POLITECHNIKA BIAŁOSTOCKA

WYDZIAŁ INFORMATYKI

KATEDRA OPROGRAMOWANIA

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 2019 r.

Karta dyplomowa

<table>
<tr><td>Politechnika Białostocka
Wydział Informatyki

Katedra Oprogramowania</td><td>Studia stacjonarne
studia I stopnia</td><td>Numer albumu studenta:101420
Rok akademicki 2019/2020
Kierunek studiów: informatyka
Specjalność: Brak</td></tr>
<tr><td colspan="3">Mateusz Pernal
TEMAT PRACY DYPLOMOWEJ: Aplikacja internetowa do obsługi system gdt w technologii Python/Django.
Zakres pracy:
1. Zakres 1
2. Zakres 2
3. Zakres 3

..... Imię i nazwisko promotora - podpis
..... Imię i nazwisko kierownika katedry - podpis</td></tr>
<tr><td>..... Data wydania tematu pracy dyplomowej - podpis promotora</td><td>..... Regulaminowy termin złożenia pracy dyplomowej</td><td>..... Data złożenia pracy dyplomowej - potwierdzenie dziekanatu</td></tr>
<tr><td colspan="3">..... Ocena promotora
..... Podpis promotora</td></tr>
<tr><td>..... Imię i nazwisko recenzenta</td><td>..... Ocena recenzenta</td><td>..... Podpis recenzenta</td></tr>
</table>

Thesis topic:

Temat po angielsku.

Summary

Abstrakt po angielsku

Spis treści

Wprowadzenie

Tu będzie wstęp

Cel pracy

Celem pracy jest stworzenie aplikacji webowej umożliwiającej obsługę systemu GDT (*Global Decision trees*) służącego do generowania drzew decyzyjnych. Strona internetowa umożliwi tworzenie oraz zarządzanie zadaniami uruchamianymi przy pomocy systemu. Jedną z ważniejszych cech programu powinno być danie użytkownikowi możliwość ustawienia parametrów konfiguracyjnych przed wystartowaniem zadania. Aplikacja internetowa powinna także udostępniać opcje związane z wyświetleniem drzewa w postaci graficznej oraz przedstawieniem wyników eksperymentu.

Zakres pracy

Zakres pracy obejmuje:

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

Organizacja pracy

Tu będzie organizacja pracy/ zawartość pracy

1. Przedstawienie problemu

1.1 Drzewa decyzyjne

Podejmowanie decyzji jest procesem myślowym, który od początku istnienia ludzkości stwarza pewne trudności, a polega on na wybraniu najlepszego rozwiązania z dostępnych. 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 w postaci diagramu. Sam diagram może przybrać formę graficzną drzewa decyzyjnego.

Podstawowymi elementami drzewa są korzeń, gałęzie, węzły oraz liście. Korzeniem jest decyzja od którego rozpoczyna się budowa całej struktury zawierającej poszczególne węzły odpowiadające za sprawdzenie pewnego warunku. Natomiast gałęzie pełnią rolę połączenia wszystkich elementów [11]. Liście są krańcowymi wierzchołkami drzewa i określają wybraną decyzje. Podczas próby określenia decyzji, należy poddać klasyfikacji posiadane dane, aby to osiągnąć konieczne jest przejście całego drzewa od samego korzenia do wynikowego liścia. Rezultatem takiej operacji będzie klasa definiująca decyzję.

1.2 Uczenie maszynowe

W otaczającym nas świecie ilość informacji produkowanych przez otoczenie oraz zbieranych przez firmy czy instytucje nadal przewyższa ilość danych, które można przeanalizować z użyciem obecnych zasobów. W celu wyciągnięcia wniosków z takiej ilości danych wykorzystuje się liczne rozwiązania technologiczne. Dzięki zastosowaniu różnych algorytmów przetwarzania danych, klasyfikacji oraz predykcji programy komputerowe posiadają możliwość uczenia się. Kierunek nauki, który zajmuje się tą dziedziną nazywamy uczeniem maszynowym. W ciągu ostatnich dziesięciu lat entuzjazm związany z wykorzystywaniem tej technologi wzrósł gwałtownie i w dużej mierze zdominował przemysł, ale również przyczynił się do jej rozwoju [8]. Uczenie maszynowe stanowi trzon wielu usług, serwisów i aplikacji. Pod względem technologicznym odpowiada za wyniki wyszukiwania w przeglądarkach, za rozpoznawanie mowy przez nasze telefony, ale także jest odpowiedzialne za prowadzenie autonomicznych samochodów.

1.3 Drzewa decyzyjne w technikach uczenia maszynowego

Drzewa decyzyjne stanowią jedne z bardziej wszechstronnych algorytmów w dziedzinie uczenia maszynowego. Z jednej strony mogą być wykorzystywane w zadaniach z zakresu klasyfikacji, a z drugiej strony również odgrywają ważną rolę w regresji [8]. Z ich pomocą możemy uzyskać potężne modele i narzędzia zdolne do uczenia się ze złożonych zbiorów danych. Dodatkowym atutem drzew jest możliwość wizualnego przedstawienia rozwiązania, które będzie zrozumiałe dla osób nie mających do czynienia z uczeniem maszynowym lub ze statystyką. Z racji wzrostu popularności tej technologi zwiększyły się nakłady pracy naukowej w celu osiągnięcia coraz to lepszych i bardziej optymalnych algorytmów pod względem wydajnościowym.

1.3.1 System GDT

Pracownicy Politechniki Białostockiej również mają wkład w budowę takich rozwiązań. Autorski system GDT (*Global Decision Trees*), który jest wykorzystywany w aplikacji inżynierskiej, służy do generowania modelu drzewa decyzyjnego na podstawie zbiorów wejściowych. Ten system jest zaimplementowany w języku c++ oraz jest skompilowany do pliku wykonywalnego, aby umożliwić jego uruchomianie z poziomu konsoli systemu operacyjnego. Całe rozwiązanie jest unikalne, a głównym założeniem jest wykorzystanie algorytmów genetycznych. Z ich pomocą przestrzeń rozwiązań danego problemu jest większa niż w klasycznym podejściu, co skutkuje możliwością osiągnięcia dokładniejszych i lepszych wyników. Metody pracy algorytmów genetycznych w dużej mierze odwzorowują działania samej natury [13]. Podczas definiowania pracy algorytmu należy podać takie parametry jak wielkość populacji, prawdopodobieństwo mutacji czy też krzyżowania się danych osobników. Wartości tych parametrów i innych są określane w pliku konfiguracyjnym opartym o strukturę XML, który jest zarazem plikiem wejściowym do aplikacji GDT. System oprócz tego pliku wykorzystuje pliki z konkretnymi rozszerzeniami:

- *.data - plik zawierający dane treningowe,
- *.test - plik zawierający dane testowe,
- *.names - plik określających nazwy klas oraz rodzaj zmiennych.

Na podstawie tych danych aplikacja GDT może stworzyć model drzewa decyzyjnego, którego przedstawienie jest zapisywane w pliku tekstowym.

1.4 Istniejące rozwiązania

2. Wizja aplikacji

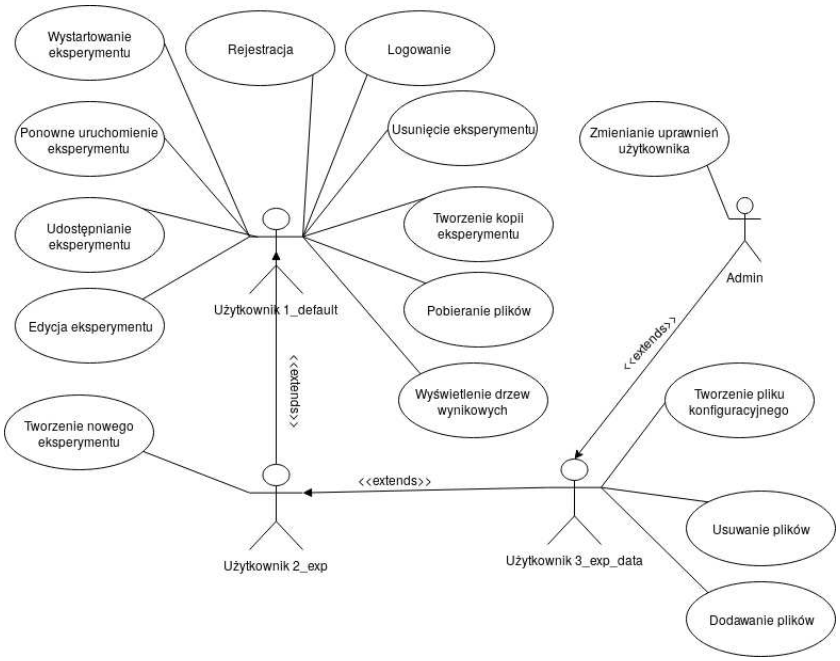
2.1 Wymagania funkcjonalne

Tworzenie aplikacji należało zacząć od nakreślenia zakresu funkcjonalności, które aplikacja będzie udostępniać użytkownikom. Podstawowym zadaniem, jakie powinna spełniać jest możliwość przeprowadzania eksperymentów przy pomocy systemu GDT. Kolejnym ważnym aspektem jest wariant zarządzania, wyświetlania, udostępniania oraz edycji poszczególnych eksperymentów. Każdy z użytkowników powinien widzieć poszczególne doświadczenia, które zostały ukończone, są w trakcie lub czekają w kolejce do obliczenia. Aplikacja powinna również przede wszystkim w przejrzysty sposób wyświetlać wyniki doświadczenia w postaci wygenerowanego drzewa decyzyjnego i poszczególnych statystyk. Podczas uruchomienia nowego zadania do obliczenia, użytkownikowi zostanie wyświetlony pasek postępu oraz oszacowana długość trwania całego zadania, przy czym w dowolnym momencie będzie mógł anulować polecenie. Wraz z możliwością tworzenia eksperymentu nie odłącznym elementem będzie funkcjonalność zarządzania plikami wejściowymi oraz wyjściowymi. Dla użytkowników początkujących zostanie przedstawiona opcja tworzenia podstawowych plików konfiguracyjnych, bez wgłębiania się w bardziej zaawansowane parametry eksperymentu. Dostęp do całej platformy wymaga założenia konta użytkownika. Natomiast możliwość rejestracji oraz logowania będzie ogólnodostępna.

System kont użytkowników powinien wyróżniać różne role, które definiowałyby 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 z bazy danych oraz edycja uprawnień do poszczególnych eksperymentów.

Biorąc pod uwagę perspektywę udostępniania przez użytkownika doświadczeń innemu użytkownikowi, ważnym aspektem będzie umożliwienie ograniczenia części akcji wykonywanych na eksperymencie. Aplikacja nie pozwoli na zablokowanie wyświetlania wraz ze statystykami. Natomiast reszta funkcjonalności możliwych do wykonania na doświadczeniu, 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.



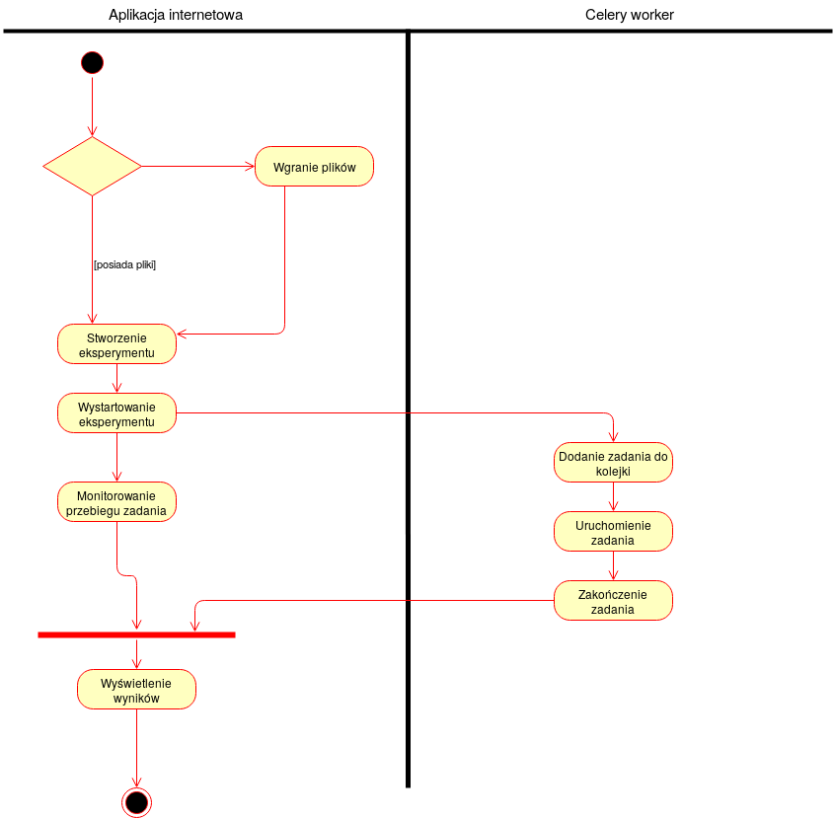
Rysunek 2.1: Diagram przypadków użycia, źródło: opracowanie własne

Na rysunku 2.1 przedstawiono wszystkie funkcjonalności w postaci diagramu przypadków użycia. W aplikacji zostały wyszczególnione trzy role dostępne do uzyskania dla każdego użytkownika oraz rola administratora całego 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.

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 prawd 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

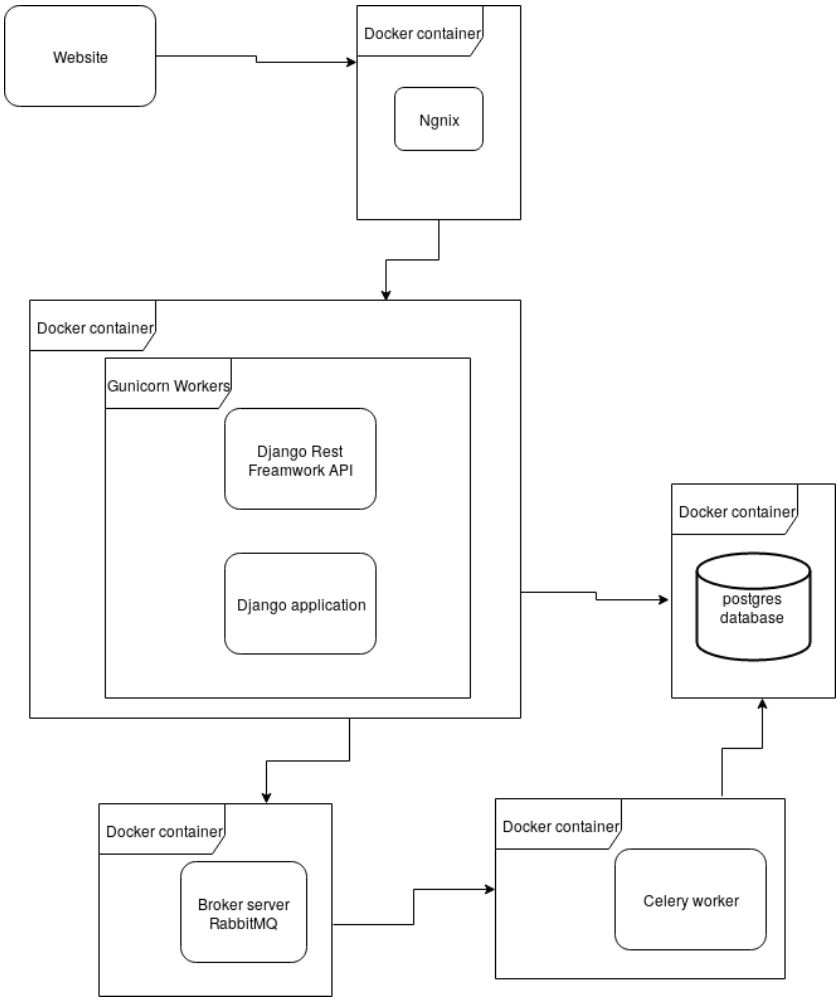
    - Częstotliwość wykonywania: Około 30 razy dziennie na każdego użytkownika,
    - Typowy czas realizacji: 10 sekund.

2.2 Wymagania niefunkcjonalne

Projekt aplikacji zostanie podzielony na dwa oddzielne komponenty jeden odpowiadający za stronę internetową, drugi natomiast za kolejkowanie i uruchamianie zadań w systemie GDT. Kompletne oprogramowanie pozwala na zarządzanie całością aplikacji w dość przejrzysty sposób. Poszczególne elementy strony serwerowej 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, jest możliwa w sposób natychmiastowy i mało inwazyjny. Jedyną rzeczą niezbędna na serwerze, aby uruchomić aplikacje jest platforma Docker.

2.3 Wykorzystane technologie

Aplikacja zostanie zaimplementowana z użyciem języka programowania Python, który zyskuję coraz większą popularność wśród programistów. Charakteryzuje się on łatwym progiem wejścia oraz wspieraniem kilku różnych paradygmatów programowania takich jak programowanie funkcyjne, proceduralne i obiektowe [14]. Po stronie serwera będzie



Rysunek 2.3: Architektura systemu, źródło: opracowanie własne

wystawione API w architekturze REST (*Representational State Transfer*) z wykorzystaniem Django oraz Django REST Framework. Natomiast interfejs graficzny strony będzie zaprojektowany przy pomocy JavaScriptu oraz biblioteki ReactJS.

Jedną z najważniejszych cech języka Python jest interpretowalność kodu źródłowego zamiast jego kompilacja [4]. Umożliwia on pisanie zarówno skryptów jak i zaawansowane programy. Kolejnym jego atrybutem 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 wiążę się z pewnymi wadami. W trakcie implementacji, programista musi sam pamiętać jaki typ w danym momencie ma zmienna. Z pomocą przychodzi 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 zostanie wykorzystany framework Django, który jest jednym z najbardziej popularnych rozwiązań webowych w języku Python. Charakteryzuje się on dużą szybkością implementacji oraz jasnym podziałem aplikacji na konkretne komponenty. Zarazem narzuca on pewną strukturę projektu, która zapewnia czystość kodu oraz możliwość ponownego używania wcześniej opracowanych modułów [3]. Społeczność zebrana wokół tej platformy, czynnie 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 edycje rekordów w bazie danych.

W celu wystawienia REST API(*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 [5]. Świadczy to nie tylko o popularności tego rozwiązania, ale też o jakości jego wykonania. Cały framework skupia się na wystawieniu dla programisty 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. Kolejność kroków pracy API możemy określić w następujący sposób:

1. Klient tworzy zapytanie i uzupełnia je o potrzebne dane,
2. Następnie następuje wysłanie zapytania 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. Jest to rozwiązanie pod licencją *Open Source* i szeroko

stosowane również z aplikacjami opartymi o technologie Django. Bazę można w łatwy sposób podpiąć pod panel administratora, ale także uzyskać dostęp z poziomu kodu aplikacji. W bazie danych będą zapisywane konta użytkowników oraz informacje o stworzonych eksperymentach wraz ze ścieżkami do plików.

Biblioteka ReactJS łącznie z JavaScriptem pozwoli na zaimplementowanie funkcjonalnego interfejsu graficznego strony internetowej. Zapewnia ona pewną strukturę projektu, która oferuje czystość kodu, czytelność oraz wygodę użytkowania. Pozwala na wstawianie wstawek kodu hmtl do kodu JavaScrioptowego za pośrednictwem języka JSX. Początkowo bibliotek została stworzona dla potrzeb wewnętrznych firmy Facebook, ale z czasem została udostępniona innym twórcom [10]. Programiści na całym świecie tak docenili te rozwiązanie, że ReactJS jest aktualnie wykorzystywany przez wielkie korporacje takie jak Netflix czy Uber, a sam projekt jest czwartym najpopularniejszym repozytorium na GitHub [6]. W celu połączenia interfejsu graficznego z logiką biznesową wystawioną za pomocą REST API została wykorzystana biblioteka Axios [1]. Cechuje się implementacją klienta http działająca po stronie strony internetowej. 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 [17]. 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 (*message broker*), który odpowiada za przesył informacji pomiędzy dwoma komponentami. Przykładem takiego oprogramowania może być broker RabbitMQ i on zostanie wykorzystany podczas implementacji aplikacji dyplomowej [16]. Oprogramowanie do wymiany wiadomości od Pivotal Software 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 [2]. W projekcie strony

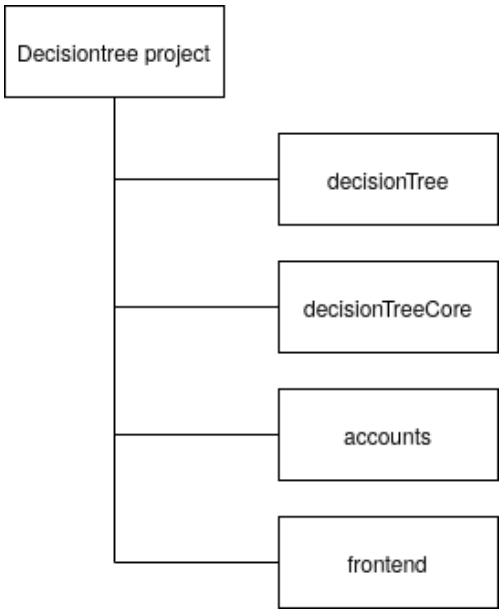
internetowej zostanie wykorzystany zaraz obok serwera www o nazwie Nginx pełniącego rolę pośrednika zapytań. Serwer www posłuży do przekierowania przychodzących pod adres domeny „decisiontree.pl" żądań prosto do aplikacji działającej pod serwerem Gunicorn.

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 łatwą wirtualizacje oraz podział projektu na oddzielne komponenty [9]. Stosując takie rozwiązanie w dowolnej chwili można podmieniać ze sobą 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.

3. Architektura rozwiązania

3.1 Architektura aplikacji

Struktura aplikacji jest oparta na podziale wynikającym z wykorzystania frameworka Django. Zgodnie z tym założeniem projekt dzieli się na poszczególne moduły, które w rozumieniu platformy nazywane są paczkami. W aplikacji można wyróżnić cztery podstawowe elementy kolejno odpowiadające za zarządzanie eksperymentami i użytkownikami, interfejs graficzny oraz aplikacja zbierająca wszystko w całość. Schemat modułów jest przedstawiony na Rys. 3.1.



Rysunek 3.1: Podział projektu na moduły, źródło: opracowanie własne

3.2 Przechowywanie danych

w celu przechowywania wszelkich informacji została wykorzystana relacyjna baza danych PostgresSQL. Platforma ta jest postawiona na oddzielnym kontenerze w celu uzyskania większej stabilności i nie zależności od głównego modułu aplikacji. Schemat struktury wszystkich tabel został przedstawiony na Rys. 3.2. Większość tabel wynika z samego zastosowania frameworka Django i DjangoRestFramework. Wykorzystując gotowe rozwiązana zostały zapewnione takie modele jak „auth_user” odpowiadający za zapisywanie

informacji o użytkownikach, czy też „authtoken_token” mająca na celu przetrzymywanie tokenów autoryzacji. Do całej struktury zostały dodane dodatkowe tabele:

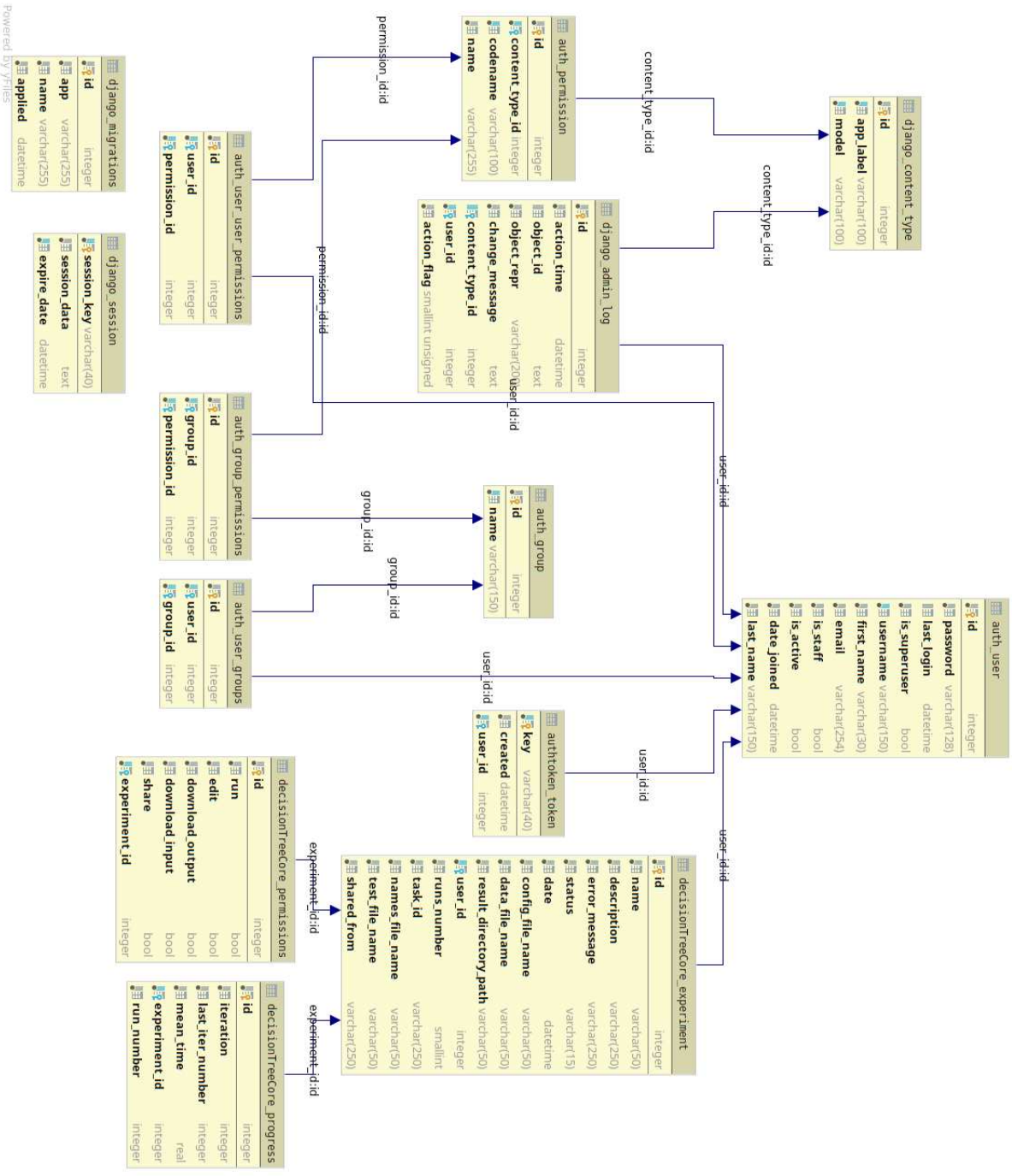
- „decisionTreeCore_experiment” przetrzymująca dane o eksperymentach,
- „decisionTreeCore_permissions” zawierająca informacje o prawach dostępowych do eksperymentu,
- „decisionTreeCore_progress” składa się z pól określających postęp wykonywania doświadczenia.

Relacja pomiędzy nowo stworzonymi tabelami, a użytkownikiem została przedstawiona na Rys. 3.3.

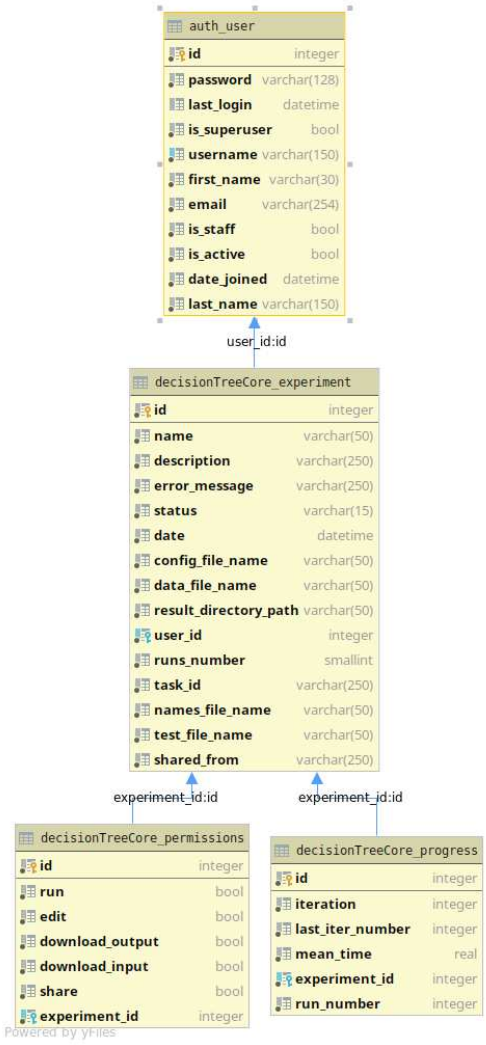
3.3 Wdrożenie aplikacji

Nazwa pola	Typ	Opis
id	integer	Klucz główny tabeli
name	varchar(50)	Nazwa eksperymentu
description	varchar(250)	Opis eksperymentu
error_message	varchar(250)	Wiadomość o błędzie, który wystąpił podczas uruchomienia eksperyment
status	varchar(15)	Pole określające w jakim statusie znajduje się eksperyment. Możliwe wartości to: „Created”, „In queue”, „Running”, „Finished”, „Error”. Zmiana statusów następuje w trakcie przechodzenia eksperymentu przez kolejne etapy
data	datetime	Data stworzenia eksperymentu przez użytkownika
config_file_name	varchar(50)	Nazwa pliku konfiguracyjnego użytego w eksperymencie
data_file_name	varchar(50)	Nazwa pliku zawierającego zbiór uczący
test_file_name	varchar(50)	Nazwa pliku zawierającego zbiór testowy
names_file_name	varchar(50)	Nazwa pliku określającego nazwy klas oraz rodzaj zmiennych
result_directory_path	varchar(50)	Ścieżka do folderu z wynikami eksperymentu
user_id	integer	ID użytkownika, który jest właścicielem eksperymentu
runs_number	smalli	Pole określające ile przebiegów algorytmu ma się odbyć podczas uruchomienia eksperymentu w systemie GDT. Pełni ważną rolę przy tworzeniu paska postępu
task_id	varchar(250)	Zawiera id zadania, które trafiło do robotnika Celery. Umożliwia zarządzanie danym zadaniem np. usunięcie z kolejki, lub anulowanie w trakcie trwania
shared_from	varchar(250)	Pole pełni rolę zapisu informacji o poprzednich właścicielach. W wartością pola są nazwy użytkowników. Podczas kolejnych udostępnień do pola są dodawane po przecinku następne loginy

Tabela 3.1: Opis pól encji „decisionTreeCore_experiment”



Rysunek 3.2: Schemat bazy danych, źródło: opracowanie własne



Rysunek 3.3: Schemat tabel dodatkowych, źródło: opracowanie własne

4. Przedstawienie aplikacji oraz testy

Podsumowanie

Tutaj będzie podsumowanie.

Bibliografia

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

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

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

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

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

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

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

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

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

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

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

[12] B. Inny. Tytu³ publikacji. In *Tytuł książki*, pages 5–32, Feb 2011.

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

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

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

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

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

Spis tabel

Spis rysunków

Spis listingów

Spis algorytmów