Implementacja narzędzia rozszerzającego usługę Google Forms

(Implementation of an extension to Google Forms service)

Agnieszka Pawicka

Praca inżynierska

Promotor: dr hab. Jan Otop, prof. UWr

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

Wrocław 2021

Streszczenie

Praca zawiera implementację oraz opis rozszerzenia do usługi Google Forms. Rozszerzenie to pozwala na automatyczne generowanie formularzy z odpowiedniego pliku w formacie JSON, konwertowanie wstawek matematycznych napisanych w LaTeX do odpowiednich symboli oraz zarządzanie niektórymi własnościami utworzynych uprzednio formularzy. W pracy znajduje się również omówienie wykorzystanych technologii oraz możliwego rozwoju narzędzia.

The paper contains implementation and description of an extension to the Google Forms service. This extension provides automatic generation of forms from JSON file, conversion of mathematical inserts (written in LaTeX) and some forms' properties management for existing forms. This paper also describes used technologies and suggests how presented extension could be further developed.

Spis treści

Rozdział 1.

Wprowadzenie

1.1. Motywacja

Przeprowadzanie testów i ankiet jest wpisane w życie akademickie oraz szkolne. Jedną ze szczególnych metod przeprowadzania testu lub ankiety jest weryfikowanie wiedzy czy jej zdalne zdobywanie. Aby to jednak było wartościowe, należy spełnić warunki takie jak:

- czytelność formy,
- możliwość weryfikowania, kto wysłał odpowiedź,
- kontrolowanie czasu wysyłanych odpowiedzi,
- zapamiętywanie przesłanych odpowiedzi.

Oprócz powyższych kryteriów możliwym byłoby wymienienie jeszcze wielu kwestii, których implementacja znacznie ułatwiłaby pracę ze zdalnymi testami – zwłaszcza dla strony przeprowadzającej test, jak np.:

- automatyczne generowanie testu z popularnego formatu,
- możliwość wstawiania symboli matematycznych,
- automatyczne ocenianie (dla testów) – być może w nietypowej skali,
- generowanie losowych testów z pytań z pewnej puli,
- udostępnianie wyników w wygodnym formacie.

Zupełnie nowe narzędzie — możliwie blisko ideału — byłoby więc ambitnym przedsięwzięciem dla zespołu programistów. Tu z pomocą przychodzi możliwość rozszerzania istniejących rozwiązań.

1.2. Gotowe rozwiązania

Do najpopularniejszych gotowych rozwiązań należą formularze Googla, oraz Microsoftu – udostępniają one już wiele z wymienionych w poprzednim rozdziale możliwości – są czytelne, zapewniają weryfikowanie tożsamości osób przesyłających odpowiedź (sprawdzają zalogowanego kontem mailowym użytkownika, w Google Forms dzieje się to przez standard autoryzujący OAuth), są popularne, bezpieczne, niezawodne, wygodne w użyciu.

Wymienione narzędzia mają jednak wady. Głównymi problemem w przeprowadzaniu większych testów/ankiet jest konieczność ręcznego „przeklikiwania się" przez interfejs do tworzenia formularzy, a w dziedzinach ścisłych również brak wbudowanej interpretacji symboli matematycznych – chociaż własności, które byłyby pomocne jest znacznie więcej. Udostępnione możliwości oceniania automatycznego nie pozwalają na ustawienie nietypowej (jak np. wykładniczej) skali oceniania. Google Forms w czystej formie nie pozwalają też na automatyczne ustawienie ram czasowych akceptacji odpowiedzi – czasu początku i końca „testu". Format odpowiedzi jest trudny do automatycznego przetwarzania.

Jest jednak ważna zaleta gotowych rozwiązań – można do nich dobudowywać dalsze usprawnienia. Niniejsza praca polega na usprawnieniu gotowego rozwiązania – jakim są Google Forms – o parę nowych możliwości.

1.3. Cel pracy

Poprzez wykorzystanie Google Apps Script i rozwiązań serwerowych bibliotek JavaScriptu powstało narzędzie usprawniające formularze Google. W niniejszej pracy zostało zaimplementowane automatyczne generowanie formularzy z plików w formacie JSON, renderowanie wstawek napisanych w LaTeX'u oraz możliwość prostego włączania oraz wyłączania opcji, mówiącej czy formularz przyjmuje w danym momencie zgłoszenia. Niektóre opcje formularzy można przekazać już na podstawie kodowania JSONowego – w konstruowanym pliku.

Rozdział 2.

Środowisko

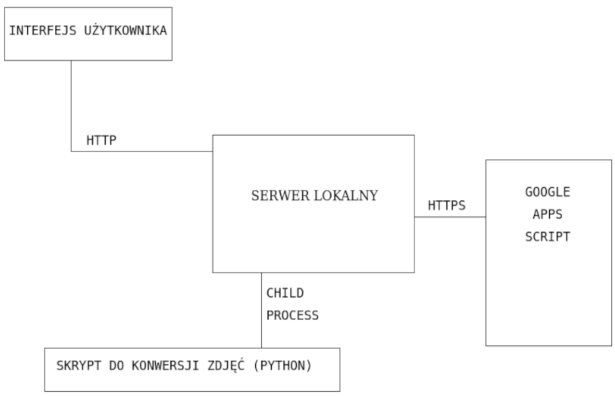
Napisanie programu rozszerzającego istniejące narzędzie wymaga zapoznania się z udostępnionym API i doboru użytych technologii. Niniejszy rozdział opisuje te właśnie aspekty.

Program składa się z trzech głównych elementów:

- interfejsu, napisanego przy pomocy HTML i CSS (2.1),
- serwera zaimplementowanego w środowisku Node.js (rozdział 2.2),
- kodu rozszerzającego narzędzie Google Forms (opisanego w 2.3),

Fragmenty kodu korzystają również z bibliotek Pythonowych – skrypt do konwersji symboli matematycznych i obrazów.

Komunikacja pomiędzy elementami pracy wygląda następująco:



Rysunek 2.1: Komunikacja pomiędzy elementami pracy

2.1. Bootstrap

Popularna biblioteka CSS, ułatwiająca budowanie interfejsów graficznych stron internetowych pisanych w HTML. Strony budowane za pomocą tego narzędzia wyświetlają się czytelnie zarówno na urządzeniach mobilnych jak i na większych ekranach. Biblioteka zawiera również wiele gotowych widżetów, co znacznie usprawnia pracę. Oficjalna strona bootstrap [10].

2.2. Node.js

Node.js jest środowiskiem uruchomieniowym JavaScriptu - służącym do tworzenia aplikacji serwerowych. Praca wykorzystuje kilka bibliotek, przede wszystkim korzysta jednak z możliwości serwerowych Node.js.

https Interfejs Node.js silnie związany z „sercem" środowiska - udostępnia narzędzia do komunikacji poprzez protokół HTTPS zarówno ze strony serwerowej jak i klienckiej. W pracy w czystej formie wykorzystywany do wysyłania zapytań pomiędzy serwerem lokalnym a serwerem Google'a.
Dokumentacja: https [2]

express „Szybki (...), minimalistyczny framework webowy dla Node.js" - narzędzie pozwala w przystępny sposób postawić serwer (korzysta z biblioteki http). Udostępnia cztery klasy, z których trzy są wykorzystywane aktywnie w pracy:

- application – odpowiada aplikacji serwerowej,
- request – zarządza odwołaniami do serwera (głównie parametrami),
- response – odpowiada za odpowiedzi serwera.

Trzon kodu narzędzia opiera się właśnie na serwerze express-owym.

Dokumentacja znajduje się tutaj: express.js [3].

cors Node.js-owy moduł pozwalający na odpowiednie ustawienia CORS (Cross-Origin Request Sharing) w rozwiązaniach typu express. CORS jest metodą rozwiązania problemów z domyślnymi ustawieniami związanymi z bezpieczeństwem — dodaje do zapytania HTTP nagłówek, opisujący które źródła mają uprawnienia do pobierania danych z serwera. Standardowo dane z jednej strony mogą być pobierane z poziomu drugiej strony, gdy obie są z tego samego źródła (ten sam schemat URL, host oraz port). CORS pozwala na uniknięcie tej konieczności.

Github modułu: cors [4]

child_process Moduł Node.js-owy pozwalający na uruchamianie podprocesów. W przypadku omawianego kodu, umożliwia uruchamianie skryptów napisanych w Pythonie z poziomu kodu Node.js-owego.

Dokumentacja znajduje się tutaj: child_process [5]

jsonschema Nowy (zaledwie dziesięciomiesięczny, wciąż w wersji Beta) moduł pozwalający na walidację formatu json zgodnie z podanym schematem.

Oficjalna strona: jsonschema [6]

2.3. Google Apps Script

Nabudowywanie na gotowym narzędziu wymaga dostępu do niego. Google udostępnia API operujące na całym szeregu klas i metod oferowanych usług. Do programowania aplikacji operujących na usługach Google'a służy platforma Google Apps Script. Kod przypisany jest do danego konta Google, możliwe jest ustawienie parametrów, takich jak: kto ma dostęp do nowotworzonej aplikacji internetowej (właściciel, zalogowany użytkownik danej organizacji, dowolny zalogowany użytkownik, każdy) pod czyim kontem jest ona uruchamiana (właściciela/współwłaściciela, czy też zalogowanego użytkownika).

Framework Udostępnione API w pewnym stopniu wymusza na użytkownikach, aby kod wykonywany na infrastrukturze Google'a był napisany w JavaScripcie, we frameworku „Google Apps Script". Program jest wykonywany po stronie serwera. Pozwala on na stosunkowo łatwe manipulowanie działaniem produktów Google takich jak formularze, arkusze, dysk i inne.

Framework powstał w 2009 roku, w JavaScript 1.6, jest jednak regularnie ulepszany.

Komunikacja Apps Script udostępnia komunikację przez protokoły HTTP oraz HTTPS — jeśli projekt zawiera funkcje doGet(e) / doPost(e), odpowiednie żądania wykonują kod z ciała tych metod. Zwracane wartości prowadzą do przekierowań zapytań: Google nie zwraca danych wartości bezpośrednio, zamiast tego tworzy stronę zawierającą zwracane wartości. Automatycznie generowany jest nowy adres URL. Wysłanie żądania GET pod ten właśnie nowy adres pozwala na dotarcie do potrzebnych danych.

API Google Apps Script udostępnia szereg klas i metod związanych z poszczególnymi narzędziami. Szczegółowa dokumentacja narzędzia znajduje się tutaj: Forms Service [1]. Główna klasa – FormApp – jest odpowiedzialna za zarządzanie formularzami – m.in. tworzenie nowych. Każdy typ pytania i element formularza ma odpowiednią klasę (jak np. CheckboxItem czy SectionHeaderItem). Poprzez klasę From

można zmieniać główne ustawienia formularzy – jak na przykład dodawanie właścicieli, tworzenie pytań, automatyczne ocenianie, manipulowanie tytułem, ustawienie, czy formularz jest „aktywny" (czy przyjmuje odpowiedzi).

2.4. Python

Rozwiązania pythonowe zostały wykorzystane w celu konwersji wstawek matematycznych (LaTeX) do obrazów. Poniżej krótki opis wykorzystanych bibliotek.

tex2pix Biblioteka pozwalająca na konwertowanie formatu .tex do różnych formatów graficznych. Metoda konwertująca format .tex na format .png zaczyna od konwersji .tex do .pdf, stąd w pracy używana jest konwersja do pdf z tej biblioteki, a dalsze manipulowanie formatem używa innych – subiektywnie prostszych w użytkowaniu – bibliotek.

Oficjalna strona: tex2pix [7].

pdf2image Biblioteka umożliwiająca konwersję formatu pdf do formatów graficznych.

Oficjalna strona: pdf2image [8].

opencv Biblioteka pozwalająca na manipulację obrazami. Udostępnia znacznie więcej możliwości niż te użyte w pracy. W szczególności pozwala na przycinanie obrazów względem ich kolorystyki – co pozwala na automatyczne przycięcie strony pdf do rozmiarów napisanego na niej tekstu. Ważna informacja: najnowsza wersja biblioteki nie ma wsparcia dla **Python 2.7** – zaleca się używanie **Python 3**. Więcej informacji na temat biblioteki: opencv [9]

base64 Biblioteka pozwala na konwersję obrazu do formatu Base64 – używanego w pracy do przesyłu obrazów pomiędzy serwerami node'owym a google'owym.

Rozdział 3.

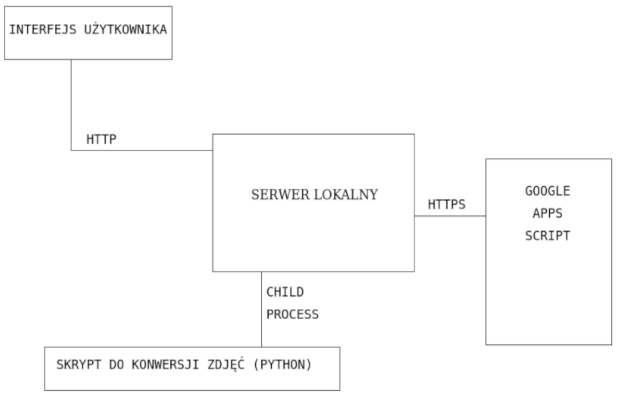
Techniczny opis programu

Możliwe jest dalsze rozwijanie przedstawianego narzędzia. Ten oto rozdział zawiera techniczny opis kodu programu oraz schemat komunikacji pomiędzy jego modułami.

Narzędzie składa się z trzech głównych części:

- interfejsu użytkownika napisanego przy użyciu frameworku CSSowego „Bootstrap”,
- serwera napisanego w Node.js. Serwer jest uruchamiany lokalnie na urządzeniu użytkownika, komunikuje się z aplikacją internetową oraz interfejsem użytkownika,
- aplikacji internetowej, napisanej we frameworku Google Apps Script po stronie Google'a, pod prywatnym kontem e-mailowym.

Poza tym lokalny serwer wykorzystuje kilka bibliotek JavaScriptowych oraz Pythonowych do dodatkowych obliczeń. Schemat połączeń w projekcie:



Rysunek 3.1: Schemat połączeń

3.1. Interfejs użytkownika

Na interfejs użytkownika składają się trzy pliki: *strona.html*, *memoryFile.js* oraz *htmlCode.js* – pierwszy z nich koduje część wizualną, drugi dane na temat formularzy, trzeci jest odpowiedzialny za zachowanie poszczególnych elementów, w tym za komunikację z serwerem.

Elementami interfejsu są:

- pole do wgrywania plików, przyjmujące formaty .json oraz .txt. Zawartość powinna spełniać wymagania opisane w instrukcji w podrozdziale 4.2,
- przycisk wgrywający plik – „generate form" wywołuje metodę getFile,
- listę formularzy, w której poprzez kliknięcie można wybrać formularz,
- kilka przycisków powiązanych z metodą getInfo().

Plik *memoryFile.js* jest modyfikowany przez serwer lokalny i służy do zapamiętywania formularzy utworzonych przez aplikację. Jako kod zawiera on wyłącznie deklarację stałej „memory", będącej obiektem JSON następującej budowy:

```
{"forms":
    [{"id": "string",
      "date": "string",
      "name": "string"}
    ]
}
```

Wartość „id” jest generowana przez usługę Google'a przy tworzeniu formularza, „date” to data utworzenia formularza (rok-miesiąc-dzień), „name” odpowiada tytułowi.

Wartość „selectedForm” z pliku *htmlCode.js* odpowiada identyfikatorowi formularza obecnie podświetlonego na niebiesko na liście formularzy. Metody zaimplementowane w *htmlCode.js*:

- changeLocation(newLocation) – pozwala na zmianę wyświetlanego adresu,
- makeHttpRequest(Url, callback) – wysyła żądanie GET do lokalnego serwera (adres URL zależny jest od rodzaju przeprowadzanej akcji), w przypadku pozytywnej odpowiedzi (HTTP status 200) wywołuje metodę callback,
- getInfo(action) – metoda przypisana większości przycisków interfejsu, jest odpowiedzialna za odpowiednie wywołanie makeHttpRequest,
- getFile() – funkcja przypisana do przycisku „Generate form”, wczytuje wgrany plik, sprawdza, czy jest on poprawnym JSONem i wywołuje makeHttpRequest z odpowiednimi argumentami.
- assign(id) – metoda przypisana do elementów listy formularzy, ustawia wartość „selectedForm” przy każdej zmianie wybranego z listy formularza,
- generateList() – tworzy listę formularzy na podstawie *memoryFile.js*, jest uruchamiana przy ładowaniu strony.

3.2. Serwer Node.js-owy

Serwer to prosta implementacja aplikacji we frameworku „express.js”. Po uruchomieniu nasłuchuje na porcie 3000 (co można zmienić w pliku *server.js*, linia 3). Możliwe są trzy odwołania do serwera:

- uploadJsonFile
- createForm
- getInfo

Aplikacja korzysta z bibliotek „express”, „cors”, „fs”, „https”, „jsonschema”, „child_process” oraz skryptów *jsonValidator.js* oraz *tex2png.js* i *text2png.py*. Zaimplementowana metoda *makeRequest(options, data)* służy do komunikacji z serwerem po stronie Google'a.

Zasady działania poszczególnych fragmentów są opisane w odpowiednich podrozdziałach.

3.2.1. Komunikacja międzyserwerowa

Za komunikację pomiędzy serwerami odpowiedzialna jest metoda makeRequest(options, data) korzystająca z Node.js-owej biblioteki „https". Zasada działania biblioteki jest następująca – https request przyjmuje dwa argumenty: pierwszy (options) deklaruje szereg parametrów (adres URL zapytania, typ metody, zawartość, nagłówki zapytania itp.), drugi jest metodą wywoływaną na wartości zwracanej przez zapytanie.

Funkcja makeRequest(options, data) jako argumenty przyjmuje klasę options (zdefiniowaną w bibliotece https) oraz dane do przesłania. Zwraca konstrukcję *Promise* — obiekt biblioteki JavaScriptu, będący „obietnicą" wykonania asynchronicznego zapytania. *Promise* posiada stan (*pending, fulfilled* lub *rejected*) oraz metody *then* i *catch*, pozwalające na wykonanie kolejnych operacji dopiero, gdy stan obiektu będzie odpowiednio *fulfilled* lub *rejected*. Konstrukcja ta pozwala na szeregowanie kolejnych zapytań. Wewnątrz metody przechwytującej odpowiedź obsługiwane jest przekierowanie zapytania (wysyłane z każdą odpowiedzią Google'a zawierającą zawartość) — czyli wysyłane kolejne zapytanie do nowego adresu.

3.2.2. Ścieżki serwera

Lokalny serwer komunikuje się z interfejsem graficznym poprzez obsługiwanie zapytań pod różnymi adresami w formacie *localhost:3000/wywoływanaAkcja* (np. *localhost:3000/getInfo*). Poniżej przedstawiono obsługiwane ścieżki serwera wraz z opisem ich działania.

uploadJsonFile Ścieżka pozwala na wgranie na serwer zakodowanego formularza (za i wstępne przetworzenie go. Ścieżka wywoływana jest z metody getFile() (patrz interfejs użytkownika, rozdział 3.1). Na początku kontrolowana jest zgodność kodowania ze schematem (plik *jsonValidator*). Jeśli podany plik jest zgodny, przeprowadzana jest konwersja LaTeX'a do obrazów (plik *tex2png.js*). Po tym etapie informacja o zgodności jest zwracana do interfejsu.

createForm Jeśli interfejs otrzyma informację o zgodności wgranego JSONa ze schematem, wywołuje zapytanie do serwera na ścieżce createForm. W tym miejscu dla pytań z wartością „tex"=true serwer podmienia wartości „text" pytań na kodowanie obrazów w base64. Następnie wysyłane jest zapytanie POST do serwera po stronie Google'a z zakodowanym formularzem. Jeśli nie będzie błędów – zwrócona odpowiedź będzie zawierała identyfikator nowego formularza. W tym miejscu następuje również modyfikacja pliku *memoryFile.js* – dodawany jest nowy element tabeli – data jest pobierana z klasy „Date", nazwa z pola „title" z kodowania formularza. Na koniec zwracana jest informacja do interfejsu.

getInfo Za pomocą tej ścieżki przekazywana jest komunikacja dotycząca istniejących formularzy pomiędzy interfejsem a serwerem Google'owym. Jest to też miejsce modyfikacji pliku *memoryFile.js* – dla żądania „delete" usuwany jest wpis dotyczący danego formularza.

3.2.3. JSON validation

Plik zawiera deklarację schematu JSON (patrz: instrukcja, rozdział 4.2). Za pomocą biblioteki jsonschema wykonywana jest walidacja zawartości przesłanego pliku.

3.2.4. Plik tex2png.js

Zadaniem tego modułu jest uruchomienie nowych procesów (child_process.spawn) – skrypt *tex2png.py* w Pythonie zajmuje się konwersją tekstu na grafiki odpowiednich rozmiarów. Głowna funkcja korzysta z konstrukcji *Promise*.

3.3. Google Apps Script

Aplikacja internetowa stanowi rozwiązanie, pozwalające na komunikację poprzez HTTP oraz HTTPS. Odwołać do niej może się każdy użytkownik znający adres URL aplikacji, kod wykonywany jest pod kontem Google właściciela aplikacji (w chwili obecnej aplikacja jest utworzona pod prywatnym kontem Google).

Po stronie Google'a znajdują się dwa pliki: *communication.gs* oraz *createFrom.gs*, odpowiadające kolejno za komunikację po HTTPS oraz za zarządzanie tworzeniem formularzy.

3.3.1. HTTPS

Plik *communication.gs* zawiera implementację metod doPost(e) oraz doGet(e). Framework zapewnia, że są one wykonywane, gdy do aplikacji przyjdą zapytania HTTPS (odpowiednio POST i GET). Do parametrów funkcji są przekazywane treści zapytań.

Metoda doGet(e) jest wykorzystywana do zapytań dotyczących tego, czy dany formularz przyjmuje przesyłane odpowiedzi oraz do uzyskiwania informacji na temat adresów URL danego formularza. Poprawne zapytanie powinno zawierać dwie wartości:

- formId – identyfikator formularza przypisywany automatycznie w momencie tworzenia,

- action – pole tekstowe mówiące o tym, co autor zapytania chce zrobić. Obsługiwane wartości:
  - *isActive* zwraca wiadomość tekstową informującą o tym, czy formularz przyjmuje odpowiedzi,
  - *activate* aktywuje formularz (po wykonaniu tej części formularz przyjmuje odpowiedzi),
  - *deactivate* dezaktywuje formularz,
  - *publisherUrl* zwraca adres url formularza dla respondentów,
  - *editorUrl* zwraca adres url do edycji formularza,
  - *delete* również dezaktywuje formularz (usuwanie formularzy z poziomu skryptu wymaga implementacji rozszerzenia do usługi Dysku Google).

Z braku udostępnionej metody usuwającej żądany formularz – zapytanie o usunięcie formularza dezaktywuje go.

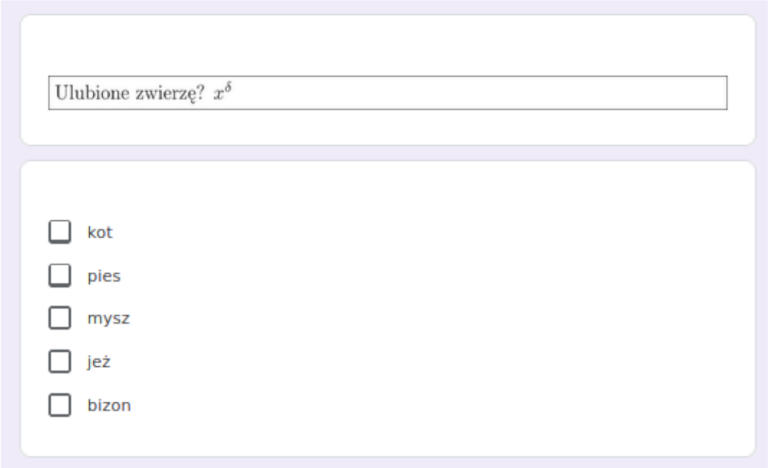
Metoda doPost(e) odpowiada za tworzenie nowych formularzy. Przesyłana zawartość zawiera zakodowany w formacie JSON formularz, wstępnie przetworzony przez lokalny serwer (zamiana pytań z wstawkami LaTeX'owymi na zdjęcia w formacie base64). Ta metoda uruchamia funkcję *createFromJSON* z pliku *createForm.gs*.

3.3.2. Zarządzanie formularzami

Plik *createForm.gs* zawiera kilka metod służących do tworzenia konkretnych rodzajów pytań:

- checkBox – pytanie zamknięte wielokrotnego wyboru,
- grid – pytanie zamknięte mające formę siatki polami do wyboru. W niniejszej pracy kolumny siatki są ustawione na „prawdę" i „fałsz", wiersze są zakodowanymi w JSONie odpowiedziami. API nie udostępnia automatycznego oceniania tego typu pytań,
- list – pytanie zamknięte jednokrotnego wyboru – odpowiedź jest wybierana z listy rozwijanej,
- text – pytanie otwarte, ocenianie automatyczne również niemożliwe.

Pytania ze wstawkami LaTeX'owymi są w formie obrazów, nie posiadają możliwych odpowiedzi – tuż pod nimi tworzone jest pytanie bez treści zawierające możliwe odpowiedzi. W praktyce wygląda to w następujący sposób:



Rysunek 3.2: Przykład

Takie rozwiązanie wynika z ograniczeń API formularzy, które nie udostępnia metod wprowadzenia zdjęć do popularnych typów pytań – jest to znany problem, zgłaszany tutaj: https://issuetracker.google.com/issues/36765518?pli=1. Metoda *image* tworzy pole z grafiką w formularzu. Funkcja *setFromFeatures* modyfikuje ustawienia formularza – chodzi tu o zapamiętywanie adresów mailowych użytkowników przesyłających odpowiedzi, limit odpowiedzi na użytkownika, ustawienie współwłaściciela formularza oraz wartości, czy formularz powinien być oceniany automatycznie.

Główną funkcją w pliku jest *createFromJSON*, która generuje formularz poprzez wywoływanie odpowiednich metod z wymienionych wyżej. Właścicielem formularza jest właściciel aplikacji internetowej (użytkownik Google, pod którego kontem tworzone są formularze). Takie rozwiązanie pozwala na uniknięcie potrzeby autentyfikacji przy każdym dostępie do aplikacji internetowej.

3.4. Konwersja wstawek z LaTeX'a

Niniejsza część korzysta z podstawowej biblioteki LaTeX'a, szablon tworzonych obrazów stanowi plik *texTemplate.tex*. Wszystkie tworzone pliki znajdują się w folderze *pictures*.

Decyzja o napisaniu tej części w Pythonie powstała stosunkowo późno. Powodem jest biblioteka openCV – udostępniająca zaawansowane metody obliczeniowe w przystępny sposób. Skrypt działa następująco:

- Wczytywany jest szablon w LaTeX'u i w odpowiednie miejsce wstawiany jest tekst z pytania (oznaczonego tex=true) – warto wspomnieć, że treść pytania jest otoczona ramką.
- Za pomocą biblioteki tex2pix plik .tex jest konwertowany do formatu pdf.

- Biblioteka pdf2image konwertuje plil formatu pdf do png.
- Biblioteka openCV pozwala na „wycięcie" z grafiki wyłącznie obszaru otoczonego ramką (jako jedyny zawiera kolory).
- „Ramka" jest zapisywana jako png.
- Przy użyciu biblioteki base64 końcowy obraz jest konwertowany do formatu base64 i zapisywane do pliku.
- Pośrednie stadia konwersji są usuwane z folderu.

3.4.1. Alternatywne rozwiązanie

Wydaje się naturalne, aby do problemu parsowania symboli matematycznych użyć gotowego silnika (opartego np. na Markdownie lub LaTeX'u). Takim silnikiem jest na przykład MathJax. Udostępnia bogate API, integrację Webową. Doskonale współpracuje z JavaScriptem. Główną komplikacją jest to, że zwracany wynik jest w HTMLu — jako, że jest to format, nieobsługiwany przez Google Forms, wymagałoby to dodatkowej konwersji. Nie wykluczone jednak, że istnieje znacząco prostsze rozwiązanie - korzystające właśnie z takiego silnika.

3.5. Skrypty instalujący i uruchamiający

Instalacja oparta jest na npm (dla Node.js) i pip (Python) – skrypt *install.sh* instaluje wymagane biblioteki przy pomocy tychże narzędzi.

Uruchomienie programu wymaga uruchomienia dwóch części – serwera lokalnego oraz interfejsu – ten drugi wymaga przeglądarki internetowej. Skrypt *run.sh* wypisuje ścieżkę pliku stanowiącego interfejs z prośbą o otwarcie go w przeglądarce i uruchamia serwer lokalny.

Rozdział 4.

Instrukcja użytkownika

4.1. Początki pracy z narzędziem

Wtyczka pozwala na wygenerowanie formularza Google Forms z pliku kodującego w formacie JSON, automatyczne definiowanie jego treści na podstawie wstawek w LaTeX'u oraz zarządzanie informacjami o tym, czy dany formularz przyjmuje przesyłane odpowiedzi. Na początek należy jednak pobrać i zainstalować zależności wykorzystywane przez narzędzie.

4.1.1. Instalacja

Na początku należy sklonować repozytorium projektu — github.com/agnpawicka/pracaInzynierska.

Następnie wejść w folder **source** i uruchomić plik o nazwie „install.sh". Pozwoli to na zainstalowanie potrzebnych bibliotek.

4.1.2. Praca z wtyczką

Aby włączyć narzędzie należy uruchomić plik o nazwie „run.sh". Uruchomi on lokalny serwer Node.js-owy oraz stronę internetową z interfejsem użytkownika.

4.2. Schemat pliku kodującego (JSON)

Poniżej znajduje się rozpisany schemat kodowania:

```
{"title": "string",
 "email": "string",
 "check": "boolean",
 "questions": {[{"type": "string",
                 "text": "string",
                 "tex" : "boolean",
                 "answers": [{"answer" : "string",
                              "correct" : "boolean"}],
                 "points": "number"}]}
}
```

Jest to opis tych wartości, które mogą być użyte – nie wszystkie są jednak wymagane. Poniżej znajduje się szczegółowy opis pól i wartości:

- title – wartość tekstowa, odpowiada nagłówkowi formularza - **pole wymagane**,
- email – adres e-mail edytora formularza (konieczne konto google) – **pole opcjonalne**,
- check – wartość boolowska mówiąca o tym, czy formularz ma udostępniać opcję automatycznego oceniania. Domyślna wartość to **false**. W przypadku ustawienia wartości na **true** należy się upewnić, że przy każdym pytaniu są poprawnie ustalone wartości „correct” oraz „points” – **pole opcjonalne**,
- questions – tablica, której każde pole zawiera informacje dotyczące jednego pytania – **pole wymagane**. Poniżej znajdują się wartości kodujące pojedyncze pytanie:
  - type – pole tekstowe dotyczące typu kodowanego pytania. Dopuszczalne wartości:
    - „checkBox” – pytanie zamknięte wielokrotnego wyboru,
    - „grid” – pytanie typu „prawda/fałsz”,
    - „list” – zamknięte jednokrotnego wyboru,
    - „text” – otwarte.

    pole wymagane,
  - text – zawiera treść pytania w formie tekstowej. Może zawierać wstawki z LaTeX'a. Należy jednak pamiętać, że JavaScript traktuje symbol „\” jako specjalny – wszystkie wystąpienia „\” należy więc zastąpić „\\” – **pole wymagane**,
  - tex – wartość boolowska, jeśli **true** treść pytania (text) będzie konwertowana do obrazu z zachowaniem konwersji symboli matematycznych i innych wstawek z LaTeX'a z biblioteki standardowej, **pole wymagane**,

– answers – tablica, każde pole zawiera jedną z możliwych odpowiedzi w następującej formie:

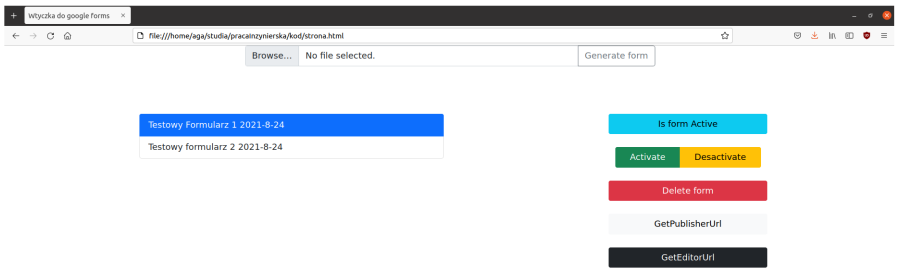
  * answer – pole tekstowe, treść danej odpowiedzi
  * correct – wartość boolowska wskazująca czy dana odpowiedź jest prawidłowa. Domyślna wartość: false.

  -pole opcjonalne

– points – wartość numeryczna, odpowiada liczbie punktów przyznawanej za poprawą odpowiedź na pytanie (dotyczy wyłącznie oceniania automatycznego) – **pole opcjonalne**

4.3. Obsługa narzędzia

Po uruchomieniu użytkownik widzi stronę w przeglądarce, jak na zdjęciu poniżej:



Rysunek 4.1: Interfejs aplikacji

Widoczne u góry pole do wgrywania plików przyjmuje formaty .txt oraz .json. W pliku powinien znajdować się zakodowany formularz (podrozdział 4.2. Schemat pliku kodującego (JSON)).
Poniżej po lewej stronie znajduje się lista utworzonych już formularzy, po prawej znajdują się przyciski służące do operowania na już istniejących formularzach.
Przycisk „Generate form” wgrywa podany plik i wykonuje na nim kolejne operacje. Poprawne wykonanie powinno przechodzić przez kolejne etapy:

- Sprawdzany jest format pliku. Jeśli zawartość jest obiektem typu JSON, dane przekazywane są do lokalnego serwera, w przeciwnym przypadku strona wyświetli alert informujący o niepoprawnym formacie.

- Serwer lokalny sprawdza zgodność pliku ze schematem (JSON schema). Po tym etapie poniżej pola do wgrywania plików powinna pojawić się jedna z poniższych informacji:
  - Validation succeded
  - Wrong JSON format
- Jeśli plik JSON jest zgodny ze schematem, następuje konwersja pytań zakodowanych jako „tex" na format graficzny.
- Po zakończonej konwersji, z lokalnego serwera wysyłany jest POST request do serwera po stronie Google, gdzie odbywa się konwersja pliku na formularz. Zdalny serwer odsyła informację po zakończonej pracy do serwera lokalnego.
- Lokalny serwer dodaje nowy formularz do listy.
- Wyświetla się komunikat **New form has been created. Please reload the page** z prośbą o odświeżenie strony.

Zachowania poszczególnych przycisków – za wyjątkiem „Generate Form" – dotyczą zawsze wybranego formularza z listy (podświetlonego w danym momencie na niebiesko). Aby wybrać formularz należy kliknąć na niego w liście formularzy.

Widoczne w interfejsie przyciski mają następujące funkcje:

Is Form Active zwraca wartość **Form is active** jeśli formularz przyjmuje odpowiedzi oraz **Form is inactive** w przeciwnym przypadku.

Activate wysyła do serwera po stronie Google'a zapytanie, jeśli aktywacja formularza przebiegła pomyślnie, zwracana jest wiadomość **Form activated**.

Deactivate wysyła do serwera po stronie Google'a zapytanie, jeśli dezaktywacja formularza przebiegła pomyślnie, zwracana jest wiadomość **Form deactivated**.

Delete Form wysyła do serwera po stronie Google'a zapytanie o dezaktywację formularza, następnie usuwa z pliku z danymi o formularzach wpis dotyczący wybranego formularza oraz w komunikacie **Form deactivated, please reload page.** prosi o odświeżenie strony.

Get Publisher Url wysyła do serwera po stronie Google'a zapytanie o adres url dla respondentów wybranego formularza. W komunikacie pojawia się odpowiedni link.

Get Editor Url wysyła do serwera po stronie Google'a zapytanie o adres url dla edytorów wybranego formularza. W komunikacie pojawia się odpowiedni link.

Rozdział 5.

Podsumowanie

5.1. Wnioski

Celem pracy była implementacja narzędzia pozwalającego na uproszczenie przeprowadzania testów i ankiet online w oparciu o istniejące już rozwiązanie – Google Forms. Efektem jest program, ułatwiający pracę z formularzami Google'a poprzez umożliwienie automatycznego generowania pytań oraz konwersję symboli matematycznych. Możliwe jest jednak dalsze rozwijanie narzędzia.

5.2. Rozwój

Możliwych rozszerzeń jest wiele – niniejszy rozdział opisuje kilka z nich.

Zarządzanie przesłanymi odpowiedziami – Google Apps Script udostępnia klasę ItemResponse – odpowiedzialną za zarządzanie odpowiedziami respondentów. Możliwe jest przechwytywanie odpowiedzi, oceny (wystawionej automatycznie, jeśli tak ustawiono), pytania, do którego jest odpowiedz (jako klasy, a wiec razem z możliwymi odpowiedziami, punktami, treścią) a także ustawienie komentarza. Dalej: możliwe jest przetwarzanie pozyskanych danych i ocenianie testów przez zewnętrzny program na niestandardowych zasadach (jak na przykład wykładnicza skala punktowa w zależności od liczby poprawnych odpowiedzi).

Możliwość ustawiania losowej kolejności pytań – choć w teorii ten problem jest rozwiązany przez udostępnione API, w praktyce pojawia się problem. Jak wspomniano wyżej – API formularzy nie udostępnia metod wprowadzenia obrazów do popularnych typów pytań. Powoduje to konieczność utrzymywania dwóch formalnie osobnych pytań (w praktyce pytania-obrazy i odpowiedzi) w jednym miejscu w formularzu.

Ustawianie czasu rozpoczęcia oraz zakończenia testu – aby mogło się to odbywać automatycznie, potrzebne jest urządzenie, które będzie regularnie i w krótkich odstępach czasowych sprawdzało, czy należy już wysłać odpowiednie zapytanie do zaimplementowanej aplikacji, czy jeszcze nie. Może się to wykonywać po stronie serwera lokalnego lub w samej aplikacji internetowej – po stronie Google'a. Zaimplementowany serwer nie uwzględnia jednak tej możliwości ze względu na konieczność ciągłego trwania w stanie uruchomionym.

Bibliografia

[1] Google. Dokumentacja forms service w google apps script. https://developers.google.com/apps-script/reference/forms.

[2] Node.js. Dokumentacja biblioteki https. https://nodejs.org/api/http.html.

[3] Node.js. Dokumentacja framework'u express.js. https://expressjs.com/.

[4] Troy Goode. Dokumentacja biblioteki cors. https://github.com/expressjs/cors.

[5] Node.js. Dokumentacja biblioteki child_process. https://nodejs.org/api/child_process.html.

[6] Tom de Grunt. Dokumentacja biblioteki jsonschema. https://www.npmjs.com/package/jsonschema.

[7] Andy Buckley. Dokumentacja biblioteki tex2pix (python). https://pypi.org/project/tex2pix/.

[8] Edouard Belval. Dokumentacja biblioteki pdf2image (python). https://pypi.org/project/pdf2image/.

[9] Maintener: Olli-Pekka Heinisuo. Dokumentacja biblioteki opencv (python). https://pypi.org/project/opencv-python/.

[10] Dokumentacja biblioteki bootstrap. https://getbootstrap.com/.