

Studio projektowe - WEB OCR

Autorzy:

- Kamil Sikora
- Konrad Kalita
- Karol Widuch

[Założenia Projektu](#)

[Wybrana metodyka](#)

[Licencja](#)

[Funkcjonalności](#)

[Architektura](#)

[Frontend](#)

[Backend](#)

[Struktura aplikacji](#)

[Opis poszczególnych komponentów](#)

[Wymagania нефunkcjonalne](#)

Założenia Projektu

Aplikacja ma umożliwiać wykorzystanie Google OCR API do analizy tekstów (plików PDF). Wyświetlane będą właściwości tekstu oraz będzie możliwość jego edycji już po sparsowaniu poprzez API.

Użytkownik będzie miał podgląd przetwarzanego pliku wraz z możliwością nawigacji między stronami.

Wybrana metodyka

Klasyczna - wybraliśmy ją ze względu na to, że pozwala na dokładne zaplanowanie poszczególnych prac oraz bezwzględne trzymanie się terminów. Projekt będzie realizowany krok po kroku, bez jakichkolwiek zmian, ponieważ jest to dość wąska tematyka to nie przewidujemy żadnych większych zmian.

Licencja

W projekcie użyta będzie licencja GPL-3.0 License.

Funkcjonalności

- Pole tekstowe wyświetlające zawartość detekcji wraz z możliwością edycji
- Podgląd PDF - widoczna aktualna strona przesłanego pliku
- Miniaturowy podgląd wszystkich stron wraz z możliwością nawigacji pomiędzy nimi
- Podgląd właściwości wykrytego tekstu
- Przycisk wywołujący modal do obsługi przesyłanie plików
- Przesyłanie plików z własnego urządzenia poprzez interakcje użytkownika z wyświetlanym modelem
- Przesyłanie plików z własnego urządzenia na zasobnik cloud storage
- Wysyłanie zapytań i otrzymywanie odpowiedzi z Google Vision Api

Architektura

Frontend

Wykorzystywane technologie:

- React
- TypeScript
- ChakraUI
- Axios (biblioteka do wysyłania zapytań i komunikacji z backendem)

Backend


Do stworzenia backendu wykorzystamy Framework Next JS API Routes wraz z Node.JS.

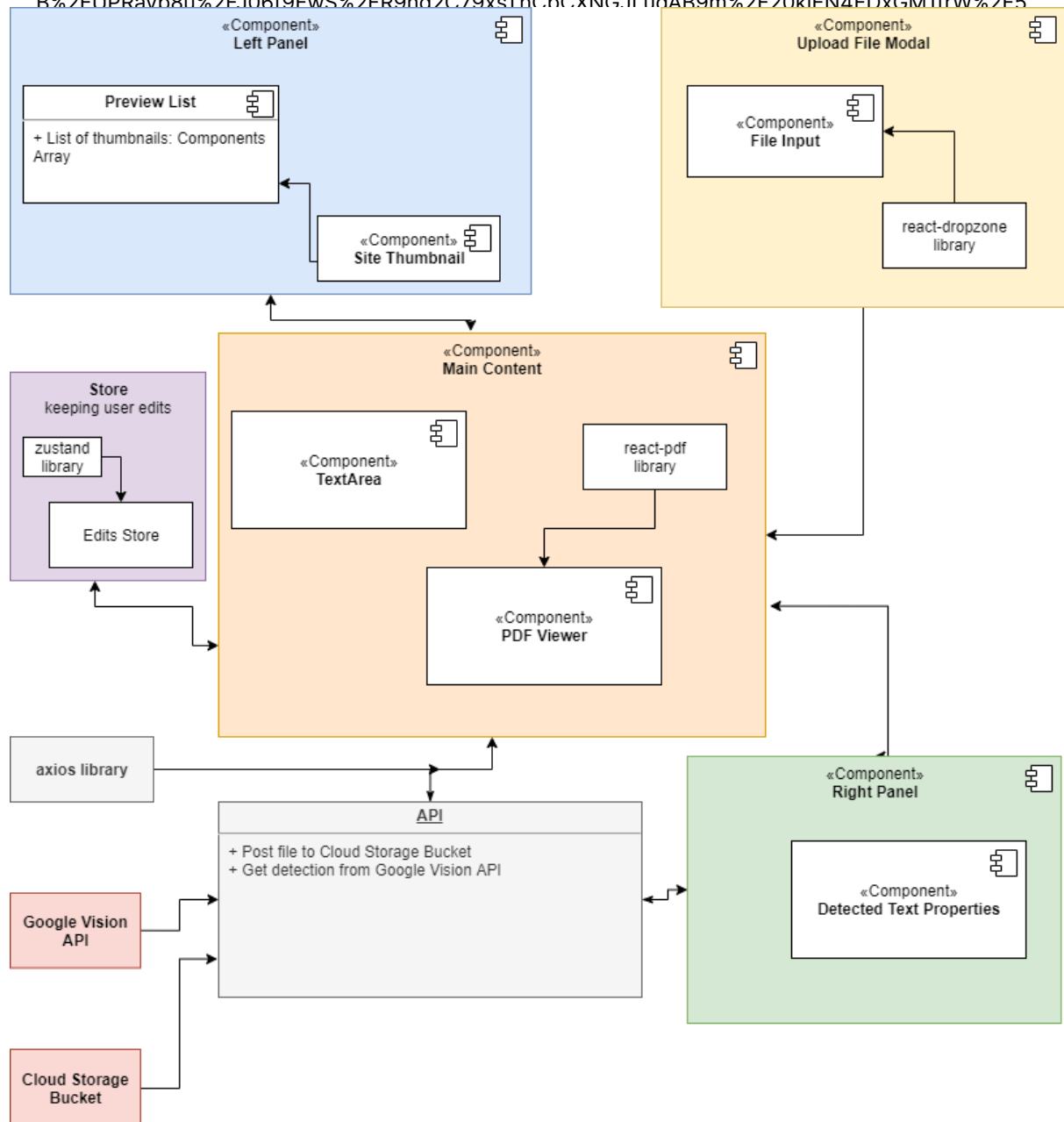
Backend będzie odpowiadał za komunikację z Google Vision Api oraz Cloud Storage Bucket.

Struktura aplikacji

Flowchart Maker & Online Diagram Software

diagrams.net (formerly draw.io) is free online diagram software. You can use it as a flowchart maker, network diagram software, to create UML online, as an ER diagram tool, to design database schema, to build BPMN online, as a circuit diagram maker, and more. draw.io can import .vsdx, Gliffy™ and

 [https://viewer.diagrams.net/?highlight=0000ff&edit=_blank&layers=1&nav=1&title=components.drawio#R7Vxtc5s4EP41%2BegMICTwx9hJ2s40M5ILrtfelxvZCJsLRhTLiZ1ffxIlzlulHZu3zKWdac0Kgbza](https://viewer.diagrams.net/?highlight=0000ff&edit=_blank&layers=1&nav=1&title=components.drawio#R7Vxtc5s4EP41%2BegMICTwx9hJ2s40M5ILrtfelxvZCJsLRhTLiZ1ffxIlzlulHZu3zKWdac0Kgbza59nVruQLMF1tv0Q4XN5Rh%2FgXhuZsL8D1hWHoGrD5f0KySyTQAllgEXmOvGkvePBeSdpTSjeeQ9aFGxmIPvPConBOg4DMWUGGo4i%2BFG9zqv98a4gXpCJ4mGO%2FKv3Lc9gykdqGtZd%2FJd5imb5ZR%2BOKZYXTm%2BU3WS%2BxQ19ylnBzAaYRpSz5tNpOiS%2BUI%2Bol6Xdb05oNLCIBO6bDk7u5%2BR%2EUPRavp8u%2EJ06t9EwS%2ER9nq2C79xsThCnCXNG.II ugAB9m%2E20kiEN4FDxGM1frW%2E5)



Opis poszczególnych komponentów

Main Content:

- W głównym oknie znajdują się dwa komponenty główne - *TextArea*, *PDF Viewer*. Znajdują się one na środku widoku.
- Komponent po lewej stronie umożliwia podgląd (odpowiednio duże okno, tak aby było czytelne) obecnie wybranej strony PDFa. Na dole tego komponentu znajdują się przyciski do nawigacji pomiędzy stronami. Wszystkie zmiany stron są zsynchronizowane z widokiem znajdującym się w panelu bocznym po lewej stronie (opis tego komponentu poniżej).
- Drugi komponent typu *TextArea* znajdujący się po prawej stronie daje wynikowy tekst po przepuszczeniu go poprzez Google OCR API. Zwrócony tekst jest w pełni edytowalny - użytkownik może w niego ingerować, a jego akcje zapisywane są w stanie aplikacji.

Left Panel:

- Panel boczny znajdujący się po lewej stronie umożliwia podgląd załadowanego lokalnie pliku PDF. Zawiera on listę komponentów *SiteThumbnail*, które reprezentują poszczególne strony PDFa. Możemy tam swobodnie poruszać się po wszystkich stronach i przełączać między nimi. Widoczne tam strony PDFa nie mają żadnych zmian i są czystym plikiem wczytanym przez użytkownika. Do poprawnego wyświetlenia PDFa użyta została biblioteka *react-pdf*.

Right Panel:

- Panel boczny znajdujący się po prawej stronie prezentuje właściwości wykrytego przy użyciu Google OCR API tekstu.

Upload File Modal:

- Komponent odpowiedzialny za umożliwienie wrzucenie pliku lokalnie od użytkownika i przesłanie go na backend (NextJS). Komponent ten używa biblioteki *react-dropzone*, która poza zwykłym wybraniem pliku z systemu plików umożliwia funkcjonalność *drag'n'drop*. Wybierany plik jest walidowany i w przypadku, gdy jego rozszerzenie różni się od *.pdf* to stosowna informacja zwrotna wyświetlana jest użytkownikowi.

- Jeśli plik zostanie zaakceptowany (ma rozszerzenie PDF) to jest on przesyłany na backend gdzie plik jest przesyłany dalej do Google Cloud Storage. Przechowywanie tam pliku jest wymagane, ponieważ tylko stamtąd Google OCR API jest w stanie wykonać detekcję. Po zapisaniu tego pliku wysyłane jest zapytanie, aby została wykonana detekcja tekstu. Jako rezultat odsyłany jest JSON z wykrytym tekstem i jego parametrami.

Store:

- mechanizm *state-management* z wykorzystaniem pakietu *zustand*. Store przechowuje wykrytą detekcję w tablicy tekstów dla każdej strony pliku. Po modyfikacji detekcji przez użytkownika elementy tablicy są modyfikowane. Dzięki takiemu przechowywaniu tekstów można się do nich łatwo odwołać w całej aplikacji oraz zachować zmodyfikowany tekst użytkownika przez cały czas jego użytkowania aplikacji

API:

- API umożliwia komunikację z Cloud Storage oraz Google Vision Api. Frontend do komunikacji z API wykorzystuje pośrednio bibliotekę *axios*.
- Upload plików do zdefiniowanego zasobnika Cloud Storage wykorzystując metodę POST. Przesyłany plik jest wcześniej wprowadzony przez użytkownika
- Wysyłanie requestów do Google Vision Api o detekcję tekstu na podstawie nazwy do wcześniej zapisanego w zasobniku Cloud Storage pliku PDF. Otrzymywanie odpowiedzi w formacie JSON zawierającej detekcję wraz z dodatkowymi właściwościami.

Do stylowania każdego z powyższych komponentów została użyta ChakraUI. Do dbania o poprawność typowania i czysty kod zostały użyte odpowiednio TypeScript + Eslint.

Wymagania niefunkcjonalne

- utworzony odpowiedni zasobnik w Cloud Storage
- posiadanie odpowiedniego pliku autoryzacyjnego aby uzyskać dostęp do zdefiniowanego zasobnika Cloud Storage

- poprawne działanie Google Vision Api
- poprawne działanie Api Cloud Storage
- termin ukończenia aplikacji na 16 czerwca 2021