POLITECHNIKA BIAŁOSTOCKA

WYDZIAŁ INFORMATYKI

PRACA DYPLOMOWA INŻYNIERSKA

TEMAT: APLIKACJA INTERNETOWA DO OBSŁUGI SYSTEM GDT W TECHNOLOGII PYTHON/DJANGO.

WYKONAWCA: MATEUSZ PERNAL

.....

podpis

PROMOTOR: DR INŻ. KRZYSZTOF JURCZUK

.....

podpis

BIAŁYSTOK 2020 r.

Karta dyplomowa

Politechnika Białostocka Wydział Informatyki	Studia stacjonarne studia I stopnia	Numer albumu studenta:101420
		Rok akademicki 2019/2020
Katedra Oprogramowania		Kierunek studiów: informatyka Specjalność: Brak

Mateusz Pernal

TEMAT PRACY DYPLOMOWEJ: Aplikacja internetowa do obsługi system gdt w technologii Python/Django.

Zakres pracy:

1. Zapoznanie z systemem GDT
2. Analiza wymagań aplikacji
3. Projekt i implementacja aplikacji
4. Testy oraz wdrożenie aplikacji.

.....	.....
Imię i nazwisko promotora - podpis	Imię i nazwisko kierownika katedry - podpis

.....	.....	.....
Data wydania tematu pracy dyplomowej - podpis promotora	Regulaminowy termin złożenia pracy dyplomowej	Data złożenia pracy dyplomowej - potwierdzenie dziekanatu

.....	.....
Ocena promotora	Podpis promotora

.....	.....	.....
Imię i nazwisko recenzenta	Ocena recenzenta	Podpis recenzenta

Subject of diploma thesis: Internet application to support the GDT system in Python/Django technology.

Summary

The aim of this thesis was to design, implement and introduce a web application to support GDT (*Global Decision Trees*) system using Python and Django technologies. In the basic version, GDT is a console program for creating decision trees. The requirement of the project was to create a web application allowing to create, delegate and manage the tasks launched by using the GDT system. The developed tool will also provide graphical representation of the obtained results.

The thesis consists of four chapters. First chapter describes the problem and presents the decision trees. It also shows the most popular existing solutions. Second chapter contains an analysis of project requirements and a overview of the technologies that have been used. Third chapter is intended to present the system architecture. It illustrates most important mechanisms and solutions created for the application. It also shows the database schema with a short description of tables. Fourth chapter contains a overview of particular views of the application, but also an example how to use them. In addition, it presents the results of load and manual tests.

Spis treści

Wprowadzenie

Proces myślowy człowieka jest w dużej mierze oparty o pewien schemat podejmowania decyzji. Podejmowane decyzje mają kluczowy wpływ na jego aktualne życie i przyszłość. Wybór najbardziej optymalnego rozwiązania danego problemu wymaga dokładnej analizy dostępnych informacji. Posiadając wystarczającą ilość danych możemy wykorzystać różne algorytmy, które mogą pomóc podjąć właściwy wybór. Mechanizm podejmowania decyzji bezpośrednio dotyczy nie tylko człowieka, a wszystkiego co znajduje się w jego otoczeniu.

Szybki rozwój technologi w XX i XXI wieku prowadzi do produkowania i gromadzenia coraz większej ilości informacji. Firmy starają się wyciągnąć ze zgromadzonych danych możliwe jak najlepsze wnioski. Poddając analizie tak duże zbiory informacji wymagane jest zastosowanie narzędzi uproszczających i przyśpieszających uzyskanie wyników. Prowadzi to do tworzenia algorytmów oraz mechanizmów zarówno obróbki danych, jak i ich analizy w celu osiągnięcia zadowalających rezultatów. W przeciągu ostatnich kilkunastu lat entuzjazm związany z technikami komputerowymi wzrósł gwałtownie i zdominował przemysł. Uczenie maszynowe wraz z analizą danych stanowi bardzo ważny element rozwiązań produkowanych przez firmy. Wspomaga takie technologie, jak rozpoznawanie mowy, pisma czy też autonomiczne samochody i roboty sprzątające. Wszystkie te rozwiązania wymagają przetwarzania ogromnych ilości informacji, w jak najkrótszym czasie oraz podjęcie wystarczająco dobrej decyzji.

W pracy tej rozwijany będzie system do uczenia maszynowego GDT (*Global Decision trees*)[1] tworzony przez pracowników Politechniki Białostockiej. System ten służy do generowania drzew decyzyjnych na podstawie zbioru uczącego. Drzewa generowane są z wykorzystaniem algorytmów ewolucyjnych (metoda alternatywna do algorytmów zachłannych typu *top-down*). Aplikacja GDT jest programem konsolowym. W celu ułatwienia dostępu do platformy GDT większemu gronu użytkowników w niniejszej pracy zostanie zaprojektowana, zaimplementowana oraz wdrożona aplikacja do obsługi systemu GDT z poziomu przeglądarki internetowej.

Cel pracy

Celem pracy jest stworzenie aplikacji webowej umożliwiającej obsługę systemu GDT. Aplikacja ta będzie umożliwiać tworzenie, zlecanie oraz zarządzanie zadaniami uruchamianymi przy pomocy systemu. Podczas tworzenia zadań użytkownik powinien móc ustawić opcje dotyczące, np. wybranego algorytmu oraz jego parametrów. Aplikacja powinna także udostępniać opcje związane z wyświetleniem drzewa wynikowego w postaci graficznej, jego eksport do pliku oraz wgląd do pozostałych wyników uruchomianego algorytmu.

Zakres pracy

Zakres pracy obejmuje:

- Zapoznanie z systemem GDT,
- Analiza wymagań aplikacji,
- Projekt i implementacja aplikacji,
- Testy oraz wdrożenie aplikacji.

Organizacja pracy

Praca została podzielona na cztery główne części. Pierwsze dwa rozdziały zawierają przedstawienie problemu, analizę wymagań i wykorzystane technologie. W dalszej części pracy została omówiona architektura aplikacji wraz z zastosowanymi rozwiązaniami. Natomiast prezentacja stworzonej aplikacji oraz opis testów został przedstawiony w rozdziale 4.

Rozdział 1 zawiera opis mechanizmu tworzenia drzew decyzyjnych. Przedstawia również zagadnienia związane z systemem GDT. Porusza też temat podobnych aplikacji dostępnych w internecie.

Rozdział 2 przedstawia analizę wymagań funkcjonalnych i niefunkcjonalnych tworzonej aplikacji. Został w nim zamieszczony diagram przypadków użycia wraz z ich opisem oraz diagram czynności. Następnie zaprezentowany został schemat rozwiązania. Rozdział kończy przedstawienie użytych technologi podczas tworzenia aplikacji.

Rozdział 3 przedstawia architekturę tworzonej aplikacji. Prezentuje najważniejsze mechanizmy oraz rozwiązania stworzone na potrzeby aplikacji. Przedstawia także schemat bazy danych wraz z krótkim opisem najważniejszych tabel.

Rozdział 4 przedstawia stworzoną aplikację. Zawiera on opis poszczególnych widoków aplikacji, ale także przykładowy sposób ich użycia. Ponadto przedstawia wyniki testów obciążeniowych i manualnych przeprowadzonych przez studentów Wydziału Informatyki Politechniki Białostockiej.

1. Przedstawienie problemu

1.1 Drzewa decyzyjne

Podejmowanie decyzji jest procesem nietrywialnym. Już od początku istnienia ludzkości dobrze podjęte decyzje pozwalały przeżyć wybranym grupom ludzi czy też zwierząt. Wpływ na optymalną decyzję mają informacje, które zostaną poddane analizie, ale także sama metoda analizy. Racjonalny wybór może być wspomagany różnymi algorytmami, czy też wizualną reprezentacją możliwych decyzji. Jedną z form graficznych jest drzewo decyzyjne.

Podstawowymi elementami drzewa są węzły oraz gałęzie. Korzeń drzewa to pierwszy węzeł od którego rozpoczyna się budowa całej struktury zawierającej poszczególne węzły odpowiadające za sprawdzenie pewnego warunku. Natomiast gałęzie pełnią rolę połączenia pomiędzy węzłami na kolejnych poziomach drzewa [2]. Liście są końcowymi wierzchołkami drzewa i zawierają decyzje. Aby otrzymać decyzję konieczne jest przejście całego drzewa od samego korzenia do wynikowego liścia. Rezultatem takiej operacji będzie klasa decyzyjna (w przypadku drzew klasyfikacyjna) lub np. model regresyjny (w przypadku drzew regresyjnych).

1.2 Uczenie maszynowe

W otaczającym nas świecie ilość generowanych oraz gromadzonych informacji nadal przewyższa ilość danych, które można przeanalizować z użyciem obecnych zasobów. Aby analizować tak duże ilości informacji wykorzystywane są najnowsze rozwiązania technologiczne, zarówno na poziomie sprzętu komputerowego oraz oprogramowania. Dzięki zastosowaniu różnych algorytmów przetwarzania danych, klasyfikacji oraz predykcji programy komputerowe posiadają możliwość uczenia się. Tzn. uczenie maszynowe (ang. *machine learning*) jest obszarem sztucznej inteligencji, który zajmuje się wykorzystywaniem komputerowego wspomagania lub podejmowania decyzji. Uczenie maszynowe w przeciągu ostatniej dekady stało się tak popularne, iż w dużej mierze zdominowało przemysł sztucznej inteligencji oraz przyczyniło się do jej rozwoju [3]. Uczenie maszynowe stanowi trzon wielu usług, serwisów i aplikacji. Pod względem algorytmicznym odpowiada za wyniki wyszukiwania w przeglądarkach oraz za rozpoznawanie mowy przez nasze telefony. Jest

także wykorzystywane w dużo trudniejszych zadaniach, jak sterowanie autonomicznymi samochodami, czy też wspomaganie lotów kosmicznych i operacji chirurgicznych w medycynie.

1.3 Drzewa decyzyjne w technikach uczenia maszynowego

Drzewa decyzyjne stanowią jeden z najbardziej rozpowszechnianych mechanizmów w obszarze uczenia maszynowego. Z jednej strony mogą być wykorzystywane w zadaniach z zakresu klasyfikacji, a z drugiej strony również odgrywają ważną rolę w regresji [3]. Mechanizm drzew decyzyjnych pozwala na budowanie modeli na podstawie ogromnych zbiorów uczących. Dodatkowym atutem drzew jest możliwość wizualnego przedstawienia sposobu dojścia do rozwiązania, które będzie zrozumiałe dla osób nie mających do czynienia z uczeniem maszynowym i statystyką.

1.3.1 System GDT

Pracownicy Wydziału Informatyki Politechniki Białostockiej od ponad 20 lat tworzą i rozwijają narzędzie do indukowania drzew decyzyjnych, nazwane GDT (*Global Decision Trees*). Narzędzie te zostało wykorzystane w pracy inżynierskiej. GDT służy do generowania drzew decyzyjnych na podstawie zbiorów wejściowych [1]. System ten jest zaimplementowany w języku C++. Podstawowa jego wersja jest programem konsolowym. Całe rozwiązanie jest unikalne, a głównym założeniem jest wykorzystanie algorytmów ewolucyjnych w procesie indukcji drzew decyzyjnych. Używając algorytmów ewolucyjnych budowane drzewa są bardziej globalne (trudniej wpaść w minimum lokalne) niż w klasycznym podejściu. Skutkuje to możliwością osiągnięcia dokładniejszych i lepszych wyników [4]. Algorytmy ewolucyjne wzorują się na ewolucji biologicznej [5]. Podczas inicjalizacji parametrów algorytmu należy podać takie parametry jak wielkość populacji, prawdopodobieństwo mutacji czy też krzyżowania się danych osobników. Wartości parametrów algorytmu są określane w pliku konfiguracyjnym opartym o strukturę XML (ang. *Extensible Markup Language*), który jest zarazem plikiem wejściowym do aplikacji GDT. Oprócz pliku z opcjami należy także określić pliki ze zbiorem danych:

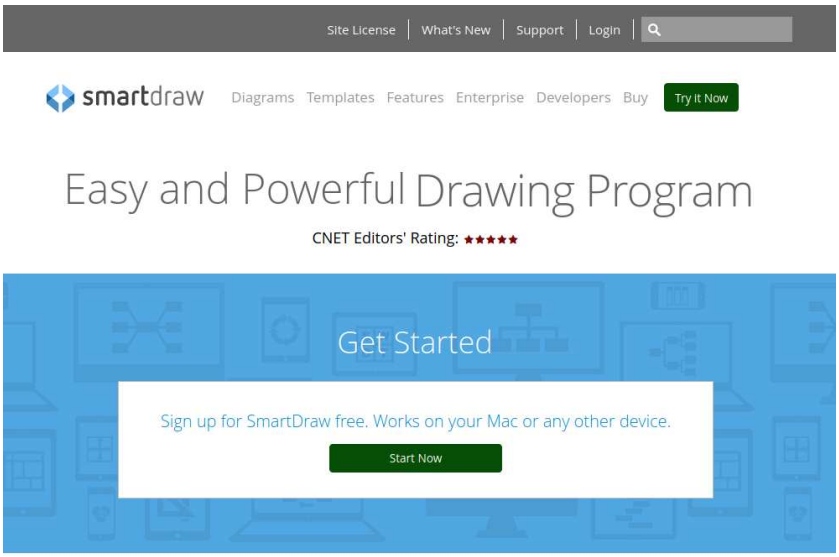
- *.data - plik zawierający dane treningowe,
- *.test - plik zawierający dane testowe,

- *.names - plik określających nazwy klas oraz rodzaj zmiennych.

Wykorzystując określony zbiór trenujący oraz wartości parametrów algorytmu w pliku XML, system GDT indukuje drzewa decyzyjne. Aplikacja zapisuje do plików tekstowych statystyki wynikowe drzewa oraz użyte ustawienia parametrów.

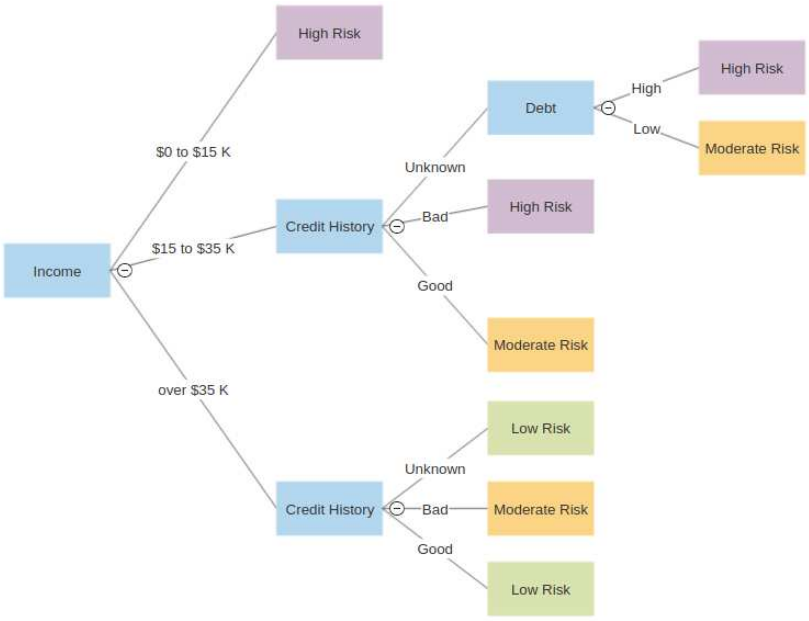
1.4 Istniejące rozwiązania

Aktualnie na rynku można znaleźć różne aplikacje pozwalające na budowanie drzew decyzyjnych. Aplikacje te różnią się miedzy sobą zakresem funkcjonalności. Rozwiązania internetowe głównie są nastawione na zarobek, ale oferują też darmowe wersje z pewnymi ograniczeniami. Istnieją też liczne rozwiązania dla programistów w postaci np. bibliotek dla wielu popularnych języków programowania. Takie biblioteki umożliwiają poprzez wykorzystanie modułów stworzenie podstawowych modeli uczenia maszynowego w tym drzew decyzyjnych. Niestety ich wykorzystanie wymaga przynajmniej podstawowej wiedzy z zakresu programowania. Dodatkowo chcąc osiągnąć bardzo wydajne rozwiązania zazwyczaj należy znać szczegóły implementacji biblioteki. Możemy także spotkać aplikacje desktopowe. Wiele takich programów jest rozwijanych przez zespoły naukowe na uniwersytetach. Do wykonania obliczeń wymagają dobrej jakości sprzętu komputerowego, który zapewni odpowiednią moc obliczeniową.



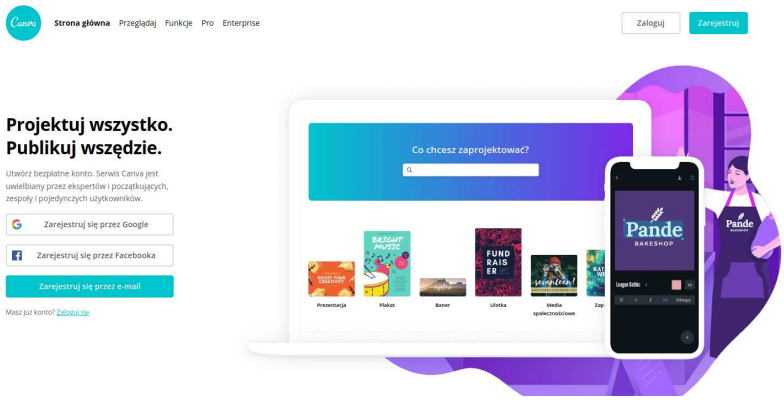
Rysunek 1.1: Strona główna aplikacji *SmartDraw* [6].

Aplikacja internetowa *SmartDraw* jest według autora pracy jedną z wygodniejszych platform do tworzenia drzew decyzyjnych [6]. Użytkownik może za darmo założyć

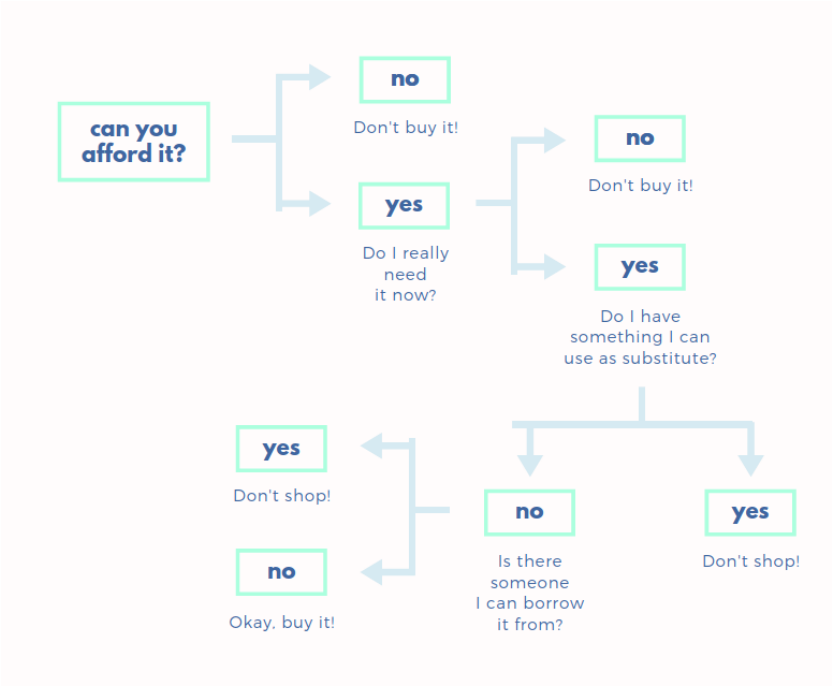


Rysunek 1.2: Drzewo stworzone w aplikacji *SmartDraw*, źródło: opracowanie własne.

konto oraz używać aplikacji bez abonamentu przez okres próbny. Widok strony głównej został przedstawiony na Rys. 1.1. Platforma ponadto udostępnia funkcjonalność tworzenia innych rodzajów diagramów. Proces budowy diagramu (np. w postaci drzewa) polega na przeciąganiu i łączeniu bloków. Istnieje również możliwość wczytania struktury drzewa z pliku o rozszerzeniu *csv (ang. *comma-separated values*). Diagramy są prezentowane w czytelny i przejrzysty sposób (Rys. 1.2). Ukończony diagram można wyeksportować do pliku graficznego lub dokumentu programu MS Word. Aplikacja niestety nie umożliwia budowy drzewa używając metody uczenia, jedynie pozwala na graficzne przedstawienie wcześniej przygotowanej struktury.



Rysunek 1.3: Strona główna aplikacji *Canva* [7].

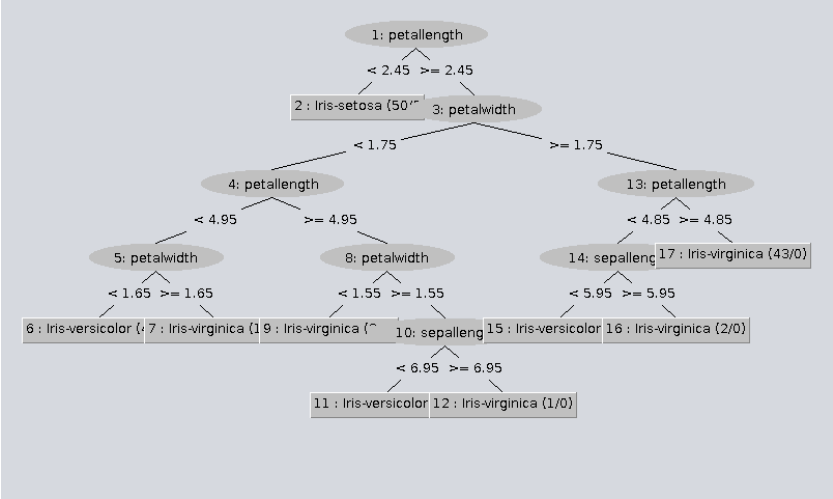


Rysunek 1.4: Drzewo stworzone w aplikacji *Canva*, źródło: opracowanie własne.

Canva to kolejne narzędzie umożliwiające tworzenie drzew decyzyjnych przy pomocy przeglądarki internetowej [7]. Dostęp do platformy wymaga założenia konta. Aplikacja także posiada rozszerzony, płatny pakiet funkcjonalności dla firm oraz osób prywatnych. Strona główna aplikacji została przedstawiona na Rys. 1.3. Użytkownik ma możliwość wizualnego przedstawienia metodą przeciągnij i upuść (ang. *drag'n drop*). W aplikacji jednak brakuje opcji indukcji drzewa decyzyjnego z wykorzystaniem zbioru uczącego. Tworzone drzewa można wzbogacić o liczne walory wizualne i dostępne gotowe motywy (Rys. 1.4). Głównymi odbiorcami aplikacji są reklamodawcy oraz osoby prowadzące rozbudowaną działalność w serwisach społecznościowych.

Weka jest aplikacją desktopową stworzoną przez naukowców zajmujących się tematami uczenia maszynowego na Uniwersytecie Waikato w Nowej Zelandii. Aplikacja została zaimplementowana w technologi JAVA SE. Pozwala to na jej uruchomienie na różnych systemach operacyjnych. Główne okno aplikacji zostało zaprezentowane na Rys. 1.5. Ponadto naukowcy zaimplementowali liczne tzw. nakładki, umożliwiające korzystanie ze stworzonych mechanizmów za pośrednictwem innych języków programowania. W aplikacji użytkownik ma dostęp do dużej ilości gotowych algorytmów uczenia maszynowego, łącznie z algorytmami indukcji drzew decyzyjnych. Tworzenie nowego eksperymentu zaczyna się od wybrania zestawu danych. Użytkownik początkujący może skorzystać z przykładowych

Rysunek 1.5: Okno główne aplikacji *Weka*, źródło: opracowanie własne.



Rysunek 1.6: Drzewo stworzone w aplikacji *Weka*, źródło: opracowanie własne.

plików. Po wczytaniu zbioru wejściowego istnieje możliwość wizualizacji danych oraz ich wstępnej obróbki. W kolejnym kroku użytkownik wybiera algorytm budowy klasyfikatora. Do wyboru jest kilka różnych algorytmów związanych z tworzeniem drzew decyzyjnych. Czas obliczeń zależy od ilości danych, wybranego algorytmy oraz posiadanych zasobów obliczeniowych. Rezultaty eksperymentu są przedstawiane w postaci tekstowej. Istnieje także możliwość wyświetlenia drzewa graficznie. Przykładowe drzewo decyzyjne stworzone za pośrednictwem programu zostało przedstawione na Rys. 1.6.

Aplikacja tworzona w ramach pracy dyplomowej jest aplikacją oryginalną. Po pierwsze ze względu na wykorzystany system GDT. Po drugie, będzie to aplikacja internetowa, która umożliwi indukowanie drzew decyzyjnych z wykorzystaniem algorytmów

ewolucyjnych. To co będzie ją odróżniać to moduł do zarządzania zadaniami, które będą uruchamiać się zdalnie, przez co użytkownik końcowy nie będzie musiał posiadać znaczących zasobów pamięciowych i obliczeniowych. Budowane narzędzie pozwoli także na graficzną i interaktywną reprezentacje otrzymanych wyników.

2. Wizja aplikacji

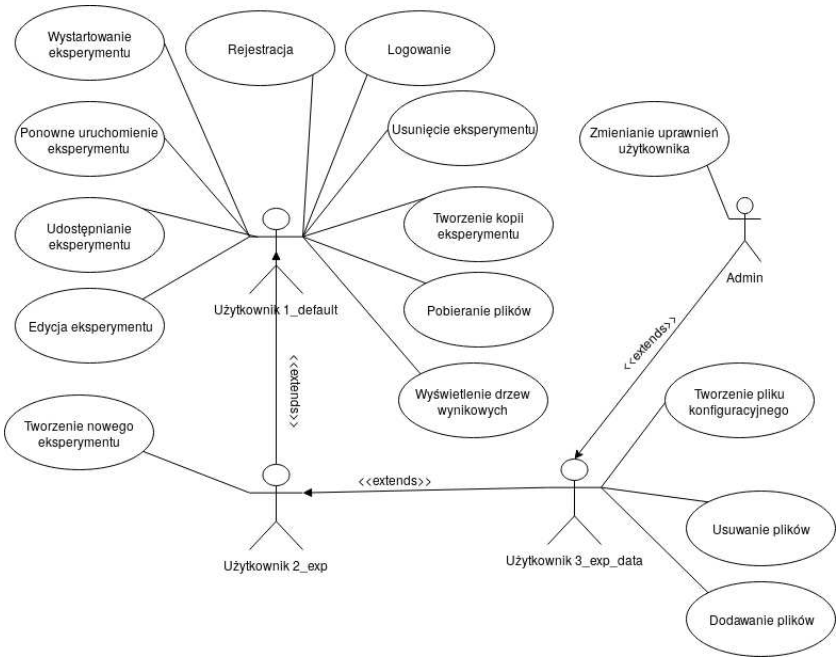
2.1 Wymagania funkcjonalne

Tworzenie aplikacji należało zacząć od nakreślenia zakresu funkcjonalności, które aplikacja będzie udostępniać użytkownikom. Podstawowym zadaniem, budowanej aplikacji jest możliwość przeprowadzania eksperymentów przy pomocy systemu GDT z poziomu aplikacji internetowej. Kolejnym ważnym aspektem jest wariant zarządzania, wyświetlania, udostępniania oraz edycji poszczególnych eksperymentów (zadań). Każdy z użytkowników powinien widzieć poszczególne zadania, które zostały ukończone, są w trakcie wykonywania lub czekają w kolejce. Aplikacja powinna również w przejrzysty sposób wyświetlać wyniki, zarówno wynikowe drzewo decyzyjne oraz statystyki obliczeń. Po uruchomieniu zadania, użytkownikowi zostanie wyświetlony pasek postępu oraz oszacowana długość trwania całego zadania. Zadanie będzie można anulować w dowolnym momencie. Użytkownik będzie posiadać możliwość zarządzania plikami wejściowymi do zadania oraz plikami z wynikami. Dla użytkowników początkujących zostanie stworzona opcja budowy podstawowych plików konfiguracyjnych, bez wgłębiania się w bardziej zaawansowane parametry eksperymentu. Dostęp do funkcji aplikacji będzie wymagał założenia konta. Nowo założone konto będzie miało domyślnie ograniczone możliwości. Natomiast możliwość rejestracji oraz logowania będzie ogólnodostępna.

Użytkownik będzie mógł posiadać jedną z dostępnych ról. Role będą definiowały dostęp do poszczególnych funkcjonalności aplikacji. Zarządzanie tymi uprawnieniami będzie się odbywać poprzez panel administratora. Administrator aplikacji dodatkowo może modyfikować oraz usuwać konta użytkowników. Co więcej z interfejsu admina będzie istniała możliwość edycji rekordów bazy danych oraz edycja uprawnień do poszczególnych eksperymentów.

Biorąc pod uwagę perspektywę udostępniania przez użytkownika eksperymentów innemu użytkownikowi, ważnym aspektem będzie możliwość ograniczenia części akcji wykonywanych na eksperymencie. Aplikacja nie pozwoli na zablokowanie wyświetlania drzewa wraz z wynikami. Natomiast reszta funkcjonalności możliwych do wykonania na zadaniu, takich jak uruchamianie, kopiowanie, edycja, usuwanie czy też pobieranie plików wejściowych lub wyjściowych może zostać ograniczona. Użytkownik posiadający

udostępniony eksperyment z pewnymi ograniczeniami, może udostępnić go dalej jeśli posiada nadane prawa do udostępniania. Przy czym nie może rozszerzyć uprawnień uprzednio zablokowanych.



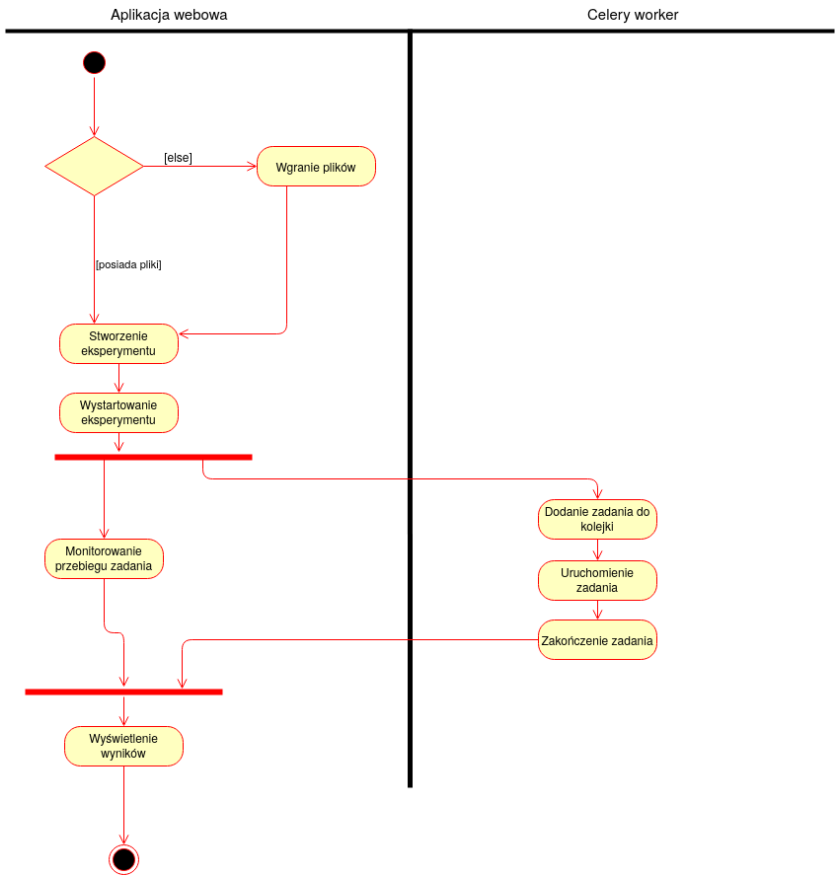
Rysunek 2.1: Diagram przypadków użycia, źródło: opracowanie własne.

Na rysunku 2.1 przedstawiono funkcjonalności w postaci diagramu przypadków użycia. W aplikacji zostały wyszczególnione trzy role dostępne do uzyskania dla użytkownika oraz rola administratora systemu. Wszystkie przypadki użycia oprócz logowania i rejestracji są dostępne tylko dla użytkowników zalogowanych. Każdy nowy użytkownik musi założyć konto, aby mieć dostęp do aplikacji. Nowo powstałe konta otrzymują uprawnienia na domyślnym poziomie „1_default", a wyższe poziomy uprawnień mogą zostać nadane przez administratora. Kolejne role rozszerzają możliwości użytkownika pod względem ilości akcji do wykonania. Poziom „2_exp" pozwala na tworzenie nowych eksperymentów, przy czym tylko najwyższy poziom uprawnień „3_exp_data" może autoryzować do wgrywania plików do aplikacji. Przebieg czynności związanych ze stworzeniem nowego eksperymentu oraz wyświetleniem wyników został przedstawiony na Rys. 2.2. W dalszej części tego podrozdziału przedstawione zostały opisy trzech wybranych przypadków użycia.

Opis przypadku użycia „Tworzenie nowego eksperymentu":

1. Aktor

    - Użytkownik.



Rysunek 2.2: Diagram czynności tworzenia i uruchomiania eksperymentu, źródło: opracowanie własne.

2. Warunki początkowe

   - Aktor jest zalogowany oraz posiada uprawnienia przynajmniej na poziomie „2_exp”.

3. Zdarzenie inicjujące

   - Naciśnięcie przycisku „New experiment” nad listą wszystkich eksperymentów użytkownika.

4. Przebieg w krokach

   - Aplikacja przechodzi do formularza tworzenia nowego eksperymentu,
   - Użytkownik wypełnia i zatwierdza formularz.

5. Przebiegi alternatywne

   - Użytkownik nie uzupełnia wszystkich pól formularza, aplikacja wyświetla powiadomienie o pustych polach.

6. Sytuacje wyjątkowe

    - Użytkownik nie posiada żadnych plików wgranych do aplikacji. Powoduje to, że pola formularza zawierające pliki są puste. Uniemożliwia to stworzenie nowego eksperymentu, a aplikacja wyświetla powiadomienie o pustych polach przy podjętej próbie zatwierdzenia.

7. Warunki końcowe

    - System przekierowuje użytkownika do listy z eksperymentami, a na liście znajduje się nowo utworzony eksperyment.

8. Zależności czasowe

    - Częstotliwość wykonywania: Około 20 razy dziennie na każdego użytkownika,
    - Typowy czas realizacji: 8 sekund.

Opis przypadku użycia „Wystartowanie eksperymentu”:

1. Aktor

    - Użytkownik.

2. Warunki początkowe

    - Aktor jest zalogowany oraz posiada stworzony eksperyment.

3. Zdarzenie inicjujące

    - Naciśnięcie przycisku „Show” w liście eksperymentów na elemencie, którego status to „Created”.

4. Przebieg w krokach

    - Aplikacja przechodzi do podglądu szczegółów wybranego eksperymentu,
    - Użytkownik klika przycisk „Start” znajdujący się na pasku możliwych czynności,
    - Aplikacja przekierowuje użytkownika do listy eksperymentów.

5. Przebiegi alternatywne

    - Po wystartowaniu eksperymentu nastąpił błąd i jest to sygnalizowane zmianą statusu na „Error”, a w szczegółach eksperymentu można podejrzeć wiadomość z błędem.

6. Sytuacje wyjątkowe

    - Użytkownikowi nie posiada praw do wystartowania konkretnego eksperymentu i w panelu akcji nie wyświetla się przycisk „Start”.

7. Warunki końcowe

    - Eksperyment zmienił swój status na „In queue” lub „Running”, a po przejściu do szczegółów wyświetla się pasek postępu oraz szacowany czas oczekiwania na zakończenie.

8. Zależności czasowe

    - Częstotliwość wykonywania: Około 20 razy dziennie na każdego użytkownika,
    - Typowy czas realizacji: 8 sekund.

Opis przypadku użycia „Wyświetlenie drzewa wynikowego”:

1. Aktor

    - Użytkownik.

2. Warunki początkowe

    - Aktor jest zalogowany oraz posiada ukończony eksperyment.

3. Zdarzenie inicjujące

    - Naciśnięcie przycisku „Show” w liście eksperymentów na elemencie, którego status to „Finished”.

4. Przebieg w krokach

    - Aplikacja przechodzi do podglądu szczegółów wybranego eksperymentu, a na samym dole karty wyświetlają się linki do drzew decyzyjnych,

- Użytkownik klika w link do drzewa decyzyjnego.

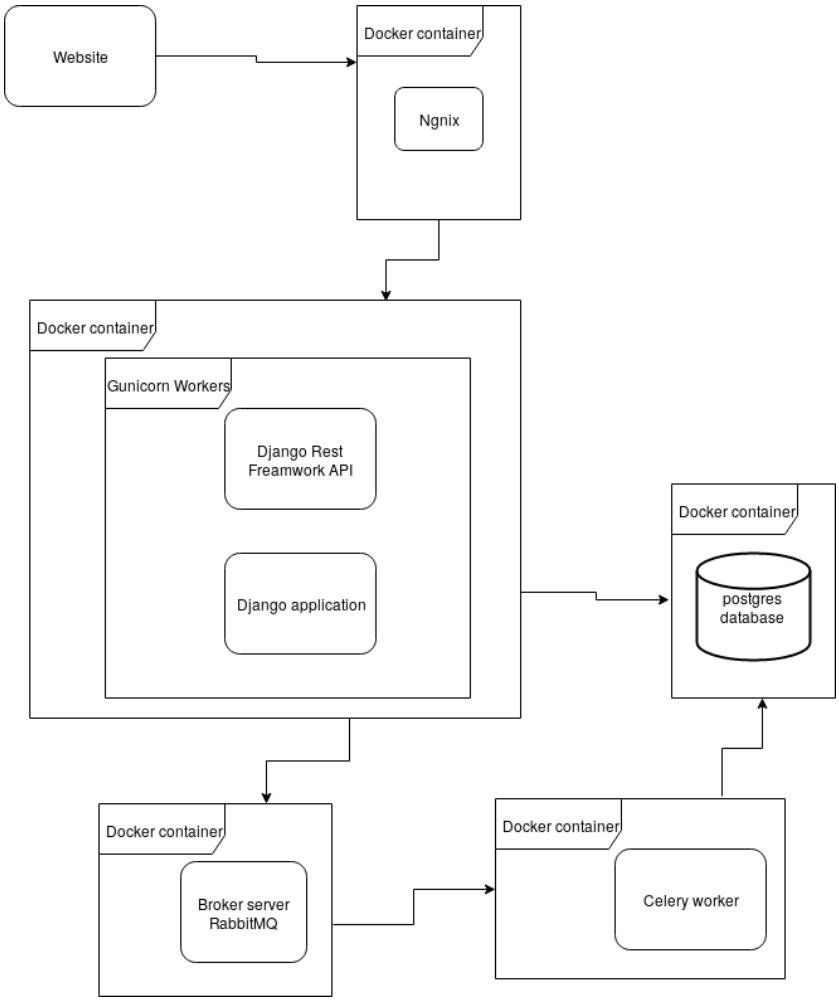
5. Przebiegi alternatywne
   - Brak.
6. Sytuacje wyjątkowe
   - Brak.
7. Warunki końcowe
   - Aplikacja wyświetliła drzewo decyzyjne wraz ze statystykami.
8. Zależności czasowe
   - Częstotliwość wykonywania: Około 30 razy dziennie na każdego użytkownika,
   - Typowy czas realizacji: 10 sekund.

2.2 Wymagania niefunkcjonalne

Aplikacja zostanie podzielona na dwa oddzielne komponenty: jeden odpowiadający za aplikację internetową, drugi natomiast za zarządzanie zadaniami na serwerze: kolejkowanie i uruchamianie zadań w systemie GDT. Stworzone oprogramowanie powinno pozwolić na zarządzanie całością aplikacji z poziomu przeglądarki internetowej. Poszczególne elementy części serwerowej aplikacji powinny być odporne na błędy, umożliwiać łatwy mechanizm wdrożenia oraz restartu modułów w przypadku takiej potrzeby. Zostanie to zapewnione poprzez konteneryzacje aplikacji. Zależności pomiędzy konkretnymi kontenerami utworzonymi przy pomocy oprogramowania Docker zostały przedstawione na Rys. 2.3. Dzięki takiemu rozwiązaniu wdrożenie aplikacji, czy też zmiana któregoś z komponentów na przykład serwera bazy danych, wymaga małego pokładu pracy i może zostać wykonana półautomatycznie. Aby uruchomić aplikację na serwerze niezbędna będzie platforma Docker. Wykorzystane technologie zostaną opisane w kolejnym podrozdziale.

2.3 Wykorzystane technologie

Aplikacja zostanie zaimplementowana z użyciem języka programowania Python (wersja 3.7.2), który jest rozpowszechnioną technologią wśród aplikacji internetowych.



Rysunek 2.3: Architektura systemu, źródło: opracowanie własne.

Wspiera on kilka różnych paradygmatów programowania takich jak programowanie funkcyjne, proceduralne i obiektowe [8]. Po stronie serwera będzie udostępniony interfejs programistyczny (ang. *API*) w architekturze REST (*Representational State Transfer*) z wykorzystaniem Django oraz Django REST Framework. Natomiast interfejs graficzny strony będzie zaprojektowany przy pomocy JavaScriptu [9] oraz biblioteki ReactJS [10].

Jedną z najważniejszych cech języka Python jest interpretowalność kodu źródłowego zamiast jego kompilacja [11]. Umożliwia on pisanie zarówno skryptów systemowych, jak i pełnoprawnych programów. Kolejną jego charakterystyczną cechą jest dynamiczne typowanie czyli tak zwany *duck typing*. Określa to sposób przypisywania typów do wartości przechowywanych w zmiennych. Typy te są określane dynamicznie podczas działania programu, w odróżnieniu od typowania statycznego, gdy wartości poszczególnych zmiennych muszą być jasno podane przed procesem kompilacji. Takie podejście do przypisywania rodzajów na pewno przyśpiesza pracę, ale może też powodować pewne

problemy. W trakcie implementacji, programista musi sam pamiętać jaki typ w danym momencie ma zmienna. Proces ten ułatwia biblioteka wbudowana w język o nazwie „Typing". Umożliwia ona w prosty sposób wprowadzenie namiastki typowania statycznego w postaci sprawdzania i podpowiedzi.

Do implementacji części biznesowej aplikacji (czyli stronie serwera) zostanie wykorzystany framework Django (wersja 2.2.6), który jest jednym z najbardziej popularnych rozwiązań webowych w języku Python. Charakteryzuje się on prostotą implementacji (przynajmniej w początkowej fazie aplikacji) oraz podziałem aplikacji na komponenty. Zarazem narzuca on pewną strukturę projektu, która zapewnia czystość kodu oraz możliwość ponownego użycia wcześniej opracowanych modułów [12]. Społeczność zebrana wokół tej platformy, dynamicznie rozwija nowe rozwiązania, które są udostępniane dla szerszego grona odbiorców. Dzięki temu istnieje łatwy dostęp do wysokiej jakości modułów bezpieczeństwa, czy też wsparcie techniczne przy występujących problemach. Kolejnym argumentem, który przemawiał za wybraniem tej technologi był wbudowany panel administratora, dostarczany wraz z całą platformą. Po konfiguracji umożliwia on nie tylko zarządzanie użytkownikami, ale także edycję rekordów w bazie danych z poziomu przeglądarki internetowej.

W celu udostępnienia REST API (ang. *Representational State Transfer Application Programming Interface*) dla interfejsu graficznego został użyty Django REST Framework (*DRF*), który wraz z podstawową wersją Django stanowi trzon aplikacji. DRF jest narzędziem używanym oraz rozwijanym przez takie rozpoznawalne marki jak Mozilla, Red Hat, Heroku [13]. Świadczy to nie tylko o popularności tego rozwiązania, ale też o jakości jego wykonania. Cały framework skupia się na dostarczeniu programiście zestawu narzędzi do budowy interfejsu restowego. W skład takiej paczki wchodzą serializatory (*serializers*), widoki (*views*), rutery (*router*) oraz wiele innych pomocniczych obiektów. Komunikacja z takim interfejsem programistycznym odbywa się za pomocą metod protokołu HTTP (*Hypertext Transfer Protocol*. Kolejność kroków pracy API możemy określić w następujący sposób:

1. Klient tworzy zapytanie i uzupełnia je o potrzebne dane,
2. Następnie wysłane jest zapytanie pod konkretny adres,
3. Serwer przetwarza żądanie klienta oraz wysyła odpowiedź,

4. Klient otrzymuje rezultat.

Do zapisu informacji związanych z działaniem aplikacji zostanie wykorzystana relacyjna baza danych PostgreSQL (wersja 10.3). Jest to rozwiązanie pod licencją *Open Source* i szeroko stosowane również z aplikacjami opartymi o technologie Django. Bazę danych można w łatwy sposób podpiąć pod panel administratora, ale także uzyskać dostęp z poziomu kodu aplikacji. W bazie danych będą gromadzone konta użytkowników oraz informacje o stworzonych eksperymentach wraz ze ścieżkami do plików.

Biblioteka ReactJS łącznie z JavaScriptem pozwoli na zaimplementowanie funkcjonalności po stronie klienta. Zapewnią one strukturę projektu, która oferuje czystość kodu, czytelność oraz wygodę użytkowania. Umożliwią one na wstawianie fragmentów kodu hmtl do kodu JavaScript za pośrednictwem języka JSX. Początkowo biblioteka ReactJS została stworzona dla potrzeb wewnętrznych firmy Facebook, ale z czasem została udostępniona innym twórcom [10]. ReactJS jest aktualnie wykorzystywany przez wielkie korporacje takie jak Netflix czy Uber, a sam projekt jest czwartym najpopularniejszym repozytorium na GitHub [14]. W celu połączenia interfejsu graficznego z logiką biznesową zostanie wykorzystana biblioteka Axios [15]. Cechuje się kompatybilnością wsteczną ze starszymi wersjami przeglądarek internetowych. Umożliwia ona budowę zapytań http oraz ich realizację. Aktualnie jest to najpopularniejsze rozwiązanie w języku JavaScript.

W celu uzyskania pełnej asynchroniczności podczas uruchamiania eksperymentów w systemie GDT, do obsługi kolejki zadań zostanie wykorzystana biblioteka Celery [16]. Takie rozwiązanie jest łatwe w integracji z innymi platformami programistycznymi. Głównym założeniem działania tej biblioteki polega na stworzeniu kolejki z zadaniami, które następnie zostaną przypisane i wykonane przez wolnego „robotnika". Liczba „robotników" działających w aplikacji może zostać określona jako jeden z parametrów uruchomienia. Komunikacja pomiędzy Celery, a aplikacją w technologi Django odbywa się przy pomocy brokera wiadomości (ang. *message broker*), który odpowiada za przesył informacji pomiędzy dwoma komponentami. Przykładem takiego oprogramowania może być broker RabbitMQ i on zostanie wykorzystany podczas implementacji aplikacji w ramach pracy dyplomowej [17]. Oprogramowanie te osiąga bardzo dobre wyniki wydajnościowe w porównaniu z produktami konkurencyjnymi.

Gunicorn jest serwerem HTTP aplikacji Django, umożliwiającym zdefiniowanie liczby „robotników" realizujących zapytania. Jego głównym założeniem jest realizacja

wszystkiego co się dzieje pomiędzy serwerem, a aplikacją webową. Dodatkowym atutem jest minimalna ilość zasobów, które zużywa oraz duża szybkość działania [18]. Będzie on wykorzystywany jako serwer produkcyjny. Gunicorn dobrze współpracuje z Nginx i umożliwi łatwe wdrożenie aplikacji. Nginx jest serwerem WWW i posłuży do przekierowania przychodzących pod adres domeny „decisiontree.pl" żądań prosto do aplikacji działającej pod serwerem Gunicorn.

Jedną z najważniejszych części całej architektury jest konteneryzacja poszczególnych elementów. W tym celu zostaną wykorzystane kontenery stworzone przy pomocy platformy Docker. Jest to oprogramowanie umożliwiający wirtualizacje oraz podział projektu na oddzielne komponenty [19]. Stosując takie rozwiązanie w dowolnej chwili można podmieniać kontenery aplikacji, zmieniając dynamicznie ich wersje oraz łatwo restartować tylko te elementy, które tego wymagają. Cały system zostanie podzielony na pięć pracujących obok siebie instancji tworzących wspólnie całość. Podział ten jest przedstawiony na Rys. 2.3.

Bibliografia

[1] Kretowski M. *Evolutionary Decision Trees in Large-Scale Data Mining*. Studies in Big Data, vol. 59, Springer.

[2] Lucid Software Inc. Lucidchart. `https://www.lucidchart.com/pages/decision-tree`, stan z 25.11.2019 r.

[3] Aurélien Géron. *Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow*. Helion S.A., 2018.

[4] Kretowski M. Jurczuk K., Czajkowski M. *Evolutionary Induction of Decision Tree for Large Scale Data*. Soft Computing, vol. 21: 7363-79.

[5] Prateek Joshi. *Artificial Intelligence with Python*. Packt Publishing, 2017.

[6] SmartDraw. Smartdraw. `https://www.smartdraw.com/`, stan z 17.12.2019 r.

[7] Canva. Canva. `https://www.canva.com/`, stan z 17.12.2019 r.

[8] Mark Lutz. *Python. Wprowadzenie. Wydanie IV*. Helion S.A., 2010.

[9] Mozilla and individual contributors. Javascript. `https://developer.mozilla.org/pl/docs/Web/JavaScript`, stan z 10.12.2019 r.

[10] Facebook Inc. Reactjs. `https://reactjs.org/`, stan z 10.12.2019 r.

[11] Python Software Foundation. Python. `https://www.python.org/`, stan z 08.12.2019 r.

[12] Django Software Foundation. Django. `https://www.djangoproject.com/`, stan z 10.12.2019 r.

[13] Collaboratively funded project. Django rest framework. `https://www.django-rest-framework.org/`, stan z 10.12.2019 r.

[14] Inc. GitHub. Github. `https://github.com/search?o=desc&q=stars%3A%3E1&s=stars&type=Repositories`, stan z 10.12.2019 r.

[15] Axios. Axios. https://github.com/axios/axios, stan z 10.12.2019 r.

[16] Ask Solem and contributors. Celery project. http://www.celeryproject.org/, stan z 10.12.2019 r.

[17] Inc. Pivotal Software. Rabbitmq. https://www.rabbitmq.com/, stan z 10.12.2019 r.

[18] Benoit Chesneau and contributors. Gunicorn. https://gunicorn.org/, stan z 10.12.2019 r.

[19] Docker Inc. Docker. https://www.docker.com/, stan z 10.12.2019 r.

[20] The PostgreSQL Global Development Group. Postgresql. https://www.postgresql.org/, stan z 10.12.2019 r.

[21] Inc. NGINX. Nginx. https://www.nginx.com/, stan z 10.12.2019 r.

[22] Locustio. Locust. https://locust.io/, stan z 17.12.2019 r.

[23] Machine Learning Group at the University of Waikato. Weka. https://www.cs.waikato.ac.nz/ml/weka/, stan z 17.12.2019 r.

Spis tabel

Spis rysunków

Spis listingów