# LINGI (?)

Lingi to aplikacja internetowa mająca na celu umożliwienie łatwej i przyjemnej nauki języków obcych poprzez wykonywanie testów dodawanych przez samych użytkowników.

Aplikacja umożliwi dodawanie różnego rodzaju testów w kilku dziedzinach.

Przykładowo:

* dziedzina „Słownictwo”
  + metody nauczania
    - nauka nowych słówek z wybranej kategorii (fiszki)
    - uzupełnianie słówek w tekście
    - quiz wymagający wybrania prawidłowego znaczenia podanego słowa
* dziedzina „Gramatyka”
  + metody nauczania
    - uzupełnienie tekstu prawidłowo odmienionym czasownikiem
* dziedzina „Czytanie”
  + metody nauczania
    - zadania typu prawda - fałsz

Każde zadanie będzie przyporządkowane do odpowiedniej dziedziny, kategorii i poziomu trudności. Przykładowo, korzystając z formularza, dodaję zadanie, które będzie wymagało od innego użytkownika prawidłowej odmiany przymiotnika podanego w nawiasie. Robimy to w dziale „Gramatyka”, wybieramy rodzaj testu „Odmień słowo”, określamy poziom na średniozaawansowany, dodajemy minimum 5 zdań. W tym momencie nasz test może być wykonany przez innych użytkowników, jednak nie mamy pewności czy jest on poprawny, wobec czego dopóki nie zostanie zweryfikowany, nie będzie testem oficjalnym (brak przyznawania punktów za niego).

Proponowane metody weryfikacji testów:

* metoda z kuratorem
  + zakładamy że użytkownikom, którzy udowodnią swój stopień zaawansowania w danym języku, np. certyfikat lektora angielskiego, nadamy rangę kuratora. Od tej pory taki użytkownik może przeglądać dodawane przez innych testy i je „approve’wować”. Po uzyskaniu 3 „approve” od różnych kuratorów test dostaje status zweryfikowany (przez kuratora). Można założyć, że w przyszłości sprawdzanie takich testów oraz dodawanie własnych przez takich kuratorów może być dla nich w jakiś sposób monetyzowane.
* ~~metoda z opiniowaniem~~ najpierw metoda wyżej, w razie czego poniższą można dodać
  + metoda zakłada, że testy będą weryfikowane przez społeczność. Po uzupełnieniu testu wystawiamy mu opinię i zgłaszamy ewentualne błędy. Jeśli średnia danego testu wyniesie powyżej 4 gwiazdek ze 100 opinii, zostanie on uznany jako zweryfikowany (przez społeczność).

Strona umożliwi dodawanie kursów dla różnych języków. Określając język testu należy określić język nauczany i tłumaczony, przykładowo test polsko-angielski to nauka angielskiego dla Polaków. Test rosyjsko-hiszpański to nauka hiszpańskiego z języka rosyjskiego.

By zachęcić do nauki wprowadzony zostanie system statystyk i progresowania. Za ukończone testy użytkownik dostaje punkty dla języka, którego się uczy. Co robić z tymi punktami jest do ustalenia.

Sama aplikacja będzie proponować na stronie głównej po zalogowaniu testy do rozwiązania, na podstawie poziomu użytkownika i kategorii którymi się do tej pory sugerował. Poza tym, że dzięki niektórym testom (fiszki, zadania gramatyczne z wyjaśnieniami) użytkownik może się uczyć języka od podstaw bez innych materiałów, ma przede wszystkim możliwość sprawdzania i utrwalania wiedzy w dziedzinach w których nie czuje się najlepiej. Na przykład po prostu nie ogarniasz różnicy pomiędzy czasami past simple i present perfect to robisz w naszej aplikacji tylko takie testy.

Jest też spory potencjał do rozbudowy, przykładowo układanie całych kursów, czyli łączenia kilkunastu różnych testów pod jakieś zagadnienie, np. jakiś nauczyciel angielskiego chciałby żeby jego grupa je zrobiła jako przygotowanie na lekcję więc wysyła dzieciakom linka do kursu który sam skomponował.

# Szczegóły techniczne

# (wszystko poniżej to propozycje do uzupełnienia)

## Technologie

**Frontend**

* **Angular** do napisania aplikacji typu SPA, pobierania i wyświetlania danych od serwera
* **Bootstrap** do zrobienia strony graficznej aplikacji

**Backend**

Jeden z poniższych do napisania serwera aplikacji (pobieranie danych z bazy danych, przesyłanie ich do klienta, logowanie, obsługa danych metodą REST – komunikacja backendu i frontendu odbywa się przez zasoby w formacie JSON, operacje CRUD - create, read, update, delete)

* **Spring (Java)**
* **.Net Core (C#)**
* **Node.js (JavaScript)**
* **Django (Python)**
* **PHP (xD)**

**Baza danych**

Do ustalenia, nie znam się i nie mam pojęcia, która będzie najlepsza do wdrożenia na serwer itd.

* **Postgres**
* **Mysql**
* **MS SQL**
* **SQLite** (jeśli nie mamy ambicji xD)
* **Inne**

Ja preferuję użyć .Net Core z IDE Visual Studio 2019. Jest tam już gotowy szablon aplikacji z Angularem i mniej więcej to ogarniam. W jednym projekcie mamy pliki czystego Angulara (i innych bibliotek jeśli potrzeba) i klasy do obsługi serwera w C#.

## Baza danych i testy

Testy będą od siebie niezależne, jeśli najdzie nas ochota, dodamy więcej i tyle. Jednak występuje tabela bazowa dla wszystkich testów.

|  |  |  |  |
| --- | --- | --- | --- |
| Tabela **Test**   * AutorID * FirstLanguageID * SecondLanguageID * ~~CategoryID~~ * DifficultyID * FieldID * MethodID * CreationDate * Verified | Słownik **Method**   * Name (np. Fiszki) | Słownik **Language**   * Name | Słownik **Field**   * Name |

Najprostszy z możliwych do zaimplementowania na początek

**Fiszki**

Użytkownik dodaje słowo i jego tłumaczenie oraz zdanie przykładowe dla lepszego zrozumienia. Powinno to być minimum 5 słów i maksymalnie 40 (?). Widzę to tak, że mamy prosty formularz u góry i przycisk „Dodaj”, po kliknięciu którego pod spodem pojawia nam się dodany przed chwilą zestaw. Dzięki Angularowi wszystko jest responsywne i od razu się pokazuje. Do tego jakieś kontrolki „Edytuj” i „Zapisz” żeby w razie czego poprawić błędy.

Działanie testu:

* Wyświetlenie na ekranie jednego słowo i dwie opcje – *Znam*/*Nie znam*
* Jeśli użytkownik wybierze opcję *Znam* to fiszka już się nie pokaże, jeśli *Nie znam*, fiszka pojawi się w następnej turze
* Pokazanie tłumaczenia i zdania przykładowego
* Po przejściu przez wszystkie słówka rozpoczyna się następna tura jeśli chociaż raz wybrano *Nie znam* przy jakiejś Fiszce

W bazie danych odzwierciedlenie testu powinno wyglądać tak:

|  |
| --- |
| Tabela **FlashcardWord**   * TestID * Slowo1 * Slowo2 * Zdanie przykładowe |

## Witryna

Widzę to tak, że mamy stronę główną z podstawowymi informacjami i możliwością zalogowania. Poza tym będzie uniwersalny dla całej aplikacji Layout na pasek nawigacyjny z rzeczami typu Log in, About, Contact, Your Tests, etc.

Po zalogowaniu użytkownikowi powinna wyświetlić się lista polecanych do rozwiązania testów, możliwość dodania własnych testów i możliwość wyszukania testów. Wybranie wyszukania testów wyświetla np. kafelki z nazwami dziedzin, po kliknięciu generuje nowe kafelki tym razem z nazwami dostępnych metod nauczania, następnie wyświetla nam listę proponowanych testów z możliwością filtrowania po poziomie trudności i kategorii.

Analogicznie możliwość dodania testu, po wybraniu metody nauczania wyświetla nam formularz dodania testu dla tej metody.

Każdy test będzie miał uniwersalne ID i będzie zwracany z API żądaniem GET. Za każdy test odpowie osobny kontroler przekazujący zasoby spod adresów takich jak */home/flashcard/2137.* Odpowiadać za to powinien jeden komponent bazowy, który zapamięta ścieżkę. Tak żeby to maksymalnie uprościć, żeby dostęp do każdego testu wyglądał tak samo. Różnice będą dopiero w komponentach wyświetlających testy.

Żeby rozpocząć pracę trzeba ustalić technologie, zaprojektować dokładną bazę danych do przechowywania podstawowych elementów. Serwer i klient powinny być maksymalnie niezależne od siebie, czyli raz a dobrze ustalić format przekazywanych danych i w zasadzie można pracować osobno w backendzie i frontendzie, bez zbędnych poprawek.