkownika plik z kartami lekcji i nastąpi jej usunięcie
5. użytkownik zatwierdza alert poprzez kliknięcie ok.

- 
- 6. system odświeża stronę kursu, bez paska usuniętej lekcji
  - 7. użytkownik kontynuuje pracę z programem
  - 8. koniec przypadku użycia

**Rozszerzenia:**

- 4.a użytkownik rezygnuje z usunięcia lekcji
- 4.a.1 użytkownik wybiera opcję Nie
- 4.a.2 system zamknie okno confirm
- 4.a.3 koniec przypadku użycia

### **3.1.5.5.a Edytowanie karty w trakcie pracy z nią (nauka, powtórka, podsumowanie nauki)**

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie pokazującym pytanie, lub pytanie i odpowiedź

- 1. uczeń wybiera opcję edytuj w postaci ikony na karcie
- 2. system otwiera okno edycji karty, w którym wyświetla wszystkie składowe karty
- 3. uczeń dokonuje zmian w poszczególnych składowych karty
  - zmiana pytania karty
  - zmiana odpowiedzi karty
  - zmiana/wprowadzenie zdania przykładowego
  - zmiana/wprowadzenie wymowy
  - dodanie/usunięcie obrazka karty
- 4. uczeń zatwierdza zmiany poprzez kliknięcie przycisku „zatwierdź”
- 5. system zamknie bieżące okno i wyświetli widok karty, jak na wejściu
- 6. uczeń kontynuuje pracę z kartami

**Rozszerzenia:**

- 4.a uczeń nie chce zatwierdzać zmian
- 4.a.1 uczeń kliknie w przycisk „anuluj”
- 4.a.2 system zamknie okno edycji karty i wyświetli kartę jak na wejściu, bez zmian
- 4.a.3 uczeń kontynuuje przerwaną pracę z kartami
- 4.a.4 koniec przypadku użycia

### **3.1.5.5.b Edycja karty z listy lekcji**

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie lekcji

- 1. system wyświetla wszystkie karty lekcji w tabeli
- 2. uczeń wyszukuje słowo/kartę i wybiera ikonę edycji z prawej strony rekordu
- 3. system otwiera okno edycji karty, w którym wyświetla wszystkie składowe karty
- 4. uczeń dokonuje zmian w poszczególnych składowych karty
  - zmiana pytania karty
  - zmiana odpowiedzi karty
  - zmiana/wprowadzenie zdania przykładowego
  - zmiana/wprowadzenie wymowy

- dodanie/usunięcie obrazka karty
- 5. uczeń zatwierdza zmiany poprzez kliknięcie przycisku „zatwierdź”
- 6. system zamyka bieżące okno i wyświetla widok karty, jak na wejściu
- 7. uczeń kontynuuje pracę z kartami

**Rozszerzenia:**

- 4.a uczeń nie chce zatwierdzać zmian
- 4.a.1 uczeń kliką w przycisk „anuluj”
- 4.a.2 system zamyka okno edycji karty i wyświetla kartę jak na wejściu, bez zmian
- 4.a.3 uczeń kontynuuje przerwaną pracę z kartami
- 4.a.4 koniec przypadku użycia

### 3.1.5.7 Usuwanie karty z programu

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie lekcji

1. system wyświetla wszystkie karty lekcji w tabeli
2. uczeń wyszukuje słowo/kartę 3. system podświetla znalezioną kartę i odznacza select po lewej stronie rekordu
3. uczeń wybiera otwiera rozwijaną listę z opcjami działania
4. uczeń wybiera opcję „usuń”
5. system wyświetla okienko confirm
6. uczeń potwierdza decyzję
7. system wyświetla tabelę bez usuniętego elementu/elementów
8. uczeń wybiera opcję „zamknij”
9. system zamyka okno i wyświetla okno edycji zestawu

**Rozszerzenia:**

- 7.a Rezygnacja z usunięcia karty
- 7.a.1 uczeń wybiera opcję „anuluj”
- 7.a.2 system zamyka okno confirm i wyświetla nie zmienioną listę kart lekcji
- 7.a.3 koniec przypadku użycia

### 3.1.5.8 Wyszukiwanie słowa/karty w liście lekcyjnej

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie lekcji

1. system wyświetla wszystkie karty lekcji w tabeli
2. uczeń wyszukuje słowo/kartę poprzez wpisanie frazy w okienko wyszukiwania
3. system podświetla znalezioną kartę i odznacza select po lewej stronie rekordu
4. **Rozszerzenia:**
  - 4.a system znalazł więcej niż jedną pasującą kartę
  - 4.a.1 znalezione rekordy są podświetlone i zgrupowane w widoku tabeli
  - 4.a.2 system zamyka okno edycji karty i wyświetla kartę jak na wejściu, bez zmian
  - 4.a.3 uczeń odznacza selecty rekordów nie podlegających usunięciu
  - 4.a.4 koniec przypadku użycia
  - 4.b uczeń kontynuuje wyszukiwanie kolejnych kart/słów

- 4.b.1 system po każdym wyszukiwaniu wyświetla fragment tabeli z podświetlonym, znalezionym rekordem
- 4.b.2 system nie odznacza selectów poprzednio znalezionych rekordów
- 4.b.3 wszystkie znalezione rekordy są zaznaczone dopóki nie zostanie podjęta decyzja co do ich usunięcia
- 4.b.4 koniec przypadku użycia
- 4.c system nie znalazł szukanego słowa
- 4.c.1 system wyświetla okno alert z informacją o braku efektu wyszukiwania
- 4.c.2 uczeń potwierdza zapoznanie się z alertem
- 4.c.3 system zamyka okno alertu i powraca do poprzedniego widoku
- 4.c.4 koniec przypadku użycia

### **3.1.6. Dostosowywanie ustawień programu**

#### **3.1.6.1 Wyświetlenie i edycja opcji programu**

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w każdym oknie programu

1. użytkownik wybiera ikonę konta użytkownika
2. system rozwija listę akcji
3. użytkownik wybiera opcję Moje opcje
4. system wyświetla okno opcji głównych programu
5. użytkownik dokonuje potrzebnych zmian
  - zmiana limitu dziennej nauki
  - zmiana limitu dziennych powtórek
6. uczeń zatwierdza zmiany poprzez kliknięcie przycisku „zatwierdź”
7. system zamyka okno opcji programu i wyświetla okno pracy
8. uczeń kontynuuje pracę z kartami

**Rozszerzenia:**

- 6.a uczeń nie chce zatwierdzać zmian
- 6.a.1 uczeń kliką w przycisk „anuluj”
- 6.a.2 system zamyka okno opcji programu i wyświetla okno pracy
- 6.a.3 koniec przypadku użycia

#### **3.1.6.2 Wyświetlanie i edycja opcji (ustawień) kursu**

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** strona główna programu

1. uczeń kliką w pasek wybranego kursu
2. system wyświetla okno kursu
3. uczeń zaznacza opcję edycji
4. system wyświetla okno opcji kursu
5. uczeń dokonuje wyboru lub uzupełnienia opcji:
  - pokaż karty czy to ma tu zostać?
  - dodaj karty
  - kalendarz powtórek
6. uczeń wybiera opcję „zamknij”
7. system zamyka okno edycji i wyświetla okno główne

### **3.1.6.3 Zmiana limitu dziennego nauki w programie**

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie opcji programu

1. użytkownik wyszukuje spośród okien, okno oznaczone jako limit dziennej nauki
2. użytkownik wybiera z boku tego okna, ikonę strzałki w góre (zwiększenie) lub strzałki w dół (zmniejszenie) w celu zmiany limitu dziennego słów do nauki
3. użytkownik wybiera opcję zatwierdź
4. system zamyka okno opcji programu i wyświetla okno przerwanej pracy

**Rozszerzenia:**

- 3.a uczeń nie chce zatwierdzać zmian
- 3.a.1 uczeń kliką w przycisk „anuluj”
- 3.a.2 system zamyka okno opcji programu i wyświetla okno pracy
- 3.a.3 koniec przypadku użycia

### **3.1.6.4 Zmiana limitu dziennego powtórek w programie**

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie opcji programu

1. użytkownik wyszukuje spośród okien, okno oznaczone jako limit dziennych powtórek
2. użytkownik wybiera z boku tego okna, ikonę strzałki w góre (zwiększenie) lub strzałki w dół (zmniejszenie) w celu zmiany limitu dziennego słów do nauki
3. użytkownik wybiera opcję zatwierdź
4. system zamyka okno opcji programu i wyświetla okno przerwanej pracy

**Rozszerzenia:**

- 3.a uczeń nie chce zatwierdzać zmian
- 3.a.1 uczeń kliką w przycisk „anuluj”
- 3.a.2 system zamyka okno opcji programu i wyświetla okno pracy
- 3.a.3 koniec przypadku użycia

### **3.1.6.5 Usunięcie dziennego limitu kart do nauki**

**Warunki wstępne:** Użytkownik jest w oknie opcji programu

1. użytkownik wyszukuje spośród okien, okno oznaczone jako limit dziennych powtórek
2. użytkownik zaznacza z boku tego okna, okienko opisane jako „bez limitu”
3. użytkownik wybiera opcję zatwierdź
4. system zamyka okno opcji programu i wyświetla okno przerwanej pracy

**Rozszerzenia:**

- 3.a uczeń nie chce zatwierdzać zmian
- 3.a.1 uczeń kliką w przycisk „anuluj”
- 3.a.2 system zamyka okno opcji programu i wyświetla okno pracy
- 3.a.3 koniec przypadku użycia

### 3.1.6.6 Usunięcie dziennego limitu powtórek

**Warunki wstępne:** Użytkownik jest w oknie opcji programu

1. użytkownik wyszukuje spośród okien, okno oznaczone jako limit dziennych powtórek
2. użytkownik zaznacza z boku tego okna, okienko opisane jako „bez limitu”
3. użytkownik wybiera opcję zatwierdź
4. system zamyka okno opcji programu i wyświetla okno przerwanej pracy

**Rozszerzenia:**

- 3.a uczeń nie chce zatwierdzać zmian
  - 3.a.1 uczeń kliką w przycisk „anuluj”
  - 3.a.2 system zamyka okno opcji programu i wyświetla okno pracy
  - 3.a.3 koniec przypadku użycia

### 3.1.7. Eksport i import danych

#### 3.1.7.1 Dodawanie kart pojedynczo

Aktorzy: użytkownik, uczeń, nauczyciel Warunki wstępne: użytkownik jest w oknie lekcji lub w oknie kursu lub w oknie głównym programu 1. uczeń wybiera opcję dodaj karty pojedynczo 2. system otwiera okno dodawania karty 3. uczeń wpisuje potrzebne dane: treść pytanie, odpowiedź, dodatkowe informacje, wymowę itp 4. uczeń wybiera opcję „zatwierdź” 5. system zapisuje dane w bazie kart dla danej lekcji 6. system zamyka okno dodawania kart i wyświetla stronę lekcji Rozszerzenia: 4.a uczeń wybiera opcję „anuluj” 4.a.1 system zamyka okno dodawania kart, nic nie ulega zapisaniu. 4.a.2 system wyświetla stronę lekcji 4.a.3 koniec przypadku użycia 4.b uczeń wybiera opcję „dodaj kolejną” 4.b.1 system zapisuje bieżącą kartę i otwiera ponownie okno dodawania karty 4.b.2 uczeń postępuje jak w punkcie 3 4.b.3 koniec przypadku użycia

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie pokazującym pytanie, lub pytanie i odpowiedź

1. uczeń wybiera opcję edytuj w postaci ikony na karcie
2. system otwiera okno edycji karty, w którym wyświetla wszystkie składowe karty
3. uczeń dokonuje zmian w poszczególnych składowych karty
  - zmiana pytania karty
  - zmiana odpowiedzi karty
  - zmiana/wprowadzenie zdania przykładowego
  - zmiana/wprowadzenie wymowy
  - dodanie/usunięcie obrazka karty
4. uczeń zatwierdza zmiany poprzez kliknięcie przycisku „zatwierdź”
5. system zamyka bieżące okno i wyświetla widok karty, jak na wejściu
6. uczeń kontynuuje pracę z kartami

**Rozszerzenia:**

- 4.a uczeń nie chce zatwierdzać zmian
  - 4.a.1 uczeń kliką w przycisk „anuluj”
  - 4.a.2 system zamyka okno edycji karty i wyświetla kartę jak na wejściu, bez zmian
  - 4.a.3 uczeń kontynuuje przerwaną pracę z kartami
  - 4.a.4 koniec przypadku użycia

### 3.1.7.2 Dodawanie kart z pliku

**Aktorzy:** użytkownik, uczeń, nauczyciel  
**Warunki wstępne:** użytkownik jest w oknie lekcji 1. uczeń wybiera opcję dodaj karty z pliku/dodaj wiele kart 2. system otwiera okno wklejania tekstu 3. uczeń wkleja skopiowany z pliku tekst 4. uczeń koryguje poprzez znaczenia odpowiedniej opcji, zastosowany w pliku separator i łamacz linii 5. system po każdorazowej zmianie ustawień separatora i łamacza linii, wyświetla ponownie aktualną zawartość okna wklejenia 6. system interpretuje dodatkową „kolumnę” wiersza, jako przykład użycia hasła 7. uczeń wybiera opcję zatwierdź 8. system zapisuje dane w bazie kart dla danej lekcji 9. system zamyka okno dodawania kart i wyświetla stronę lekcji Rozszerzenia: 7.a uczeń wybiera opcję „anuluj” 7.a.1 system zamyka okno dodawania kart, nic nie ulega zapisaniu. 7.a.2 system wyświetla stronę lekcji 7.a.3 koniec przypadku użycia

#### **Edytowanie karty w trakcie pracy z nią**

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie pokazującym pytanie, lub pytanie i odpowiedź

1. uczeń wybiera opcję edytuj w postaci ikony na karcie
2. system otwiera okno edycji karty, w którym wyświetla wszystkie składowe karty
3. uczeń dokonuje zmian w poszczególnych składowych karty
  - zmiana pytania karty
  - zmiana odpowiedzi karty
  - zmiana/wprowadzenie zdania przykładowego
  - zmiana/wprowadzenie wymowy
  - dodanie/usunięcie obrazka karty
4. uczeń zatwierdza zmiany poprzez kliknięcie przycisku „zatwierdź”
5. system zamyka bieżące okno i wyświetla widok karty, jak na wejściu
6. uczeń kontynuuje pracę z kartami

#### **Rozszerzenia:**

- 4.a uczeń nie chce zatwierdzać zmian
  - 4.a.1 uczeń kliką w przycisk „anuluj”
  - 4.a.2 system zamyka okno edycji karty i wyświetla kartę jak na wejściu, bez zmian
  - 4.a.3 uczeń kontynuuje przerwaną pracę z kartami
  - 4.a.4 koniec przypadku użycia

### 3.1.7.3 Eksport kart z kursu do pliku txt

**Aktorzy:** użytkownik/uczeń

**Warunki wstępne:** Użytkownik jest w oknie kursu

1. uczeń wybiera opcję eksportuj
2. system wybiera z zasobów bazy danych karty należące do bieżącego kursu
3. system przygotowuje plik .txt zawierający z każdej karty następujące kolumny:
  - pytanie
  - odpowiedź
  - wymowa

- przykładowe zdanie
  - data kolejnej powtórki
  - ostatni odstęp pomiędzy powtórkami
4. plik ulega zapisaniu na dysku komputera użytkownika
  5. koniec przypadku użycia