

POLITECHNIKA BIAŁOSTOCKA

WYDZIAŁ INFORMATYKI

PRACA DYPLOMOWA INŻYNIERSKA

TEMAT: System rezerwacji terminów spotkań

WYKONAWCA: Piotr Szumowski

OPIEKUN PRACY DYPLOMOWEJ : dr hab. inż. Ireneusz Mrozek, prof. PB

BIAŁYSTOK 2024 ROK

Subject of diploma thesis

Appointment booking system

SUMMARY

The purpose of the thesis was to create a web application that would be an appointment booking system. Application should allow organizing and to plan a meeting between two peoples. In the context of the company, especially between service providers and clients. The backend of the project was created in Python language and Django framework, while the application part responsible for the client side and interface was built in JavaScript language with use of libraries as jQuery and FullCalendar. Used framework for styling was Tailwind CSS. There was used in project extension for jQuery DataTables that allow to make more interactive tables. The project was created in Pycharm IDE with help of git system of control version. The first chapter serves as an introduction to the entire work. The following second chapter includes specification, functional requirements together with user cases and non-functional requirements. The third one contains descriptions of major solutions existing on the market. Afterward, their fourth chapter is about used technology in project and possible alternatives. In the fifth chapter there is a specification of used architecture, project structure and database. The six chapter designation is to show the most important implementation problems and solutions used in the project. The seventh chapter is a presentation of the completed application. The last chapter contains a summary of the entire work.

Spis treści

1.	Wstęp.....	1
2.	Analiza wymagań systemu.....	3
2.1	Specyfikacja problemu.....	3
2.2	Wymagania funkcjonalne	4
2.2.1	Użytkownik niezalogowany	4
2.2.2	Użytkownik zalogowany - klient	7
2.2.3	Użytkownik zalogowany – usługodawca	10
2.2.4	Użytkownik zalogowany – admin.....	12
2.3	Wymagania нефункционалне	14
2.3.1	Czas odpowiedzi	14
2.3.2	Kompatybilność z przeglądarkami internetowymi	15
2.3.3	Prosty oraz intuicyjny w obsłudze interfejs	15
3.	Porównanie konkurencyjnych rozwiązań na rynku.....	16
3.1	Kalendarz Google	16
3.2	Microsoft Outlook Calendar	17
3.3	Doodle.....	18
4.	Zastosowane technologie wraz z dostępnymi alternatywami	20
4.1	Technologie po stronie serwera	20
4.1.1	Python.....	20
4.1.2	Django	21
4.1.3	SQLite3	22
4.2	Technologie po stronie klienta.....	22
4.2.1	JavaScript	22
4.2.2	jQuery.....	23
4.2.3	Tailwind CSS	23
4.2.4	FullCalendar	24
4.2.5	DataTables.....	24
4.3	Pozostałe narzędzia	25
4.3.1	Pycharm.....	25

4.3.2	Git.....	25
4.4	Konkurencyjne technologie	26
4.4.1	Java.....	26
4.4.2	C#	26
4.4.3	Bootstrap	27
5.	Projekt aplikacji.....	28
5.1	Architektura	28
5.2	Struktura projektu	28
5.3	Baza danych	30
6.	Implementacyjne rozwiązanie projektu.....	34
6.1	Wybrane szczegóły implementacyjne.....	34
6.1.1	Komunikacja frontendu z backendem.....	34
6.1.2	Wysyłanie maili w dodatkowym wątku	36
6.1.3	Autoryzacja użytkownika podczas logowania	37
6.2	Testy.....	40
6.2.1	Testowanie manualne	40
6.2.2	Testy jednostkowe.....	41
6.3	Wdrożenie aplikacji na serwer	41
7.	Prezentacja aplikacji.....	43
7.1	Strona startowa z kalendarzem	43
7.2	Rejestracja i logowanie	44
7.3	Profil Użytkownika	45
7.4	Tabele wydarzeń	46
7.5	Dodawanie dostępnego terminu.....	47
7.6	Rezerwowanie i proponowanie terminu	48
7.7	Potwierdzanie lub odrzucanie rezerwacji	49
7.8	Edytowanie i usuwanie wydarzeń.....	49
8.	Podsumowanie.....	51
	Bibliografia	52
	Spis kodów źródłowych.....	52

1. Wstęp

Człowiek jako istota społeczna, nieustannie nawiązuje kontakty i spotyka się z niezliczoną liczbą obcych ludzi w trakcie swojego życia. Znacząca część tych spotkań wymaga wcześniejszego zaplanowania w celu ustalenia dogodnego terminu dla obu stron. W przypadku firm i organizacji, które cechują się dużą aktywnością oraz prowadzą ogromną ilość spotkań z klientami, manualne wprowadzanie do harmonogramu wszystkich tych wydarzeń może być zbyt czasochłonne bez efektywnej metody. Tradycyjne rozwiązania tego problemu, takie jak korzystanie z usług sekretarek czy sekretarzy, którzy pełnią rolę pośrednika w ustalaniu terminów, są z pewnością sprawdzone, lecz często nieskuteczne i niewydajne. W kontekście finansowym, zwłaszcza dla mniejszych firm czy organizacji, koszty zatrudnienia dodatkowych osób mogą być bardzo obciążające i przerastać dostępne możliwości finansowe. Ponadto, ograniczenia czasowe, związane z obsługą jedynie ograniczonej liczby osób jednocześnie oraz konieczność dostosowania się do harmonogramu pracy takiego personelu są z pewnością nie pożądanym rezultatem. W tym kontekście, powstaje potrzeba opracowania innych skuteczniejszych narzędzi, które nie tylko usprawnią procedurę rezerwacji terminów, ale także będą dostępne bez względu na czas i miejsce, eliminując ograniczenia tradycyjnych podejść.

Nowoczesne aplikacje internetowe jawią się właśnie jako jedno z potencjalnych rozwiązań tego problemu, które oferuje innowacyjny sposób na organizację spotkań w bardziej elastyczny i przystępny sposób dla obu stron. W przeciwieństwie do tradycyjnych metod, aplikacje te eliminują ograniczenia czasowe i finansowe, oferując dostępność przez całą dobę przy jednoczesnym niskim koszcie utrzymania. Ponadto, interaktywność tych platform pozwala wielu użytkownikom równocześnie korzystać z systemu, co sprawia, że są one nie tylko efektywne, ale także skalowalne dla różnych przedsiębiorstw. Na rynku wciąż przybywa coraz więcej różnego oprogramowania i aplikacji starających się możliwie rozwiązać ten problem. Duża ilość aplikacji tego typu wymusza konkurencję, co zapewnia wyższą jakość oraz oferuje użytkownikom coraz to bardziej specyficzne funkcje użycia w celach zapełnienia wszystkich potrzeb na rynku. Jednakże stworzenie w pełni uniwersalnej i kompleksowej aplikacji jest niezwykle trudne, ze względu na bardzo zróżnicowane potrzeby użytkowników.

Celem niniejszej pracy jest stworzenie zaawansowanej aplikacji internetowej pełniącej rolę systemu rezerwacji terminów, który umożliwi zaplanowanie terminów spotkań bez

konieczności osoby pośredniczącej. Aplikacja ta jest kierowana głównie w stronę usługodawców poszukających prostego i efektywnego sposobu na tworzenie oraz zarządzanie dostępnymi terminami, proponowanymi terminami spotkań i rezerwacjami. System powinien być jak najbardziej uniwersalny przy jednoczesnym zachowaniu prostoty w obsłudze i przyjaznym interfejsie dla użytkownika. Usługodawcy mają możliwość dodawania dostępnych terminów oraz akceptowania lub odrzucania rezerwacji klientów, co gwarantuje im elastyczność i możliwość dostosowania do własnych potrzeb. Z kolei klienci mogą skorzystać z opcji rezerwacji lub proponowania nowego terminu spotkania, co zapewnia im swobodę wyboru. W celu usprawnienia korzystania z systemu zaimplementowano kilka kluczowych funkcjonalności takich jak złożone filtrowanie, wyszukiwanie wydarzeń oraz powiadamianie użytkowników o dotyczących ich zmianach zarówno w aplikacji jak i mailowo.

Rozdział pierwszy tej pracy został przeznaczony na przedstawienie zagadnienia, którego dotyczy projekt. W drugim rozdziale przedstawiono analizę wymagań zarówno funkcjonalnych jak i нефункциональных oraz specyfikację problemu. W trzecim rozdziale jest przegląd aktualnie istniejących popularnych rozwiązań na rynku wraz z porównaniem ich zalet oraz wad. Czwarty rozdział został poświęcony na opisanie technologii zastosowanych w projekcie oraz alternatywnych do nich rozwiązań. Kolejny piąty rozdział został przeznaczony na opisanie architektury, struktury projektu oraz bazy danych. Szósty rozdział opisuje implementacyjne rozwiązania zawarte w projekcie, testy oraz sposób wdrożenia aplikacji na platformę hostingową. Siódmy rozdział to prezentacja aplikacji, która może służyć też jako skrócona instrukcja użytkowania, a ostatni rozdział to podsumowanie całej pracy.

2. Analiza wymagań systemu

W niniejszym rozdziale skoncentrowano się na przedstawieniu wymagań jakie stawia użytkownik względem aplikacji. Omówiono w nim wymogi funkcjonalne i нефункционалне, które są kluczowe dla zaimplementowanego systemu. Celem tego podejścia jest zapewnienie dostarczenia użytkownikom potrzebnych funkcjonalności i zapewnienia pozytywnego wrażenia użytkownika (ang. UX - User Experience). W procesie analizy, użytkownicy zostali podzieleni na konkretne grupy, co umożliwiło precyzyjne przypisanie im odpowiednich ról. Ta segmentacja użytkowników pozwoliła na dostosowanie interfejsu do indywidualnych potrzeb poszczególnych grup, zapewniając im dostęp do odpowiednich funkcji i treści w ramach serwisu. Takie podejście przyczynia się do bardziej spersonalizowanego i efektywnego korzystania z aplikacji, dostarczając użytkownikom dokładnie tego, czego potrzebują w zależności od ich roli czy oczekiwań oraz skutkuje wzrostem bezpieczeństwa, wydajności i stabilności systemu.

2.1 Specyfikacja problemu

Aplikacje internetowe pełniące funkcję systemów rezerwacji terminów mają za zadanie przede wszystkim umożliwienie dwóm stronom skutecznego zaplanowanie pasującego terminu spotkania. Najważniejszą zatem funkcjonalnością tworzonego serwisu powinna być zatem obsługa tworzonych dostępnych terminów spotkań, propozycji terminów oraz rezerwacji. Istotne jest uwzględnienie różnych scenariuszy gdzie zarówno usługodawca jak i klient mogą inicjować proces ustalania termin spotkania, oraz obsłużenia wszelkich przypadków następujących po tym etapie. Ważnym aspektem jest również możliwość zmiany wcześniej zainicjowanych ofert. Edycja i usuwanie wcześniej złożonych propozycji, umożliwia dostosowanie się do ewentualnych zmian w planach. Jednakże tworzy również zamieszanie i może być mylące dla drugiej strony, z tego powodu potrzeba też środka pozwalającego na ostateczne potwierdzenie spotkania oraz komunikacji z drugą stroną w celu poinformowania o bieżących zmianach.

Wymagany będzie również środek pozwalający na komfortowe i intuicyjne wyszukanie lub wyfiltrowanie pasującego terminu wśród ogromnej ilości potencjalnych wydarzeń. Wszystkie funkcjonalności powinny zostać zaimplementowane w bezpiecznej i responsywnej aplikacji internetowej z przejrzystym i intuicyjnym interfejsem użytkownika, który umożliwi łatwe nawigowanie i obsługę systemu. Interfejs powinien być

prosty oraz czytelny nawet dla osób bez doświadczenia w korzystaniu z tego typu serwisów oraz nie posiadających wyrafinowanych umiejętności planowania. W celu zwiększenia dostępności dla osób nieznających języków obcych, cały interfejs powinien być dostępny w języku polskim, co dodatkowo ułatwi obsługę systemu dla szerokiego grona użytkowników.

2.2 Wymagania funkcjonalne

Jednym z ważniejszych etapów podczas projektowania wybranej aplikacji jest precyzyjne określenie funkcjonalności jakie będą dostępne dla użytkownika końcowego, a także określenia wszelkich zależności występujących pomiędzy nimi. Ten krok pozwala na dokładne zdefiniowanie odpowiednich grup użytkowników i przedzielenia im dostępu do wybranych części systemu. Następne podrozdziały skupiają się na szczegółowym określeniu przypadków użycia aplikacji dla poszczególnych grup użytkowników w celu kompleksowego zrozumienia ich indywidualnych potrzeb i wymagań.

2.2.1 Użytkownik niezalogowany

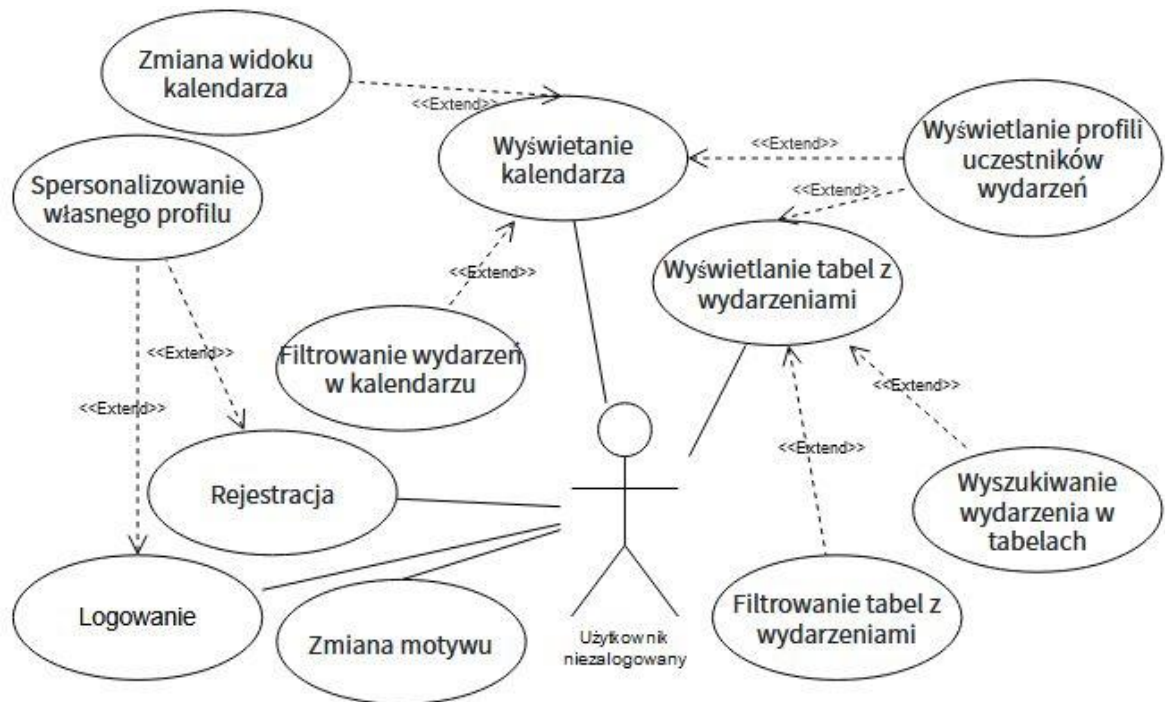
Użytkownik niezalogowany na stronie jest traktowany jako gość. Aby zagwarantować otwartość i przejrzystość ofert w systemie dla potencjalnego użytkownika, część funkcjonalności serwisu nie wymaga żadnej formy uprzedniej autentykacji.

Kluczowe funkcjonalności dostępne dla niezalogowanego użytkownika:

- Przeglądanie kalendarza z wszystkimi wydarzeniami
- Przeglądanie tabel wydarzeń
- Wyświetlanie profili użytkowników biorących udział w wydarzeniach
- Filtrowanie wydarzeń w kalendarzu po usługodawcy i usługobiorcy
- Wyszukiwanie wydarzeń w tabeli po dowolnym parametrze wydarzenia
- Filtrowanie wydarzeń w tabeli po zakresie czasowym

Każdy użytkownik ma też możliwość zmiany motywu na ciemny/jasny w celu poprawienia doznań wizualnych oraz zmiany widoku kalendarza na bardziej mu odpowiadający. Dostępne widoki kalendarza to widok całego miesiąca, widok konkretnego tygodnia oraz widok pojedynczego dnia. Oczywiście w każdej chwili użytkownik może się również zarejestrować i/lub zalogować w celu zdobycia większej ilości uprawnień i możliwości interakcji z pozostałymi użytkownikami w systemie.

Na poniższym diagramie przypadków użycia (Rys. 2.1) przedstawiono funkcjonalności dostępne dla niezalogowanego użytkownika aplikacji.



Rys. 2.1 Diagram przypadków użycia niezalogowanego użytkownika (opracowanie własne)
Opis przypadków użycia – Rejestracja/logowanie i spersonalizowania własnego profilu

- Uczestniczący aktorzy:
 - o Użytkownik niezalogowany
- Podstawowy ciąg zdarzeń:
 - o Użytkownik uruchamia aplikację po raz pierwszy
 - o Niezalogowany użytkownik klika w przycisk "Logowanie" lub "Rejestracja" znajdujące się na górze strony po prawej stronie paska nawigacji
 - o Jeśli użytkownik nie ma wcześniej utworzonego konta może przejść z panelu logowania do panelu rejestracji za pomocą przycisku rejestracji na pasku nawigacji lub linku pod przyciskiem logowania w formularzu
 - o W formularzu rejestracji użytkownik podaje nazwę użytkownika, imię, nazwisko, adres email oraz dwukrotnie hasło.
 - o Po wypełnieniu formularza i zatwierdzeniu przyciskiem "Zarejestruj się", a także po udanej walidacji danych, użytkownik zostaje automatycznie zalogowany i przekierowany do głównej strony wraz z powiadomieniem o pomyślnie udanej rejestracji.

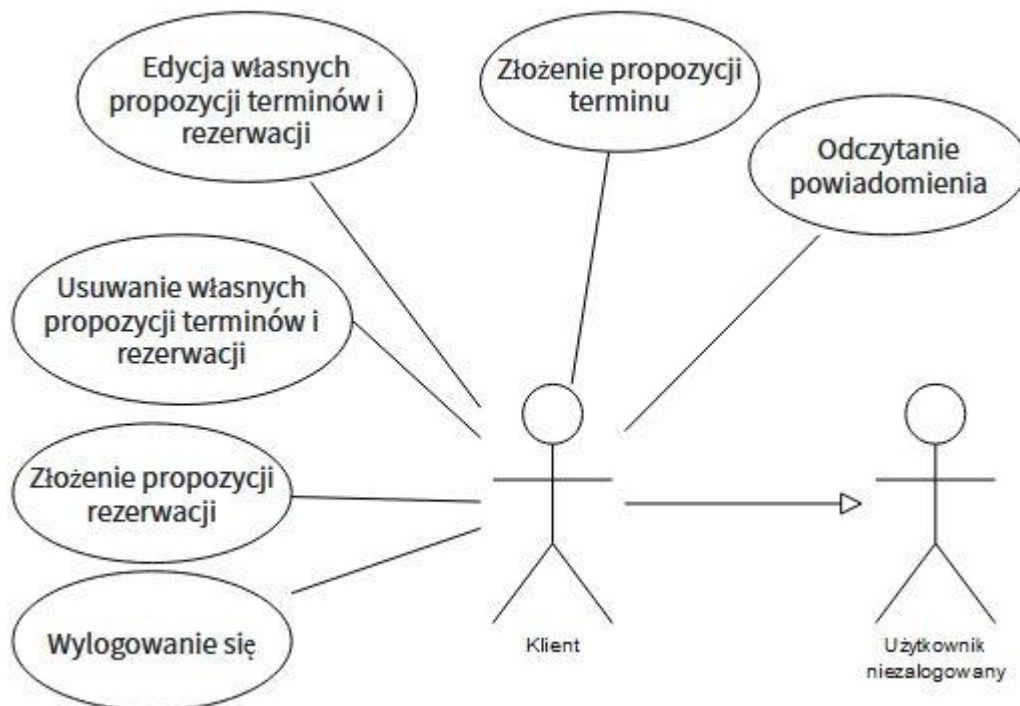
- o Od tej chwili użytkownik będzie zalogowany z wszystkimi uprawnieniami klienta.
- o Następnie użytkownik ma możliwość spersonalizowania swojego profilu poprzez kliknięcie ikonki z domyślnym obrazkiem użytkownika na górze strony po prawej stronie paska nawigacji i wybrania opcji "edytuj profil".
- o W trakcie edycji profilu użytkownik może uzupełnić bardziej szczegółowe dane takie jak wiek, płeć, zdjęcie profilowe czy opis użytkownika, a także zmienić wcześniej podane dane, takie jak nazwa użytkownika, imię, nazwisko czy adres email.
- o Zmiany w profilu może zatwierdzić przyciskiem "aktualizuj dane" lub cofnąć zmiany za pomocą przycisku "resetuj dane".
- Alternatywne ciągi zdarzeń:
 - o Użytkownik podaje nieprawidłowe dane
 - ◆ System waliduje podane dane i zwraca stosowny komunikat z błędem oraz informacją o wymaganych ograniczeniach w systemie, nie przysyłając żądania rejestracji/zmiany danych na serwer.
 - o Użytkownik próbuje użyć danych zajętych przez innego użytkownika
 - ◆ System informuje użytkownika o wykorzystywaniu już podanych danych przez innego użytkownika.
- Zależności czasowe
 - o Częstotliwość wykonania: W zależności od popularności aplikacji od kilku do kilkudziesięciu razy dziennie.
 - o Przewidywane spiętrzenia: Największa ilość rejestracji wystąpi podczas początkowego wdrażania klientów danej firmy do systemu.
 - o Typowy czas realizacji: Około jednej minuty.
 - o Maksymalny czas realizacji: Praktycznie maksymalnie 3 minuty.
- Wartości uzyskane przez aktorów po zakończeniu przypadku użycia
 - o Aktywne konto użytkownika w systemie.
 - o Klient posiada dostęp do większej części funkcjonalności serwisu.
 - o Użytkownik ma spersonalizowany profil wraz z osobistymi informacjami.

2.2.2 Użytkownik zalogowany - klient

Po pomyślnym zakończeniu rejestracji, użytkownik automatycznie otrzymuje status klienta, co umożliwia mu na korzystanie z dodatkowych funkcji związanych głównie z tworzeniem i zarządzaniem własnymi terminami spotkań. Z wyjątkiem funkcjonalności związanej ściśle powiązanej z logowaniem i rejestracją, klient posiada wszystkie funkcjonalności gościa, a ponadto zyskuje również kilka dodatkowych funkcjonalności:

- Składania własnych propozycji terminów spotkań
- Otrzymywania powiadomień o aktualnościach dotyczących wydarzeń w których bierze klient udział. Po kliknięciu powiadomienia użytkownik zostanie przekierowany do lokalizacji wydarzenia w kalendarzu, którego dotyczyło powiadomienie. A samo wydarzenie będzie w kalendarzu specjalnie oznaczone w celu łatwego odróżnienia go od pozostałych wydarzeń.
- Zgłaszania chęci zarezerwowania dostępnego terminu spotkania
- Edytowania własnych zarówno propozycji terminów jak i zgłoszeń rezerwacji
- Usuwania własnych propozycji terminów i zgłoszeń rezerwacji

Na poniższym diagramie (Rys. 2.2) przedstawiono przypadki użycia dla klienta.



Rys. 2.2 Diagram przypadków użycia klienta (Opracowanie własne)

Opis przypadków użycia – Złożenia propozycji terminu i złożenia propozycji rezerwacji

- Uczestniczący aktorzy
 - o Użytkownik zalogowany – klient
- Podstawowy ciąg zdarzeń
 - o Klient może dodać propozycję terminu w kalendarzu poprzez wybranie odpowiedniego zakresu dat lub pojedynczej daty na kalendarzu lub kliknięciu przycisku po prawej stronie powyżej kalendarza. Jeśli użytkownik skorzystał z kalendarza to wystarczy, że poda:

- ◆ Godzinę początku spotkania,
- ◆ Godzinę końcową wydarzenia.

W przypadku wykorzystania przycisku, dodatkowo należy podać:

- ◆ Datę początku wydarzenia,
- ◆ Datę zakończenia wydarzenia.

Aktor zatwierdza wprowadzone dane za pomocą przycisku „Dodaj propozycję”. Jeśli operacja zakończy się powodzeniem, żądanie zostanie wysłane asynchronicznie do serwera, a nowe wydarzenie pojawia się w kalendarzu użytkownika po krótkim czasie, bez potrzeby odświeżania strony.

- o Klient ma także możliwość zgłoszenia oferty rezerwacji dostępnego terminu usługodawcy. Dostępne terminy są oznaczone w kalendarzu kolorem zielonym. Po kliknięciu na dostępny termin użytkownik zobaczy jedno z dwóch okienek. Jeśli dostępny termin nie miał ustawionych podziałów czasowych użytkownik zobaczy okienko w którym będzie mógł wybrać czy chce zarezerwować cały dostępny termin, czy tylko jego fragment. Wybranie pierwszej opcji kończy operację natomiast w przypadku drugiej opcji użytkownik musi jeszcze podać przedział czasowy w którym chce dokonać rezerwacji i zatwierdzić ją przyciskiem „rezerwuj”. Jeśli natomiast dostępny termin miał ustawiony podział czasowy wtedy użytkownik ma tylko jedną opcję z listą godzin. Użytkownik wybiera wtedy pasującą godzinę z listy i zatwierdza rezerwację przyciskiem poniżej z napisem „rezerwuj”.
- Alternatywne ciągi zdarzeń
 - o Użytkownik zostanie wylogowany w trakcie. W takim przypadku nie zostanie dodana zaproponowana zmiana, a wyświetli się błąd z serwera.

- o Użytkownik ma już zaplanowane spotkanie, które nachodzi na dany zakres czasowy.

- ◆ Jeśli klient posiada zatwierdzony termin spotkania w tym czasie, to nie ma możliwości dodawania nowych terminów, ponieważ mogłoby to spowodować kolizję terminów.

- ◆ W przypadku, gdy użytkownik już wcześniej złożył propozycję terminu spotkania dla tego samego usługodawcy w tym samym terminie, również nie ma możliwości dodania nowego zgłoszenia w nachodzącym na siebie czasie.

W powyższych sytuacjach zostanie wyświetlony komunikat o niemożliwości stworzenia nowego terminu z powodu występującej kolizji, a samo żądanie zostanie odrzucone przez serwer.

- o Zakresy czasowe podane przez klienta nie będą możliwe do zrealizowania:

- ◆ Jeśli użytkownik poda czas początkowy wydarzenia będący po czasie początkowym.

- ◆ Jeśli podany czas wydarzenia wykracza poza dostępny czas terminu.

W takich przypadkach zostanie zwrócony komunikat błędu informujący użytkownika o niemożliwości przetworzenia podanego żądania.

- Zależności czasowe

- o Częstotliwość wykonania: Bardzo częste. W przypadku bardziej popularnych aplikacji możliwe nawet kilkaset razy na godzinę.

- o Przewidywane spiętrzenie: Największe użycie aplikacji prawdopodobnie będzie w godzinach popołudniowych, kiedy duża część ludzi pracuje.

- o Typowy czas realizacji: 20-30 sekund.

- o Maksymalny czas realizacji: 1-2 minuty.

- Wartości uzyskane przez aktorów po zakończeniu przypadku użycia

- o Klient będzie miał szansę na skuteczne umówienie spotkania jeśli usługodawca zatwierdzi podaną propozycję terminu/rezerwacji, co stanowi kluczową kwestię dla znaczenia istnienia całej aplikacji.

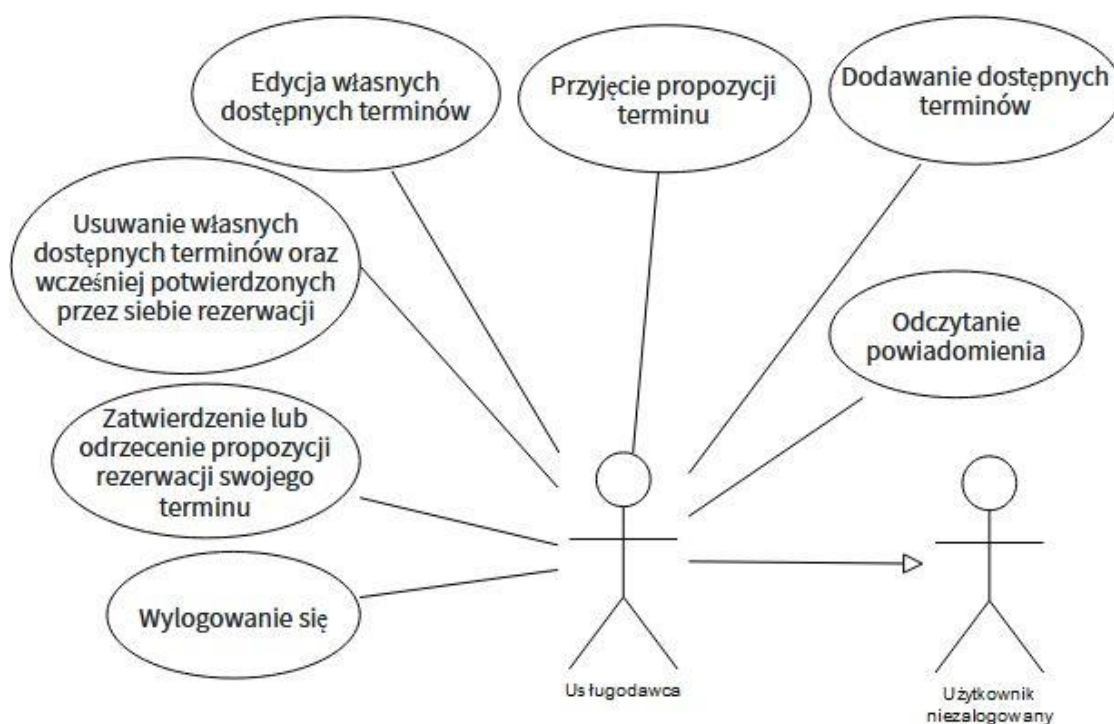
2.2.3 Użytkownik zalogowany – usługodawca

Admin systemu ma możliwość przydzielenia roli „usługodawcy” dla danego użytkownika. Usługodawca podobnie jak klient rozszerza funkcjonalności osoby niezalogowanej z wyjątkiem funkcjonalności dotyczących autoryzacji, jednakże w przeciwieństwie do klienta, usługodawca posiada funkcje pozwalające nie tylko ustawiać dostępne terminy kiedy jest możliwe zrealizowanie spotkania, co pozwala klientom na zgłaszanie propozycji rezerwacji, ale również może zatwierdzać propozycje terminów.

Pełna lista funkcjonalności stworzonych specjalnie dla usługodawcy:

- Dodawanie do kalendarza dostępnych terminów spotkań
- Przyjęcie propozycji terminu zaproponowanego przez klienta
- Zatwierdzenie lub odrzucenie propozycji rezerwacji swojego terminu przez klienta
- Edycja własnych dostępnych terminów
- Usuwanie własnych dostępnych terminów oraz usuwanie potwierdzonych przez siebie rezerwacji
- Odczytanie powiadomień dotyczących wydarzeń w których bierze udział.

Na poniższym diagramie przypadków użycia (Rys. 2.3) pokazano wszystkie funkcjonalności specjalnie przeznaczone usługodawcy.



Rys. 2.3 Diagram przypadków użycia usługodawcy (opracowanie własne)

Opis przypadków użycia – Przyjęcie propozycji terminu klienta i zatwierdzenie lub odrzucenie rezerwacji swojego terminu

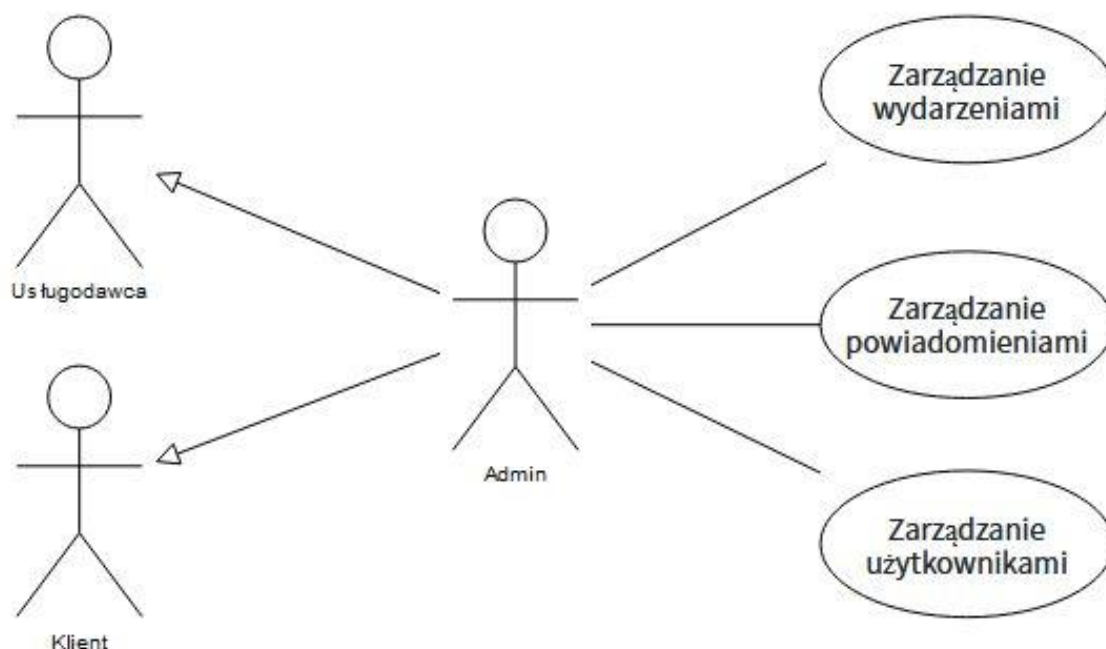
- Uczestniczący aktorzy
 - o Użytkownik zalogowany – Usługodawca lub admin
- Podstawowy ciąg zdarzeń
 - o Usługodawca wybiera wydarzenie w kalendarzu, którego chce zmienić status, poprzez kliknięcie na nie.
 - o Po wybraniu wydarzenia pojawia się okienko w którym użytkownik ma dwa przyciski do wyboru. Niebieski przycisk zatwierdzający wydarzenie oraz czerwony przycisk odrzucający rezerwację lub anulujący operację przyjęcia propozycji. Operacja kończy się po kliknięciu któregoś z przycisków.
- Alternatywne ciągi zdarzeń
 - o Usługodawca spróbuje wybrać wydarzenie należące do innego usługodawcy
 - ◆ System nie pozwala na wybranie wydarzenia należącego do innego usługodawcy, nie pojawia się okienko z możliwością zmiany statusu.
 - o Aktor podejmuje próbę zaakceptowania spotkania w terminie, w którym już aktualnie posiada zatwierdzone spotkanie.
 - ◆ Pojawia się komunikat o kolizji terminów, a samo żądanie utworzenia spotkania nie zostaje zrealizowane.
 - o Użytkownik zostanie wylogowany w trakcie.
 - ◆ Nie zostanie dodana zaproponowana zmiana i wyświetla się błąd z serwera.
- Zależności czasowe
 - o Częstotliwość wykonania: W zależności od popularności aplikacji istnieje możliwość wykonania nawet kilkaset razy w ciągu godziny.
 - o Przewidywane spiętrzenie: W godzinach pracy usługodawców.
 - o Typowy czas realizacji: 3 sekundy
 - o Maksymalny czas realizacji: 10 sekund
- Wartości uzyskane przez aktorów po zakończeniu przypadku użycia
 - o Odrzucony lub zatwierdzony nowy termin spotkania.
 - o Usunięcie nachodzących się czasowo niezatwierdzonych terminów klienta w przypadku potwierdzenia terminu spotkania.

2.2.4 Użytkownik zalogowany – admin

W każdym systemie istnieje rola administratora, która odgrywa kluczową rolę dla poprawnego działania serwisu. Podczas wdrożenia aplikacji, możliwe jest stworzenie użytkownika, który będzie posiadał status admina, tym samym nadając mu pełny dostęp do wszystkich funkcji systemu. Administrator posiada szczególne uprawnienia, które dają mu całkowity dostęp do wszystkich wydarzeń, użytkowników, ich informacji profilowych oraz powiadomień. To właśnie administrator jest odpowiedzialny za zarządzanie użytkownikami i przydziela innym użytkownikom rolę na przykład usługobiorcy. Administrator ma też możliwość tworzenia nowych administratorów. Wyznaczenie administratora powinno być starannie przemyślane i ograniczone tylko do zaufanych osób. Niewłaściwe korzystanie z uprawnień administratora może prowadzić do uszkodzenia systemu i utraty danych. Admin posiada szereg funkcji i możliwości, a przede wszystkim do jego głównych zadań należy:

- Zarządzanie wszystkimi użytkownikami. Administrator ma możliwość modyfikacji ról oraz danych wszystkich użytkowników, w tym może również zmieniać dane profilowe użytkowników takie jak np. zdjęcie profilowe czy opis.
- Zarządzanie wszystkimi wydarzeniami. Administrator posiada kompletny dostęp do wszystkich wydarzeń w systemie. Jest w stanie dodawać, edytować, usuwać, zmieniać status wydarzeń, a także przydzielać je innym użytkownikom.
- Zarządzanie wszystkimi powiadomieniami użytkowników. Administrator ma kontrolę nad powiadomieniami skierowanymi do użytkowników systemu. Może dodawać, edytować i usuwać powiadomienia oraz zmieniać ich status, określając czy zostały przeczytane czy też nie. Ta funkcjonalność umożliwia adminowi efektywne zarządzanie komunikacją w systemie, dostosowując ją do bieżących potrzeb użytkowników.

Na poniższym diagramie przypadków użycia (Rys. 2.4) pokazano funkcjonalności specjalnie dedykowane dla admina.



Rys. 2.4 Diagram przypadków użycia admina (opracowanie własne)

Opis przypadku użycia – Zarządzanie użytkownikami

- Uczestniczący aktorzy
 - o Użytkownik zalogowany - admin
- Podstawowy ciąg zdarzeń
 - o Administrator klika swoje zdjęcie profilowe znajdujące się na górze po prawej stronie nagłówka, wybiera opcję „Strona administracyjna” z rozwiniętej listy.
 - o W panelu administracyjnym wybiera tabele „Users” lub „User profiles”
 - o Następnie w wybranej tabeli szuka rekordu z użytkownikiem, którego chce zmienić. Może w tym celu skorzystać z narzędzi filtrowania po prawej stronie lub wyszukiwania powyżej tabeli, a także sortowania kolumn klikając na nagłówki kolumn.
 - o Po znalezieniu rekordu, który chce zmienić, klika na nazwę użytkownika z hiperłączem, a po wyświetleniu danych danego rekordu, dokonuje zmian i zatwierdza je poniższym niebieskim przyciskiem lub usuwa czerwonym przyciskiem po prawej.
 - o Istnieje także możliwość usunięcia większej ilości rekordów naraz poprzez zaznaczenie pól wyboru po lewej stronie w liście rekordów, wybraniu akcji usuwania powyżej tabelki i zatwierdzeniu tej akcji.
- Alternatywne ciągi zdarzeń

- o Osoba nieuprawniona próbuje uzyskać dostęp do panelu administracyjnego poprzez na przykład wklejenie adresu w pole przeglądarki:
 - ◆ Następuje wtedy przekierowanie do panelu logowania
- o Wprowadzone dane przez administratora nie przeszły walidacji serwera.
 - ◆ Jeżeli admin spróbuje dodać dane niewłaściwego typu lub naruszające strukturę systemu powodując np. kolizję zatwierdzonych terminów. Zostanie wyświetlony komunikat z błędem, a zmiany nie zostaną zatwierdzone do systemu.
- Zależności czasowe
 - o Częstotliwość wykonania: W zależności od potrzeb serwisu i zakresu ingerencji w działania użytkowników, przewidywane przeciętne użycie od 1 do 10 razy w tygodniu w przypadku braków żadnych niespodziewanych sytuacji.
 - o Przewidywane spiętrzenie: Przede wszystkim na początku istnienia serwisu, gdy zachodzi potrzeba dodania nowych usługodawców do systemu.
 - o Typowy czas realizacji: 30 sekund.
 - o Maksymalny czas realizacji: 3 minuty.
- Wartości uzyskane przez aktorów po zakończeniu przypadku użycia
 - o Wybrany użytkownik ma zmienione dane, status, rolę lub informacje osobiste zgodnie z dokonanyimi przez administratora modyfikacjami.

2.3 Wymagania нефункционалне

2.3.1 Czas odpowiedzi

Współcześnie rozwijane systemy informatyczne powinny cechować się wysoką responsywnością i krótkim czasem oczekiwania na odpowiedź serwisu po każdym działaniu użytkownika. Te aspekty są kluczowe dla oceny jakości systemu oraz wpływają na ogólne doświadczenia użytkowników korzystających z serwisu. Gdy użytkownik nie otrzymuje informacji zwrotnej w ciągu kilku sekund może zacząć się niecierpliwić oraz wątpić w skuteczność działania aplikacji. Zbyt wolne odpowiedzi w krytycznym przypadku mogą nawet sprawić, że użytkownik postanowi zakończyć współpracę i zrezygnuje z korzystania z systemu. Dlatego dla tej aplikacji zostało postawione wymóg, żeby każda akcja odbywała

się jak najszybciej to możliwe i w standardowych warunkach czas oczekiwania nigdy nie przekraczał 3 sekund.

2.3.2 Kompatybilność z przeglądarkami internetowymi

Użytkownicy niezależnie od konfiguracji sprzętowej, systemu operacyjnego czy używanej przeglądarki internetowej od pewnego standardu powinni mieć dostęp i możliwość skorzystania z wszystkich funkcji aplikacji. W miarę możliwości, aplikacja powinna zachowywać się jednolicie we wszystkich popularnych przeglądarkach internetowych. W tym celu zastosowano nowoczesne technologie i rozwiązania, które są powszechnie dostępne, sprawdzone i obsługują większość udziału w rynku. Ponadto aplikacja została przetestowana na różnych popularnych przeglądarkach internetowych takich jak Mozilla Firefox, Google Chrome, Microsoft Edge czy Brave w celu upewnienia się, że funkcjonalności oraz wygląd aplikacji będą takie same dla wszystkich.

2.3.3 Prosty oraz intuicyjny w obsłudze interfejs

Wielu użytkowników nie posiada zaawansowanych doświadczenia ani umiejętności w zakresie korzystania z Internetu. Co więcej, istnieje duża grupa osób, które nie posługują się językiem angielskim, nawet na poziomie podstawowym. Dlatego, żeby zapewnić, że wszyscy użytkownicy będą mogli tak samo korzystać z systemu stworzono aplikację z czytelnym i prostym interfejsem w języku polskim. Aby zwiększyć spersonalizowanie i poprawić komfort użytkowania, aplikacja oferuje dwie opcje motywów, ciemny i jasny. To pozwala użytkownikom dostosować wygląd serwisu do swoich indywidualnych preferencji. Oba motywy zostały starannie zaprojektowane i zapewniają odpowiednią kolorystykę z właściwym kontrastem i przejrzystością. W celu ułatwienia nawigacji rozmieszczenie elementów w aplikacji jest zgodne z standardowymi konwencjami, co powinno przekładać się na łatwość znalezienia poszczególnych funkcji bez większych problemów nawet dla nieoznajmionego z systemem użytkownika.

3. Porównanie konkurencyjnych rozwiązań na rynku

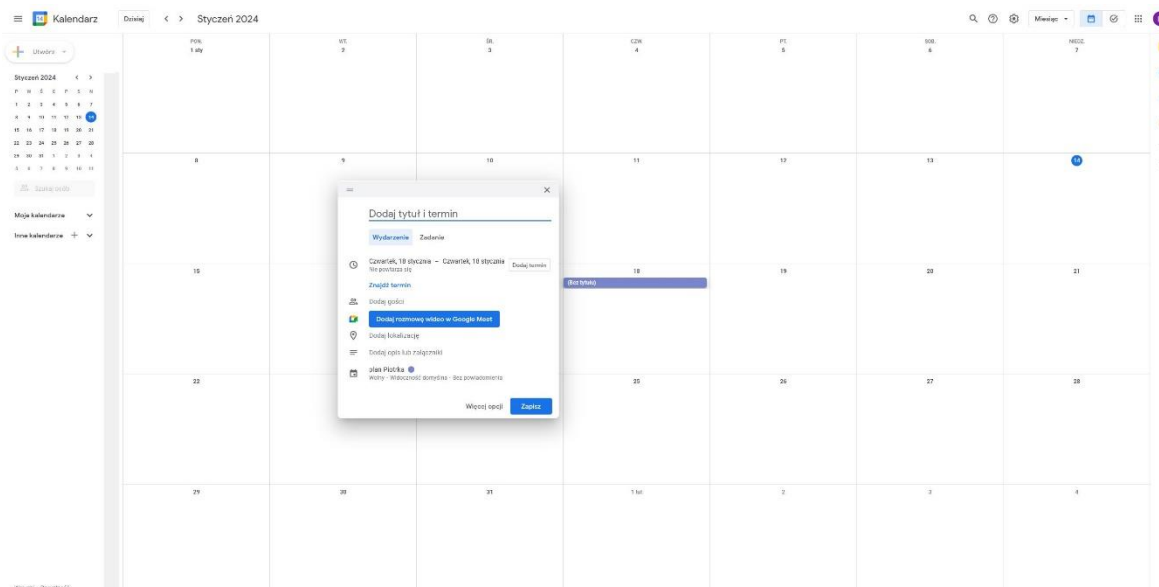
Na rynku istnieje zróżnicowana oferta wielu aplikacji, które dostarczają zbliżoną funkcjonalność, każda z nich ma swoje unikalne cechy, zalety oraz wady. Aplikacje różnią się zarówno docelową grupą do której jest kierowana jak i spektrum możliwości jakie oferują. Jednakże łączącą cechą dla tych aplikacji jest próba rozwiązania wspólnego problemu, czyli jak skutecznie umożliwić ustalanie terminów spotkań. W niniejszym rozdziale skoncentrujemy się na przedstawieniu oraz porównaniu trzech najbardziej popularnych platform, które są rozwijane przez renomowane i szeroko znane firmy. Omówimy trzy różne aplikacje, zwracając uwagę na ich unikatowe podejście do rozwiązania wspólnego problemu w zróżnicowany sposób.

3.1 Kalendarz Google

Kalendarz Google to jedna najpopularniejszych aplikacji umożliwiających korzystanie z kalendarza i ustalanie terminów spotkań. Ten powszechnie używany terminarz umożliwia organizację różnorodnych wydarzeń, zarówno indywidualnych, jak i zespołowych. Posiada liczne funkcjonalności, takie jak powiadomienia, przypomnienia, filtrowanie czy wyszukiwanie. A z racji na duże wsparcie firmy Google kalendarz posiada dużą integrację z innymi usługami Google w tym synchronizację z Gmail, Google Drive czy innymi narzędziami od Google. Serwis jest ciągle rozwijany i wciąż pojawiają się w nim nowe funkcjonalności, a wszelkie błędy są skrupulatnie naprawiane przez zespół odpowiedzialny za rozwój aplikacji. Intuicyjny interfejs, zbliżony do pozostałych narzędzi Google, ułatwia korzystanie z aplikacji, a różnorodne widoki i kolorowe oznaczenia różnych wydarzeń zwiększają przejrzystość kalendarza.

Niestety, jedną z wad kalendarza Google jest brak domyślnej walidacji, która uniemożliwiałaby nakładania się czasowo na siebie spotkań jednego użytkownika. Nakładające się spotkania mogą być przyczyną konfliktów oraz są w większości przypadków stanowią niepożądane zjawisko. Ręczne monitorowanie nakładających się wydarzeń może być nieskuteczne i zawodne. Dodatkowo w kalendarzu Google brakuje funkcji, w którym jedna ze stron, na przykład klient, oferuje propozycję terminu, a dopiero druga strona, na przykład usługodawca, zatwierdza podany termin. Nie ma też możliwości aby pierwsza strona np. usługodawca dodawała termin w którym jest dostępna, a druga strona np. klient dodawała propozycję spotkania w tym terminie lub jego części, a następnie

znowu pierwsza strona zatwierdzała lub odrzucała propozycję drugiej strony. Ponadto ze względu na ogromną liczbę funkcjonalności kalendarz może być przytłaczający dla mniej zaawansowanych użytkowników internetowych. Pomimo tych wad kalendarz Google wciąż jest jednym z najlepszych rozwiązań dostępnych na rynku.



Rys. 3.1 Kalendarz Google

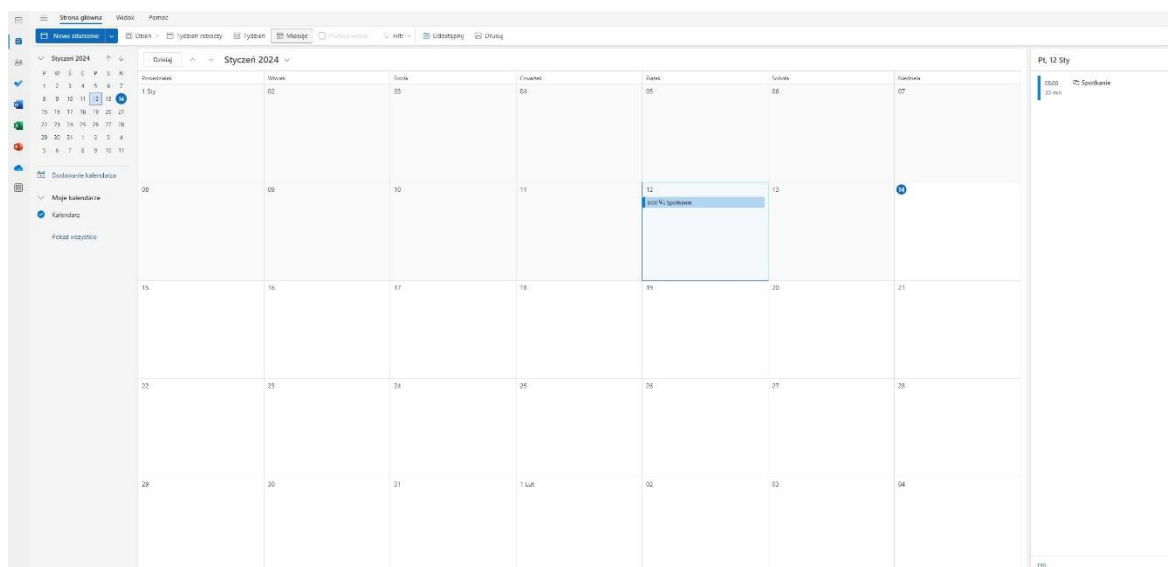
Źródło: Google Calendar, <https://calendar.google.com/calendar> z 14.01.2024

3.2 Microsoft Outlook Calendar

Microsoft Outlook Calendar to konkurencyjne rozwiązanie oferowane przez firmę Microsoft. Narzędzie to umożliwia efektywne zarządzania kalendarzem, własnymi wydarzeniami oraz organizację spotkań pomiędzy wieloma użytkownikami. Stanowi integralną część większego pakietu biurowego Microsoft Outlook i posiada domyślną integrację z pocztą Outlook, umożliwiając automatyczne wysyłanie zaproszeń do spotkania i otrzymywania powiadomień o nadchodzących wydarzeniach. Posiada także możliwość łatwej synchronizacji z pozostałymi narzędziami od Microsoftu takimi jak Microsoft Teams czy Microsoft 365. Aplikacja stale ewoluuje, wprowadzając nowe funkcjonalności, a błędy są systematycznie naprawiane przez zespół inżynierów firmy Microsoft.

Jednakże, istnieją pewne wady związane z korzystaniem z Outlook Calendar. Konieczność posiadania konta Microsoft, może stanowić istotne ograniczenie dla dużej części użytkowników preferujących używanie innych platform. Złożoność i liczne rozbudowane funkcjonalności mogą sprawić, że aplikacja będzie trudna w obsłudze dla osób niezaznajomionych z jej funkcjonowaniem, w tym liczne widoki mogą powodować

konsternacje wśród nowych użytkowników. Powodując potrzebę wdrożenia potencjalnych użytkowników do działania aplikacji, co jest niepożądanym rezultatem w przypadku relacji pomiędzy usługodawcami, a potencjalnymi klientami. Kolejnym istotnym mankamentem Microsoft Outlook Calendar jest brak domyślnej walidacji czasowej terminów, co może prowadzić do nakładania się terminów spotkań pojedynczego użytkownika i powstawania konfliktów w jego harmonogramie. Mimo tych wad Microsoft Outlook Calendar pozostaje potężnym narzędziem z dużym potencjałem rozwoju, szczególnie dla użytkowników już korzystających z innych produktów firmy Microsoft.



Rys. 3.2 Kalendarz w aplikacji Microsoft Outlook Calendar

Źródło: Microsoft Outlook Calendar, <https://outlook.office.com/calendar/view/month> z 15.01.2024

3.3 Doodle

Doodle to platforma umożliwiająca sprawną organizację terminów spotkań, zarówno indywidualnych jak i grupowych. Użytkownicy mogą w szybki sposób ustawić swój harmonogram zarówno dostępnych terminów jak i zaplanowanych spotkań. Doodle posiada system powiadomień i przypomnień, automatycznie informując użytkowników za pomocą wiadomości e-mail. Aplikacja oferuje zarówno możliwość ustawiania lokalnych spotkań jak i wideokonferencji integrując się z popularnymi narzędziami takimi jak Google Meet, Zoom, Microsoft Teams, i wieloma innymi.

Nie mniej jednak główną wadą Doodla jest ograniczona funkcjonalność w bezpłatnej wersji oraz możliwość występowania reklam w darmowym pakiecie. Aplikacja ma możliwość integracji z Outlook oraz Kalendarze Google, ale tylko w płatnej wersji. Ponadto mimo przyjaznego interfejsu śledzenia większej ilości wydarzeń w serwisie może sprawić

pewne trudności. Podobnie jak w przypadku Kalendarza Google czy Microsoft Outlook Calendar brakuje możliwości ustawienie domyślnego blokowania nachodzących się wydarzeń dla pojedynczego użytkownika w celu uniknięcia konfliktów terminów. Należy także zauważyć, że aktualnie Doodle nie posiada polskiej wersji językowej, co może stanowić znaczący problemem dla grupy osób nie posługującej się językami obcymi.

Add your times


Duration

15 min 30 min **60 min** All day Custom

Week Month

← → Today January 14 - January 20 Poland, Warsaw (GMT+1) ▾

	SUN 14	MON 15	TUE 16	WED 17	THU 18	FRI 19	SAT 20
All day							
9:00 AM							
10:00 AM							
11:00 AM							
12:00 PM							
1:00 PM							
2:00 PM							
3:00 PM							
4:00 PM							
5:00 PM							

 Connect your calendar

Rys. 3.3 Kalendarz w Doodle

Źródło: Doodle, <https://doodle.com/meeting/organize/groups> z 15.01.2024

4. Zastosowane technologie wraz z dostępnymi alternatywami

W niniejszym rozdziale przedstawione zostaną użyte technologie oraz narzędzia podczas projektowania, tworzenia i implementacji aplikacji, wraz z alternatywami rozwiązaniami dostępnymi na rynku i uzasadnieniem dlaczego wyboru konkretnej technologii. W rozważaniu wyboru technologii były brane pod uwagę język programowania, powiązane biblioteki oraz frameworki, wtyczki (ang. plug-in), rozszerzenia (ang. extension), dostępne środowisko programistyczne oraz pozostałe narzędzia. Podczas analizy poszczególnych technologii zostanie wyjaśnione jaki wpływ miała dana technologia na rozwój projektu oraz dlaczego została wybrana wśród konkurencyjnych rozwiązań. Kluczowymi aspektami podczas rozpatrywania wyboru technologii było, aby technologia była niezawodna, spełniała najnowsze standardy oraz była udostępniona dla społeczności programistycznej. Istotne było istnienie szczegółowej dokumentacji, która pozwoliła na pełne zrozumienie i wykorzystanie całego potencjału danego rozwiązania.

4.1 Technologie po stronie serwera

W kontekście aplikacji istotną rolę odgrywa nie tylko to co jest widoczne dla użytkownika końcowego, ale również serwer (ang. backend), który pełni szereg kluczowych zadań. Serwer odpowiada za złożone zadania takie jak komunikacja z bazą danych, przetwarzanie danych, wykonywanie logiki biznesowej, autoryzacja, obsługa żądań od użytkowników, integracja z zewnętrznymi usługami za pomocą API oraz zarządzaniem zasobami aplikacji. Wszystkie te aspekty są niezbędne dla prawidłowego działania aplikacji. Separacja serwera od interfejsu użytkownika pozwala na realizację logiki biznesowej bez ujawniania jest stronom zewnętrznym, co nie tylko zwiększa bezpieczeństwo, ale również umożliwia dokładną kontrolę dostępu do kluczowych zasobów systemu.

4.1.1 Python

Językiem programowania, który został użyty do stworzenia serwera był Python. Python to interpretowany, obiektowy język wysokiego poziomu ogólnego przeznaczenia, który znajduje zastosowanie w wielu różnych dziedzinach. Składania kodu jest przejrzyste i zbudowane na podobieństwo języka mówionego, a bloki kodu są oznaczone przez wcięcia, eliminując potrzebę stosowania nawiasów klamrowych lub innych znaczników blokowych. Python wspiera programowanie zarówno obiektowe, strukturalne jak i funkcyjne. Typy

danych są sprawdzane dynamicznie, a kolektor śmieci (ang. Garbage collection) odpowiada za zarządzanie pamięcią. Język został stworzony z myślą o redukcji powtarzającego się kodu poprzez wykorzystanie obiektowego podejścia do programowania. Główna biblioteka wyposażona jest w wiele uniwersalnych rozwiązań a także posiada ogromną ilość bibliotek pomocniczych. Python cieszy się ogromną aktywną społecznością, która nieustannie rozwija nowe biblioteki i frameworki. Język jest dostęp na różnych platformach takich jak Windows, Linux czy macOS. Wybór autora na stworzenia serwera w języka programowania Python wynikał z chęci użycia popularnego frameworku Django służącego do tworzenia aplikacji webowych oraz pozytywnych doświadczeń autora z tym językiem [1].

4.1.2 Django

Django to otwarty źródłowy framework przeznaczony do tworzenia aplikacji internetowych. Django zostało napisane i jest oparte o język programowania Python. Django jest głównie stosowane do tworzenia serwerów internetowych. Django wykorzystuje wzorzec architektoniczny model-widok-szablon (ang. model-view-template), w którym model odpowiada za strukturę i komunikacją z bazą danych, widok za przetwarzanie logiki biznesowej, a szablon za prezentację danych na stronie w formie HTML (hipertekstowy język znaczników). Jednym z głównych założeń przyświecających Django jest redukcja powtarzania kodu poprzez wykorzystanie komponentów wielokrotnego użytku. Domyślnym systemem zarządzania baz danych dla Django jest SQLite, ale oficjalnie wspiera ono też takie systemy bazodanowe jak MySQL, Oracle, PostgreSQL czy MariaDB. Framework ten posiada również posiada ORM (mapowanie obiektowo relacyjne), które ułatwia pracę z bazami danych i umożliwia korzystanie z nich bez konieczności bezpośredniego tworzenia zapytań SQL [2].

Głównymi powodami, które zdecydowały o wyborze przez autora tej technologii do tworzenia części serwerowej było:

- możliwość stworzenia aplikacji internetowej w prosty i efektywny sposób
- znajomość języka programowania na którym bazuje Django przez autora i jego wcześniejsze doświadczenie w używaniu tego frameworka
- nowoczesność, rozwojowość i bezpieczeństwo technologii
- aktywna i pomocna społeczność programistów korzystających z tego rozwiązania
- dobrze i szczegółowo udokumentowana funkcjonalność wraz z przykładami użycia

- intuicyjne i zrozumiałe komunikaty błędów, które ułatwiają debugowanie i identyfikację napotkanych problemów podczas tworzenia oprogramowania
- wbudowany panel administratorski umożliwiający zarządzanie danymi

4.1.3 SQLite3

Sqlite3 to aktualnie najnowsza wersja SQLite. SQLite to otwarty źródłowy system zarządzania relacyjną bazą danych. Jest on domyślnym systemem zarządzania bazami danych dla projektów Django. SQLite implementuje większość standardu SQL-92 dla SQL, co oznacza, że obsługuje szerokie spektrum operacji i funkcjonalności związanych z bazami danych. Jego lekka struktura jest w zupełności wystarczająca dla małych i średnich aplikacji.

4.2 Technologie po stronie klienta

Dla pozytywnego odbioru aplikacji przez użytkowników, oprócz działania serwera i funkcjonalności, niezwykle ważny jest interfejs. To właśnie interfejs jest głównym czynnikiem, który decyduje o wrażeniach użytkownika (ang. UX – User Experience). Istotne jest, aby widoki charakteryzowała dobra kolorystyka, kontrast, spójność stylów, a także, aby aplikacja była responsywna, a użytkownik nigdy nie musiał długo czekać na reakcję. Interfejs powinien być intuicyjny, logiczny oraz łatwy w obsłudze.

4.2.1 JavaScript

JavaScript to skryptowy język programowania, który wraz z HTML i CSS (kaskadowe arkusze stylów) stanowi podstawowe narzędzie do tworzenia stron internetowych. Jest niekwestionowanie najpopularniejszym językiem programowania używanym do tworzenia aplikacji internetowych po stronie klienta. Pomimo powstawania nowych języków takich jak np. TypeScript wciąż JavaScript dominuje na rynku i jest stosowany w większości stron internetowych. JavaScript jest językiem dynamicznie typowanym. JavaScript pozwala na tworzenie bardziej dynamicznych stron internetowych i jest głównie stosowany do zapewnienia interakcji poprzez reakcje na zdarzenia i działania użytkownika. Język ten pozwala na pewien stopień walidacji danych wprowadzanych do formularza bez konieczności angażowania serwera, a także umożliwia implementację zaawansowanych efektów wizualnych na stronie. JavaScript obsługuje asynchroniczność poprzez mechanizmy takie jak obietnice (ang. promises) oraz funkcje zwrotne (ang. callbacks). JavaScript posiada wiele popularnych i rozbudowanych bibliotek oraz frameworków takich

jak jQuery, React, Vue.js czy Angular, które oferują szereg funkcjonalności i ułatwiają pracę programistyczną.

4.2.2 jQuery

W celu zapewnienia możliwości asynchronicznego przetwarzania danych oraz zlikwidowania potrzeby przeładowania strony po wysłaniu żądania do serwera. Aplikacja wykorzystuje mechanizm przesyłanie danych za pomocą jQuery.ajax. jQuery jest to biblioteka Javascript używana przede wszystkim do manipulacji obiektywnym modelem dokumentu (ang. DOM - Document Object Model), zarządzaniem zdarzeniami oraz ułatwiania używania asynchronicznego JavaScript i XML (ang. AJAX – Asynchronous Javascript and XML). Cechą charakterystyczną jQuery jest kompaktowość, kod napisany w jQuery jest zazwyczaj krótszy, bardziej zwięzły i czytelniejszy niż adekwatny kod powstały w czystym JavaScript. Warto zaznaczyć, że pomimo spadku popularności wśród programistów w ostatnich latach jQuery wciąż jest jedną z najczęściej używanych bibliotek JavaScript. Wszechstronność połączona z prostotą użycia sprawiają, że duża część programistów wciąż decyduje się na użycie tego rozwiązania w swoich projektach.

4.2.3 Tailwind CSS

Tailwind CSS to otwarty źródłowy CSS framework, który umożliwia efektywne i szybkie projektowanie oraz implementowanie stylów interfejsów aplikacji internetowych, poprzez dostarczenie gotowego zbioru klas styli CSS (kaskadowe arkusze stylów). Tailwind CSS umożliwia używanie predefiniowanych klas, dzięki czemu programista może zaoszczędzić czas na tworzeniu styli oraz mieć bardziej spójny stylistycznie projekt. Kolejną zaletą są bezpieczne zmiany gdzie dana klasa dotyczy lokalnej witryny i nie ma ryzyka uszkodzenia pozostałych witryn poprzez zmianę klasy na danej stronie. Przyjęta konwencja nazewnictwa klas jest klarowna i w logiczny sposób opisuje jakie zmiany wprowadza na dany element. Ponadto Tailwind CSS dostarcza narzędzia do definiowania różnych stylów w zależności od różnych rozmiarów ekranów na których jest wyświetlana strona, zdarzeń czy też wybranego motywu w przeglądarce. Sam framework jest bardzo prosty w użyciu i łatwy do opanowania.

4.2.4 FullCalendar

FullCalendar to biblioteka JavaScript przeznaczona do łatwego tworzenia interaktywnych widoków kalendarza w aplikacjach internetowych. FullCalendar posiada wiele interaktywnych funkcji takich jak wyświetlanie, dodawanie, edycja czy usuwanie różnych wydarzeń w formie kalendarza. Umożliwia w prosty sposób dostosowywanie wyglądu kalendarza oraz posiada domyślnie zaimplementowany wybór różnych widoków takich jak widok dnia, tygodnia, miesiąca czy roku. Biblioteka daje możliwość obsługi szerokiego zakresu zdarzeń dotyczących kalendarza takich jak zmiana widoku, kliknięcia czy nawigacja użytkownika po kalendarzu. FullCalendar pozwala integrować się z wieloma źródłami danych z których może pobierać dane na temat wydarzeń w kalendarzu, co umożliwia dynamiczne wczytywanie i aktualizowanie wydarzeń w kalendarzu bez potrzeby odświeżania strony. Niewątpliwymi zaletami biblioteki są prostota w użyciu oraz elastyczność, która sprawi, że kalendarz jest responsywny, dopasowuje się do różnych rozmiarów ekranu i adaptuje się do zmiany wielkości okna. Dzięki temu, że FullCalendar jest jedną z najpopularniejszych bibliotek JavaScript z kalendarzem posiada on rozwiniętą dokumentację i aktywną społeczność. Zastosowanie tej biblioteki umożliwiło stworzenie przejrzyste i intuicyjnego interfejsu wyposażonego w responsywny kalendarz z licznymi funkcjami pozwalającymi na komfortowe korzystanie z aplikacji.

4.2.5 DataTables

DataTables to wtyczka (ang. plug-in) do biblioteki jQuery, zaprojektowana do wyświetlania interaktywnych, spersonalizowanych tabel danych na stronie internetowej. Wtyczka oferuje szereg funkcji, które ułatwiają zarządzanie tabelami, co sprawia, że jest idealna do implementacji w projektach, gdzie wymagane jest efektywne prezentowanie dużych zbiorów danych w tabeli. Poniżej znajduje się przegląd głównych funkcji, które umożliwia DataTables:

- Filtrowanie tabeli na podstawie określonych kryteriów
- Wyszukiwanie i wyświetlanie tylko określonych danych
- Sortowanie tabeli, istnieje możliwość sortowania zarówno po pojedynczej kolumnie jak i wielu kolumnach
- podział na strony, w przypadku istnienia większej ilości danych, przydatnym może być podział tabeli na strony i wyświetlanie tylko części danych jednocześnie

- grupowanie wierszy tabeli według wcześniej zdefiniowanych własnych warunków grupowania
- spersonalizowany opis tabeli na przykład informację o liczbie aktualnie wyświetlanych wierszy

DataTables została wykorzystana w tym projekcie, ponieważ jest to bardzo łatwa w użyciu wtyczka, która pozwala w prosty i szybki sposób stworzyć interaktywne tabele z dużą ilością przydatnych funkcjonalności dotyczących zarządzania tabelami. Wtyczka umożliwia łatwe spersonalizowanie tabel do własnych potrzeb i jest kompatybilna z frameworkami stosowanymi do kontroli stylów takimi jak Tailwind CSS czy Bootstrap.

4.3 Pozostałe narzędzia

4.3.1 Pycharm

Pycharm to zintegrowane środowisko programistyczne (ang. IDE - Integrated Development Environment) od firmy JetBrains dla języka programowania Python. Jest to jedno z najpopularniejszych IDE wśród programistów Python oferujące szereg przydatnych funkcji podczas rozwoju projektu takich jak [3]:

- Inteligentny edytor kodu podświetlający składnię języka Python oraz umożliwiający autouzupełnianie kodu
- Sprawdzanie i weryfikacje błędów składniowych na bieżąco w trakcie pisania kodu
- Refaktoryzacja Pythona
- Zestaw narzędzi wspierających tworzenie aplikacji internetowych w Django
- Wbudowana integracja z systemem kontroli wersji Git
- Wbudowany termin
- Możliwość instalacji dodatkowych pluginów
- Debugowanie
- Zintegrowane testy jednostkowe

4.3.2 Git

Pomocnym narzędziem wspomagającym rozwój oprogramowania jest Git. Jest to rozproszony system kontroli wersji umożliwiający śledzenie i zapisywanie wprowadzonych zmian kodu, co przyczynia się do sprawniejszego zarządzania projektem. Wszystkie zmiany

kodu są przechowywane w historii, co nie tylko zwiększa bezpieczeństwo, ale także minimalizuje ryzyko utraty kodu w przypadku awarii. Jednym z popularnych serwisów hostingowych wykorzystujących system kontroli wersji Git jest Github. Github umożliwia przechowywanie kodu w zdalnych repozytoriach na serwerze. Narzędzie kontroli wersji takie jak Git jest przydatnym, a wręcz niezbędnym narzędziem podczas tworzenia oprogramowania. Dzięki niemu zespół programistyczny może nie tylko skutecznie współpracować, monitorować zmiany w kodzie oraz zachować przejrzystość i porządek w historii projektu, ale także ma gwarancje bezpieczeństwa i możliwość przywrócenia wcześniejszej wersji kodu.

4.4 Konkurencyjne technologie

4.4.1 Java

Java to obiektowy język programowania z silnym typowaniem. Wszystkie obiekty są instancjami klasy `Object` z której dziedziczą podstawowe zachowania i właściwości. Java umożliwia dziedziczenie tylko po jednej klasie, ale możliwe jest też dziedziczenie po wielu interfejsach. Kod źródłowy jest kompilowany do kodu bajtowego, a następnie wykorzystywany przez maszynę wirtualną. Ta właściwość sprawia, że kod Javy jest niezależny od systemu operacyjnego i procesora, a wykonuje go tzw. wirtualna maszyna Javy, która (między innymi) tłumaczy kod uniwersalny na kod dostosowany do specyfiki konkretnego systemu operacyjnego i procesora. W tej chwili wirtualna maszyna Javy jest już dostępna dla większości systemów operacyjnych i procesorów. Jednak z uwagi na to, że kod pośredni jest interpretowany, taki program jest wolniejszy niż kompilowany do kodu maszynowego. Z tego względu maszynę wirtualną często uzupełnia się o kompilator JIT. [4] W Javie zarządzanie pamięcią odbywa się za pomocą kolektora śmieci (ang. *Garbage collection*)

4.4.2 C#

C# to bezpiecznie typowany, obiektowy język programowania wychodzący spod szyldu firmy Microsoft. Język balansuje pomiędzy prostotą, wyrazistością i wydajnością. Jest kompilowany do specjalnego rodzaju kodu pośredniego CIL (*Common Intermediate Language*), który jest uruchamiany na platformie .NET. W C# zarządzanie pamięcią jest zautomatyzowane, dzięki mechanizmowi kolekcji śmieci (ang. *Garbage collection*).

Hierarchia dziedziczenia w C# opiera się na klasie bazowej System.Object, co oznacza, że wszystkie typy w C# dziedziczą z tej podstawowej klasy. Język ten wspiera paradygmaty programowania obiektowego, proceduralnego i funkcyjnego, dzięki czemu jest elastyczny i dostosowuje się do różnych stylów programowania. [5]

4.4.3 Bootstrap

Bootstrap to otwarty źródłowy CSS framework pozwalający na tworzenie responsywnych, przyjaznych dla urządzeń mobilnych stron internetowych i aplikacji. Zawiera zarówno elementy HTML, CSS oraz JavaScriptu. Jest łatwy w użyciu, w pełni konfigurowalny i posiada dużą ilość gotowych komponentów.

5. Projekt aplikacji

5.1 Architektura

Stworzona aplikacja jest aplikacją typu klient-serwer i jest zbudowana w oparciu o architekturę warstwową [6] bazującą na wzorcu architektonicznym model-widok-szablon (ang. model-view-template) [7]. Architektura ta polega na podziale odpowiedzialności aplikacji na poszczególne warstwy takie jak warstwa danych czyli model, warstwa logiki biznesowej czyli view i warstwa prezentacji czyli template.

5.2 Struktura projektu

Struktura projektu jest podzielona na 3 aplikacje współpracujące ze sobą. Każda z aplikacji ma swoje unikalne zadanie.

Podstawowa aplikacja utworzona wraz z projektem Django:

- **settings** – plik w którym znajdują się ustawienia projektu Django. Są w nim skonfigurowane kluczowe kwestie projektu takie jak:
 - o połączenie z bazą danych,
 - o oprogramowanie pośrednie (ang. middleware),
 - o lokalizacja katalogów z szablonami, plikami statycznymi oraz mediami,
 - o wymagana walidacja autoryzacyjna,
 - o ustawienia hosta mailowego,
 - o strefa czasowa,
 - o lokalizacja,
 - o sekretny klucz,
 - o dozwolone adresy ip w projekcie.
- **urls** – główny plik definiujący ścieżki url dla całego projektu. Zawiera ścieżki z pozostałych aplikacji oraz ścieżki do plików medialnych i panelu administracyjnego.
- **asgi** – plik dający możliwość asynchronicznej obsługi żądań w Django.
- **wsgi** – plik służący jako połączenie pomiędzy serwerem, a aplikacją webową napisaną w języku Python. Nie jest używany podczas rozwoju projektu, ale jest używany w wdrożeniu projektu w celu prawidłowego skonfigurowania z serwerem na którym znajduje się dany projekt.

Główną aplikację w której znajduje się większość funkcjonalności zawartej w projekcie:

- **templates** – katalog zawierający wszystkie szablony głównej aplikacji. Szablony w Django są używane do generowania HTML w oparciu o dane przekazane z widoków.
- **templatetags** – katalog zawierający niestandardowe tagi szablonów Django. Umożliwia tworzenie własnych tagów szablonów oferujących dodatkową funkcjonalność w szablonach.
- **static** – katalog stworzony do przechowywania wszystkich plików statycznych aplikacji, takich jak skryptów napisanych w JavaScript, arkuszy stylów CSS oraz obrazków czy ikon.
- **media** – katalog, w którym mogą znajdować się pliki multimedialne przesyłane przez użytkowników, takie jak obrazki czy widoki. W kontekście tego projektu są to zdjęcia profilowe użytkowników.
- **context_processors** – zawiera pliki z funkcjami dodającymi dodatkowe dane do kontekstów wszystkich szablonów w aplikacji. Dane te są globalne w zakresie aplikacji i dostępne we wszystkich szablonach.
- **urls** – plik w którym definiuje się ścieżki url powiązane z konkretnymi widokami.
- **forms** – plik w którym przechowuje się formularze Django. Formularze używane są do przetwarzania danych wejściowych od użytkownika, takich jak pola w formularzach HTML.
- **views** – plik zawierający widoki Django występujące w aplikacji. Widoki obsługują logikę biznesową, przetwarzają dane oraz renderują odpowiedzi do użytkownika za pomocą szablonów.
- **decorators** – służą do modyfikowania podstawowego zachowania widoku. Umożliwiają opakowanie danego widoku w dodatkową funkcjonalność, pozwalając tym na redukcję powtarzającego się kodu. W tym projekcie zostały użyte do autoryzacji oraz walidacji roli użytkownika.
- **functions** – plik zawierający wszystkie dodatkowe funkcje, które nie renderują szablonów, ani nie zwracają wyników w JSON. W pliku tym znajduje się również klasa pozwalająca na wielowątkowe operacje i została ona użyta do wysyłania maili w drugim wątku.
- **admin** – plik konfiguracyjny stanowiący rozszerzenie dla panelu administracyjnego Django. Umożliwia efektywne tworzenie tabel dla wszystkich używanych modeli w panelu administracyjnym.

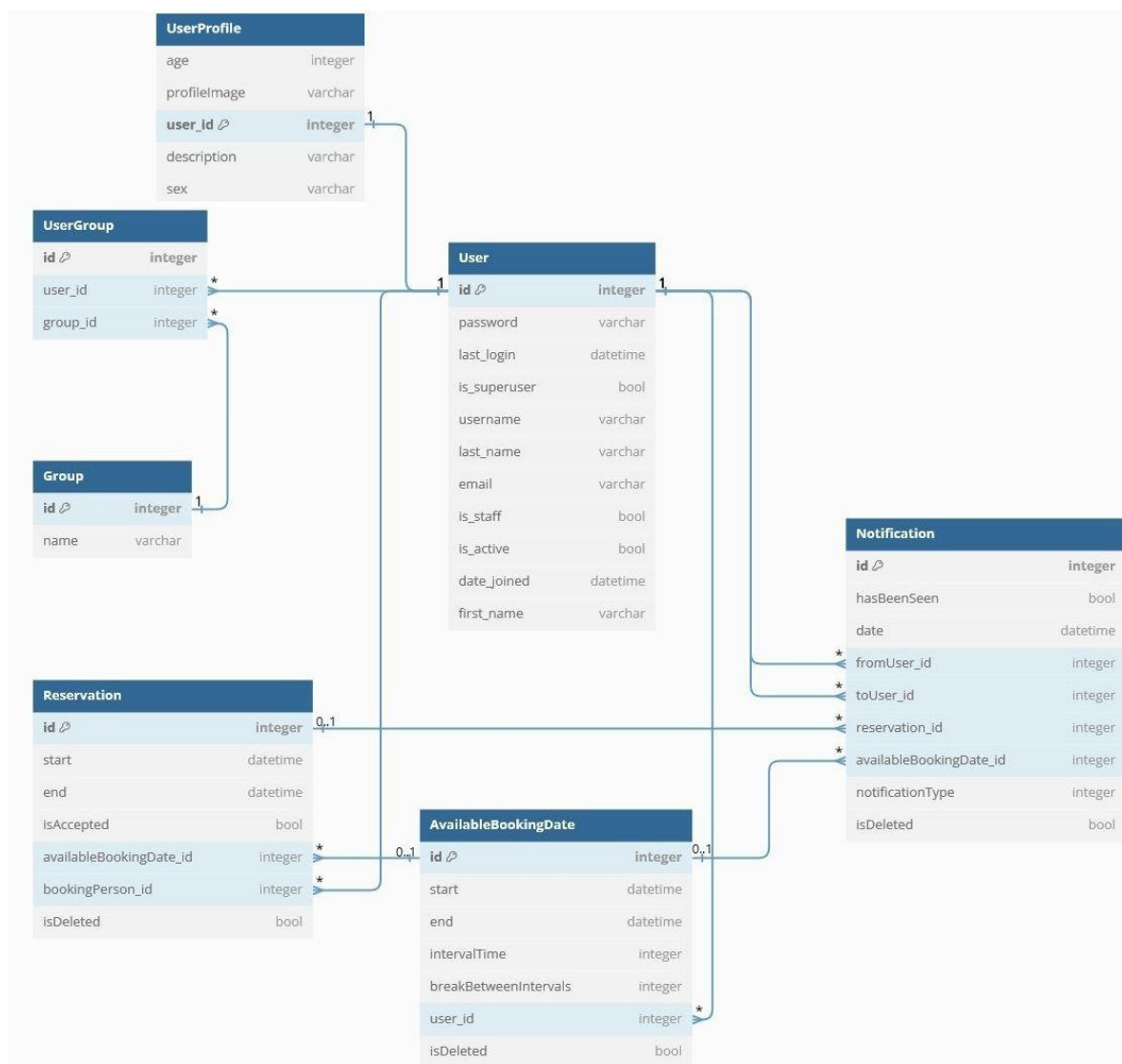
- **models** – zawiera wszystkie klasy modeli Django wraz funkcją pozwalającą na automatyczne tworzenie obiektu klasy UserProfile po stworzeniu obiektu klasy User. Modele definiują strukturę tabel w bazie danych.
- **migrations** – katalog w którym przechowywane wszystkie pliki migracje Django. Migracje są mechanizmem pozwalającym na zmianę struktury bazy danych w celu dopasowania jej struktury do klas modeli. Pliki migracyjne przechowują wszystkie zmiany w modelach klas i są automatycznie generowane przez Django po wykonaniu poleceń python manage.py makemigrations oraz migrate.
- **tests** – plik z wszystkimi testami projektu Django.

Aplikacje powiązaną z motywami i frameworkiem Tailwind CSS. Aplikacja ta jest odpowiedzialna za możliwość zmiany motywu oraz dostarczenie dla całego projektu klas wbudowanych w Tailwind CSS.

- **static** – przechowuje wszystkich pliki statyczne aplikacji w tym konfiguracyjne skrypty django-browser-reload.
- **static_src** – zawiera node_modules oraz pliki konfiguracyjne Tailwind CSS.
- **templates** – katalog z dodatkowymi szablonami. To w tym katalogu jest base.html, który stanowi podstawę dla większości szablonów w projekcie.
- **views** – plik zawiera widok odpowiedzialny za przełączanie motywu pomiędzy motywem ciemnym a jasnym.
- **context_processors** – zawiera funkcję sprawdzającą jaki aktualnie jest ustawiony motyw w serwisie.

5.3 Baza danych

Tabele baza danych w projekcie zostały utworzone po napisaniu modelki klas w Django i użyciu mechanizmu migracji. Wykorzystano tutaj podejście „kodu najpierw, baza danych później” (ang. Code First, Database Later). Było to możliwe, ponieważ Django wspiera mapowanie obiektowo-relacyjne (ang. ORM - Object-Relational Mapping) dla baz danych o relacyjnym charakterze.



Rys. 5.1 Schemat relacji bazy danych (opracowanie własne)

- **User** – tabela zawierająca dane wszystkich zarejestrowanych użytkowników aplikacji. Tabela zawiera zarówno podstawowe dane użytkowników takie jak hasło, login, nazwa użytkownika, imię, nazwisko, email oraz część danych dotyczących uprawnień użytkownika takich jak czy użytkownik ma uprawnienia administratorskie, wszystkie możliwe uprawnienia, czy jest aktywny oraz kiedy się zarejestrował na stronę.
- **Group** – tabela zawierająca nazwy grup użytkowników. To właśnie ta tabela pozwala na zarządzanie wszystkimi istniejącymi grupami w systemie. Przechowuje ona nazwy grup oraz ich identyfikatory.
- **UserGroup** – tabela łącząca pomiędzy tabelami User oraz Group. Pozwala na uniknięcie bezpośredniej relacji wiele do wielu i zastąpienia jej dwoma relacjami jeden do wielu. Zawiera takie pola jak identyfikator grupy, odwołującym się

powiązanej grupy, pole identyfikatora użytkownika wskazujące na rekord w tabeli użytkownika oraz własne identyfikator.

- **UserProfile** – tabela przechowująca dane profilowe użytkowników nie zawarte w tabeli User. Tabela jest połączona relacją jeden do jednego z tabelą User. Każdy użytkownika posiada swój profil. Profil jest automatycznie tworzony wraz z zarejestrowaniem się użytkownika. UserProfile przechowuje dane takie jak wiek, płeć, opis profilowy oraz link do zdjęcia profilowego.
- **AvailableBookingDate** – tabela zawierająca dane wszystkich dostępnych terminów użytkowników. Zawiera informacje o dacie rozpoczęcia i zakończenia się dostępnego terminu, identyfikator użytkownika, którego dotyczy dostępny termin oraz może opcjonalnie zawierać pojedynczy przedział czasowy wizyty i czas przerwy na które dzielony jest dostępny termin. W przypadku istnienia wartości czasu wizyty jest nałożone ograniczenie, że wielokrotność czasu wizyty z przerwą musi równać się całemu przedziałowi czasu. Poza tym posiada wartość typu logicznego, czy dany termin został usunięty, domyślnie ta wartość jest fałszywa, lecz gdy ktoś usunie dany rekord, zamiast usuwać rekordu z tabeli, wartość tego pola zmienia się na prawdę. Jest to tak zwane miękkie usuwanie (ang. soft delete), które umożliwia łatwe przywracanie danych oraz jest szybsze od klasycznego usunięcia całego wiersza, ponieważ operacji zaktualizowania zazwyczaj są szybsze od operacji usuwania. Wadami takiego rozwiązania jest przechowywanie większej ilości danych w bazie danych.
- **Reservation** – tabela zawierająca dane wszystkich rezerwacji. Każda rezerwacja ma swój czas początkowy i końcowy oraz identyfikator użytkownika, który chce zarezerwować termin. Rezerwacja może mieć identyfikator dostępnego terminu, ale nie musi. Brak identyfikatora dostępnego terminu oznacza, że klient zaproponował swój własny termin. Jeśli rezerwacja posiada identyfikator dostępnego terminu oznacza to, że klient zaproponował termin rezerwacji w czasie dostępności danego usługodawcy. Pole isAccepted odpowiada za wartość logiczną, czy rezerwacja została zatwierdzona przez usługodawcę. Podobnie jak w przypadku AvailableBookingDate rezerwacje również mają pole z wartością logiczną isDeleted, które umożliwia miękkie usuwanie i wszystkie konsekwencje z tym związane.
- **Notification** – tabela zawierająca dane wszystkich powiadomień. Większość powiadomień jest generowana automatycznie gdy jeden użytkownik wpłynie na terminy drugiego na przykład poprzez potwierdzenie lub odrzucenie terminu, z tego

powodu tabela posiada dwa pola powiązane z identyfikatorami użytkowników fromUser_id oraz toUser_id. Posiada również pole notificationType, które definiuje jaki to jest rodzaj powiadomienia oraz pole hasBeenSeen służące jako wartość logiczna czy dane powiadomienie zostało odczytane przez użytkownika. Pole date przechowuje datę stworzenia powiadomienia. Notyfikacja może również opcjonalnie posiadać identyfikator dostępnego terminu lub rezerwacji, którego dotyczy powiadomienie. Powiadomienia posiadają miękkie usuwanie.

6. Implementacyjne rozwiązanie projektu

Niniejszy rozdział zostanie poświęcony na szczegóły implementacji aplikacji, zastosowane rodzaje testów oraz sposób wdrożenia aplikacji na serwer platformy hostingowej.

6.1 Wybrane szczegóły implementacyjne

W tym rozdziale zostaną przedstawione kluczowe szczegóły z etapu implementacji aplikacji z wybranymi fragmentami kodów aplikacji, opisem ich działania, a także wytłumaczeniem potrzeby ich użycia oraz korzyści z tego wynikających.

6.1.1 Komunikacja frontendu z backendem

Komunikacja frontendu z backendem odbywa się poprzez użycie jQuery i wysyłanie danych za pomocą Ajaxa co umożliwia stworzenie asynchronicznej aplikacji webowej. Dane są wysyłane z skryptu w JavaScript w formacie JSON, a następnie otrzymywane i przetwarzane w konkretnym widoku Django. Funkcja przyjmuje jako parametry adres url danego widoku, dane oraz parametr będący wartością logiczną odpowiadający za to czy błąd w odpowiedzi może spowodować cofnięcie operacji kalendarza. Również niezwykle ważnym jest zawarcie csrf_token-a w nagłówku wysłanej wiadomości oraz sprecyzowanie rodzaju metody protokołu http na POST, ponieważ bez tego Django nie zaakceptuje żądania i zwróci błąd serwera. W prawidłowym przypadku wysłania danych odpowiedni widok w Django odczyta wszystkie zawarte dane w wiadomości, przetworzy je, wykona swoją funkcjonalność, a następnie po przetworzeniu danych widok w Django zwraca wiadomość w formacie JSON. Prawidłowe zakończenie operacji po stronie serwera zakańcza się wysłaniem wiadomości z statusem 'success' lub 'successWithoutNeedToRefetch'. Natomiast jeśli wystąpią jakieś błędy w trakcie przetwarzania danych na przykład podczas walidacji na serwerze, wtedy zostanie zwrócona wiadomość do frontendu z informacją błędu i statusem error, a następnie frontend wyświetli odpowiedni komunikat dla użytkownika.

To podejście pozwala na zaktualizowanie wydarzeń w kalendarzu po stronie użytkownika bez potrzeby odświeżania strony, co poprawia wrażenia użytkownika oraz pozwala na wprowadzanie tylko niezbędnych zmian w interfejsie. Sprawia to, że używanie interfejsu jest płynniejsze oraz potrzebne dane na stronie są wczytywane tylko raz na początku wczytania strony, a następnie są tylko aktualizowane.

Na poniższym kodzie źródłowym 6.1 pokazano funkcję napisaną w JavaScript odpowiadającą za przysyłanie danych z strony klienta do serwera.

```
function passDataByAJAX(url, data, possibleNeedRevert=false) {
    $.ajax({
        type: 'POST',
        url: url,
        headers: {
            'X-CSRFToken': csrf_token
        },
        data: data,
        success: function(response) {
            if (response.status === 'success') {
                refetchCalendarEvents();
                return response;
            } else if (response.status === 'successWithoutNeedToRefetch') {
                return response;
            }
            displayBackendError(response.message)
            if (possibleNeedRevert) {
                selectedEventInfo.revert()
            }
            return response;
        },
        error: function (response) {
            return response;
        }
    });
}
```

Kod źródłowy 6.1 Funkcja przesyłające dane do backendu za pomocą AJAX

Kod źródłowy 6.2 pokazuje fragment kodu implementujący przykładowy widok, który odbiera dane wysłane po stronie klienta. W tym przypadku odczytuje identyfikator wiadomości, której ma zostać zmieniony status na odczytaną, a następnie próbuje zmienić dane tego rekordu w obiekcie bazy danych. W przypadku prawidłowego zakończenia operacji zwraca z powrotem komunikat o pomyślnym zakończeniu operacji, a w przypadku wyrzucenia wyjątku przez serwer przy walidacji żądania zwraca komunikat z błędem.

```
def readNotification(request):
    if request.method == 'POST':
        ic(request.POST)
        try:
            responseData = {'status': 'successWithoutNeedToRefetch', 'message': 'Przeczytano powiadomienie.'}
            notificationId = request.POST.get('id')
            notification = Notification.objects.get(pk=int(notificationId))
            notification.hasBeenSeen = True
            notification.save()
        except ValidationError as e:
            ic("Wystąpił błąd: ", e)
            responseData = {'status': 'error', 'message': e.message}
        return JsonResponse(responseData)
```

Kod źródłowy 6.2 Widok służący do zmiany statusu powiadomienia na odczytane

6.1.2 Wysyłanie maili w dodatkowym wątku

Wraz z wykonaniem się wielu działań po stronie klienta istnieje również potrzeba wysyłania powiadomień mailowych dla pozostałych użytkowników w celu poinformowania ich o zmianie sytuacji. Standardowa operacja wysyłania maili zajmuje około dwóch sekund, co stanowi dość długi czas. Jest to sytuacja, w której dobrze byłoby uniknąć opóźnień po stronie klienta. Oczekiwanie dodatkowych dwóch sekund na efekty swoich działań może być uciążliwe dla użytkownika, zwłaszcza gdy potrzebuje szybkiej informacji. Aby umożliwić obie te operacje, zdecydowano się skorzystać z dodatkowego wątku. Klasa `threading.Thread` w języku Python umożliwia zarządzanie dodatkowym wątkiem, który będzie się wykonywał równolegle do głównego wątku. Ten dodatkowy wątek jest używany do obsługi operacji wysyłanie maili, bez blokowania głównego wątku. Dzięki temu użytkownik nie musi czekać na zakończenie operacji mailowej, co zwiększa responsywność interfejsu. Kod źródłowy 6.3 przedstawia funkcję napisaną w języku Python wysyłającą mail przy użyciu klasy `EmailThread`. Klasa ta dziedziczy z klasy `threading.Thread` i pozwala na użycie dodatkowego wątku.


```

def sendMail(subject, message, receivers):
    ic(subject, message, receivers)
    EmailThread(subject, message, receivers).start()

! usage  ~ rzymski *
class EmailThread(threading.Thread):
    ~ rzymski
    def __init__(self, subject, message, recipient_list):
        self.subject = subject
        self.recipient_list = recipient_list
        self.message = message
        threading.Thread.__init__(self)

    ~ rzymski *
    def run(self):
        send_mail(subject=self.subject, message=self.message, from_email='pracainzynierska@gmail.com',
                  recipient_list=self.recipient_list)

```

Kod źródłowy 6.3 Funkcja wysyłająca maile oraz klasa tworząca dodatkowy wątek

6.1.3 Autoryzacja użytkownika podczas logowania

Niezałogowany użytkownik ma ograniczony dostęp do funkcjonalności aplikacji. Aby się zalogować, użytkownik podaje swoje dane w formularzu na stronie logowania, zatwierdza je i wysyła do serwera. Szablon strony logowania (Kod źródłowy 6.4) jest renderowany z wykorzystaniem dedykowanego formularza. Szablon zawiera również w formularzu token CSRF (ang. cross-site request forgery) niezbędny we wszystkich formularzach Django i będący odpowiedzialny za chronienie strony przed atakami CSRF. Kod zawierający token CSRF jest w drugiej linii kodu ‘{% csrf_token %}’.

Po kliknięciu przez użytkownika przycisku „zaloguj się” formularz jest wysyłany za pomocą metody POST do widoku logowania. Widok otrzymuje dane przesłane metodą POST, odczytuje je i przesyła do formularza w celach wstępnej walidacji poprawności danych. Kod źródłowy 6.5 pokazuje implementację formularza, który jest wykorzystywany do renderowania elementów HTML typu wejść (ang. input), a także do późniejszej walidacji poprawności danych podanych przez użytkownika. Formularz logowania posiada tylko dwa pola dla nazwy użytkownika oraz hasła. To formularz określa typ wymaganych danych, narzuca wymóg podania danych do obu pól oraz ogranicza wartości jakie mogą być podane przez użytkownika, na przykład długość użytkownika nie może przekraczać 254 znaków. Formularz pozwala też na stworzenie tekstu zastępczego (ang. placeholder) dla danych wejść przed wpisaniem jakichkolwiek danych przez użytkownika. To w formularzu odbywa się sprawdzenie, czy jest możliwe uwierzytelnienie użytkownika po podanej nazwie

użytkownika i hasło, oraz walidacja tego czy zalogowany użytkownik nie ma statusu zablokowanego.

```
<form method="POST" action="" class="space-y-4 md:space-y-6">
  {% csrf_token %}
  <div class="relative flex items-stretch">
    <div class="flex items-center whitespace-nowrap rounded-l border border-solid border-gray-300 bg-gray-50"
      <i class="fas fa-user text-xl"></i>
    </div>
    {{ form.username|add_class:customClass }}
  </div>
  <div class="relative flex items-stretch">
    <div class="flex items-center whitespace-nowrap rounded-l border border-solid border-gray-300 bg-gray-50"
      <i class="fas fa-key text-xl"></i>
    </div>
    {{ form.password|add_class:customClass }}
  </div>
  <div class="form-text text-red-600 dark:text-red-500 small">
    {{ form.password.errors }}
    {% for error in errors %}
      {% if error == 'Username or password is incorrect' %}
        <p>Nazwa użytkownika lub hasło jest nieprawidłowe.</p>
      {% else %}
        <p>{{ error }}</p>
      {% endif %}
    {% endfor %}
  </div>
  {% if request.GET.next %}
    <input type="hidden" name="next" value="{{request.GET.next}}" />
  {% endif %}
  <div class="login_container">
    <input type="submit" value="Zaloguj się" class="w-full text-gray-50 bg-primary-600 hover:bg-primary-700" />
  </div>
```

Kod źródłowy 6.4 Część szablonu logowania używająca formularza

```
class LoginForm(AuthenticationForm):
    username = forms.CharField(max_length=254, widget=forms.TextInput(
        attrs={'class': 'form-control', 'placeholder': 'Nazwa użytkownika'}), required=True)
    password = forms.CharField(widget=forms.PasswordInput(
        attrs={'class': 'form-control', 'placeholder': 'Hasło'}), required=True)

    @staticmethod
    class Meta:
        model = User
        fields = ['username', 'password']

    @staticmethod
    def confirm_login_allowed(self, user):
        if not user.is_active:
            raise ValidationError("This account is inactive.", code="inactive")
```

Kod źródłowy 6.5 Formularz logowania

Ważnym łącznikiem między stroną klienta, a serwerem z bazą danych w Django są widoki. Kod źródłowy 6.6 przedstawia widok logowania, w tym widoku jeśli używana metoda protokołu http jest inna niż POST widok logowania tworzy formularz logowania, który następnie wysyła za pomocą mechanizmu kontekstu (ang. context) do szablonu.

Pozwala to na wyrenderowanie formularza z wejściami po stronie klienta. Klient po uzupełnieniu danych i zatwierdzeniu, przesyła je z szablonu za pomocą metody POST. Widok wtedy sprawdza walidację danych używając nowego formularza logowania. Jeśli formularz jest walidowany prawidłowo to wtedy widok wyczytuje z żądania nazwę użytkownika oraz hasło. Z zdobytą nazwą użytkownika oraz hasłem widok próbuje się zautoryzować. Jeśli istnieje taki użytkownik wtedy widok się loguje na jego dane. Następnie jeśli użytkownik próbował dostać się na jakąś konkretną podstronę, która wymagała zalogowania to widok odsyła użytkownika do podstrony do której próbował się wcześniej dostać, w przeciwnym wypadku odsyła go do podstawowej strony z kalendarzem. Jeśli nie ma użytkownika o takiej nazwie użytkownika i hasle to wtedy użytkownik pozostanie w szablonie z logowaniem oraz wyświetli się błąd o braku takiego użytkownika w systemie.

```
@unauthenticatedUser
def loginUser(request):
    errors = []
    form = LoginForm()
    if request.method == "POST":
        form = LoginForm(request, request.POST)
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']
            user = authenticate(request, username=username, password=password)
            if user is not None:
                login(request, user)
                if 'next' in request.POST:
                    return redirect(request.POST['next'])
                return redirect('calendar')
            else:
                errors.append('Username or password is incorrect')
    return render(request, 'accounts/login.html', {'form': form, 'errors': errors})
```

Kod źródłowy 6.6 Widok logowania

Żeby zapobiec możliwości próby ponownego zalogowania lub zarejestrowania się, gdy użytkownik jest już aktualnie zalogowany, na widokach logowania oraz rejestracji nałożony jest dekorator sprawdzający czy dany użytkownik na pewno nie jest już zautoryzowany w systemie. Użycie dekoratora widać w pierwszej linii kodu 6.6 z '@unauthenticatedUser'. Kod źródłowy 6.7 przedstawia implementację dekoratora. Jeśli użytkownik jest już zalogowany to zamiast wykonać widok logowania, użytkownik zostanie przeniesiony do strony z kalendarzem. Jest to mechanizm uniemożliwiający ponowną próbę logowania z innego okna przeglądarki lub poprzez skopiowanie adresu url i wklejenia go do pola przeglądarki.

```

def unauthenticatedUser(viewFunc):
    @rzymski
    def wrappedFunc(request, *args, **kwargs):
        if request.user.is_authenticated:
            return redirect('calendar')
        else:
            return viewFunc(request, *args, **kwargs)
    return wrappedFunc

```

Kod źródłowy 6.7 Dekorator sprawdzający autentykację użytkowników

6.2 Testy

Jednym z nieodłącznych etapów tworzenia aplikacji internetowych jest testowanie stworzonego oprogramowania. Testowanie oprogramowania to proces, który ma na celu identyfikację wszelkiego rodzaju niedociągnięć i błędów, uniemożliwiających prawidłowe działanie danego oprogramowania. Jest to przedostatni krok przed wprowadzeniem produktu na rynek. Obejmuje badanie, analizę, obserwację i ocenę jego różnych aspektów takich jak zgodność ze specyfikacją techniczną, funkcjonalność, wydajność, szybkość, przyjazność UI/UX i inne. Wszystko po to, aby oprogramowanie poprawnie działało i spełniało wszystkie zakładane cele. Testowanie jest ważne, ponieważ skutecznie zapobiega wszelkim błędom i końcowo sprowadza się do poprawienia wydajności oprogramowania. Tym samym pozytywnie wpływa na doświadczenia użytkownika końcowego [8]. Jest wiele metod testowania oprogramowania począwszy od testowania ręcznego, aż po testy automatyczne, w poniższych podrozdziałach przedstawione zostaną rodzaje testowania zastosowane przy tworzeniu tej aplikacji.

6.2.1 Testowanie manualne

Niewątpliwie jednym z podstawowych rodzajów testów jest ręczne sprawdzenie przez osobę testującą, czy dana funkcjonalność zachowuje się prawidłowo poprzez testera. Tester ma możliwość analizy oraz sprawdzenia jakości kodu, a także interakcji ze stroną bez użycia zautomatyzowanych narzędzi czy programów. Osoba testująca zachowuje się wtedy jako typowy użytkownik aplikacji i sprawdza jej możliwości poprzez użytkowanie aplikacji i interakcję z nią. Jest to podstawowa metoda testowania, lecz jest niezwykle ważna i pozwala wyeliminować dużą ilość błędów, a także sprawdzić czy aplikacja zachowuje się prawidłowo przy konkretnym użyciu. Metoda ta cechuje się niestety dużym nakładem pracy

testera, a także jest nieodporna na nieprzewidywane przez testera przypadki użycia i błędy ludzkie, ponieważ tester również może nie zauważyć części istniejących nieprawidłowości.

6.2.2 Testy jednostkowe

Django dostarcza możliwość stworzenia testów jednostkowych. Testy jednostkowe są podstawowym rodzajem testowania automatycznego. W pliku `tests.py` programista może umieścić wiele testów jednostkowych, które będą sprawdzać poprawność i funkcjonalność najmniejszych, izolowanych jednostek kodu, czyli poszczególnych funkcji, klas czy metod. Celami testów może być wiele istotnych dla działania aplikacji zagadnień i funkcjonalności takich jak na przykład:

- dostęp do bazy danych, w tym wczytywanie i zapisywanie
- działanie widoków, czy widok generuje poprawną odpowiedź html, obsługuje błędy i przetwarza dane wejściowe
- weryfikacja prawidłowego wykonywania się funkcji pomocniczych oraz usług stworzonych w aplikacji
- sprawdzenie poprawności działania formularzy pod kątem walidacji oraz obsługi danych wejściowych oraz zwracane przez nie wyniki

Stworzone testy jednostkowe pozwalają developerowi na bieżące monitorowanie poprawności i jakości kodu, co ma znaczący wpływ na rozwój i utrzymanie aplikacji.

6.3 Wdrożenie aplikacji na serwer

Aplikacja została wdrożona na platformę hostingową PythonAnywhere umożliwiając tym potencjalne korzystanie z aplikacji dla użytkowników z całego świata. PythonAnywhere to platforma hostingowa, która umożliwia łatwe wdrożenie aplikacji opartych na języku Python. Sam proces jest stosunkowo łatwy i obejmuje raptem kilka kroków takich jak:

- założenie konta na stronie hostingowej,
- uruchomienie konsoli bash na serwerze,
- wgranie wszystkich plików aplikacji na serwer na przykład za pomocą komendy `git clone 'link do repozytorium na githubie'`,
- następnie utworzenia wirtualnego środowiska komendą `mkvirtualenv --python=/usr/bin/python3.10 venv`,

- przejście za pomocą polecenia `cd` 'lokalizacja katalogu' do katalogu zawierającego `requirements.txt`
- zainstalowanie potrzebnych modułów komendą `pip install -r requirements.txt`
- ustawienie na stronie hostingowej prawidłowych ścieżek do źródła kodu, głównego katalogu aplikacji, plików statycznych i multimedialnych
- ustawienie interfejsu bramy serwera `www` (ang. WSGI - Web Server Gateway Interface)
- ponownego przeładowania aplikacji webowej na `pythonanywhere.com`

Aplikacja będzie dostępna domyślnie przez 3 miesiące na serwerze PythonAnywhere, ale jest możliwe przedłużenie jej dostępności. Adres *piotr.szumowski.bialystok.pl* został skonfigurowany w celu przekierowania do działającej aplikacji.

7. Prezentacja aplikacji

W tym rozdziale skupiono się na przedstawieniu, opisanu i wyjaśnieniu poszczególnych funkcjonalności aplikacji. Każdy podrozdział jest przeznaczony dla innego elementu i może służyć jako pewien rodzaj instrukcji obsługi.

7.1 Strona startowa z kalendarzem

Na stronie startowej znajduje się kalendarz z wszystkimi wydarzeniami (Rys. 7.1). Pod paskiem nawigacji, a powyżej kalendarza po lewej stronie znajduje się filtr wydarzeń pozwalający ograniczyć wyświetlane wydarzenia do wydarzeń tylko konkretnych usługodawców lub klientów. Po prawej stronie natomiast znajduje się przycisk umożliwiający dodanie nowego wydarzenia. Przycisk ten jest niewidoczny dla niezalogowanych użytkowników i zmienia swoje działanie oraz zawartość w zależności od roli jaką posiada zalogowany użytkownik.

Pasek nawigacyjny zawiera:

- odnośniki do tabeli z wydarzeniami oraz kalendarza,
- dzwoneczek, który pokazuje liczbę powiadomień, a po kliknięciu wyświetla ich listę
- miniaturowe zdjęcie profilowe, które po kliknięciu wyświetla listę opcji takich jak przejście do profilu, edycja profilu, wylogowanie się, a administrator dodatkowo ma możliwość przejścia do panelu administracyjnego.
- Przycisk pozwalający na zmianę motywu

Sam kalendarz ma trzy opcje widoków. Domyślny jest widok miesięczny, ale istnieje możliwość przełączenia na widok tygodniowy lub dniowy za pomocą przycisków po lewej stronie kalendarza. Nawigowanie po kalendarzu odbywa się za pomocą przycisków ze strzałkami po prawej stronie kalendarza. Jeśli użytkownik jest zalogowany to naciśnięcie na kalendarz spowoduje wyskoczenie okienka z możliwością dodania nowego terminu.

Najechnięcie na wydarzenie wyświetla odpowiedź z głównymi danymi wydarzenia takimi jak zakres trwania wydarzenia oraz linkami do profili użytkowników, którzy biorą w nim udział. Zalogowany użytkownik ma też możliwość kliknięcia na cudze wydarzenie. W zależności od roli użytkownika oraz rodzaju klikniętego wydarzenia wywołają się różne okienka pozwalające wykonanie podstawowych operacji niezbędnych do ustalania i zarządzania terminami. Zielone wydarzenia w kalendarzu to dostępne terminy usługodawców, propozycje terminów oraz rezerwacji są oznaczone kolorem żółtym,

a zatwierdzone przez obie strony wydarzenia są koloru fioletowego. W przypadku kliknięcia powiadomienia użytkownik zostanie przeniesiony do miejsca w kalendarzu, którego to powiadomienie dotyczy, a wydarzenie będzie oznaczone kolorem niebieskim z czerwoną ramką.



Rys. 7.1 Widok głównej strony z kalendarzem (Opracowanie własne)

7.2 Rejestracja i logowanie

Gdy użytkownik nie jest zalogowany to w pasku nawigacji zamiast miniaturowego przycisku wyświetlane są dwa przyciski prowadzące do widoku rejestracji i logowania. Formularz logowania (Rys. 7.2) wymaga podania nazwy użytkownika, hasła oraz kliknięcia przycisku „zaloguj się” w celu zalogowania się. Posiada również link prowadzący do panelu rejestracji poniżej przycisku. Formularz jest obsługiwany przez rozszerzenia przeglądarki takie jak menadżer haseł i pozwala na zapamiętywanie danych logowania. Formularz rejestracji (Rys. 7.3) zawiera 6 pól, które musi wypełnić użytkownik: nazwę użytkownika, imię, nazwisko, adres email, hasło oraz powtórzone hasło, przycisk do zatwierdzenia rejestracji „Zarejestruj się” oraz prowadzący do panelu logowania. Oba formularze

wyświetlają listę komunikatów błędów w przypadku podania złych danych oraz podpowiedzi w przypadku nie podania danych w wymaganej komórce.

Panel logowania do systemu rezerwacji terminów. Na górze znajduje się logo domu i tytuł "System rezerwacji terminów". W środku sekcja "Zaloguj się" zawiera dwa pola tekstowe: "Nazwa użytkownika" (z ikoną osoby) i "Hasło" (z ikoną klucza). Poniżej jest przycisk "zaloguj się" w kolorze fioletowym. Na dole link "Nie masz konta? Zarejestruj się" w kolorze fioletowym.

Rys. 7.2 Panel logowania

(Opracowanie własne)

Panel rejestracji konta do systemu rezerwacji terminów. Na górze znajduje się logo domu i tytuł "System rezerwacji terminów". W środku sekcja "Zarejestruj konto" zawiera sześć pól tekstowych: "Nazwa użytkownika" (z ikoną karty), "Imię" (z ikoną osoby), "Nazwisko" (z ikoną osoby), "Adres email" (z ikoną envelope), "Hasło" (z ikoną klucza) i "Powtórz hasło" (z ikoną klucza). Poniżej jest przycisk "Zarejestruj się" w kolorze fioletowym. Na dole link "Masz konto? Zaloguj się" w kolorze fioletowym.

Rys. 7.3 Panel rejestracji konta

(Opracowanie własne)

7.3 Profil Użytkownika

Każdy użytkownik posiada swój profil (Rys. 7.4). Profil zawiera najważniejsze dane użytkowników takie jak imię, nazwisko, email, wiek, płeć, opis profilowy oraz zdjęcie profilowe. Profile są dostępne publicznie i każdy użytkownik może zobaczyć profile innych użytkowników poprzez najechanie na wydarzenie w kalendarzu, a następnie kliknięcie w link prowadzący do profilu innego użytkownika. Użytkownicy mają możliwość edytowania swoich danych profilowych oraz zmiany nazwy użytkownika. Nazwa użytkownika nie jest nigdzie wyświetlana dla innych użytkowników i służy wyłącznie do logowania.

System rezerwacji terminów

Lista wydarzeń

KALENDARZ

5

Imie: Piotr

Nazwisko: Szumowski

Email: pizzabardzosmaczna@wp.pl

Wiek: 22

Płeć: Mężczyzna

Opis:

A, wie pan, moim zdaniem to nie ma tak, że dobrze, albo że niedobrze. Gdybym miał powiedzieć, co cenię w życiu najbardziej, powiedziałbym, że ludzi. Ludzi, którzy podali mi pomocną dłoń, kiedy sobie nie radziłem, kiedy byłem sam, to właśnie przypadkowe spotkania wpływają na nasze życie.

Edytuj dane

admin

Piotr

Szumowski

pizzabardzosmaczna@wp.pl

22

Mężczyzna

A, wie pan, moim zdaniem to nie ma tak, że dobrze, albo że niedobrze. Gdybym miał powiedzieć, co cenię w życiu najbardziej, powiedziałbym, że ludzi. Ludzi, którzy podali mi pomocną dłoń, kiedy sobie nie radziłem, kiedy byłem sam, to właśnie przypadkowe spotkania wpływają na nasze życie.

Przeglądaj...

Nie wybrano pliku.

Aktualizuj dane

Resetuj dane

Rys. 7.4 Profil z włączoną edycją (Opracowanie własne)

7.4 Tabele wydarzeń

Wszyscy użytkownicy oprócz kalendarza mają dostęp do tabel z listami wydarzeń (Rys. 7.5). Zostały stworzone dwie tabele, jedna zawiera wszystkie dostępne terminy, a druga wszystkie rezerwacje oraz propozycje terminów użytkowników. Oprócz przeglądania użytkownicy mają możliwość:

- Filtrowania tabel. Możliwe jest filtrowanie tabel po zakresie wydarzeń. Pokazywanie tylko wydarzeń w których użytkownik bierze udział oraz pokazywanie tylko terminów

46

- Wyszukiwania rekordów w tabelach po dowolnej pasującej wartości takiej jak imię i nazwisko usługodawcy lub klienta, czasie wizyty, dacie i godzinie początku lub końca wydarzenia.
- Zmiany wielkości konkretnej tabeli i ilości wierszy które pokazuje jednocześnie. Dostępne opcje to pokazywanie 10, 20, 25, 50, 100 i wszystkich wierszy.

Rys. 7.5 Tabela z wydarzeniami (Opracowanie własne)

Usługodawcy oraz administrator mogą dodawać dostępne terminy. Żeby dodać dostępny termin wystarczy, że użytkownik kliknie przycisk po prawej stronie powyżej kalendarza lub kliknie w kalendarzu na miejsce gdzie chce dodać dostępny termin. Jeśli chce stworzyć termin, który będzie trwał dłużej jak jeden dzień może użyć przycisku lub zaznaczyć na kalendarzu dni w których ma trwać wydarzenie. W każdym z tych przypadków pojawi się okienko pozwalające na sprecyzowanie godzin w których ma odbywać się wydarzenie oraz opcjonalnie użytkownik może podać przedziały czasowe ile ma trwać pojedyncze spotkanie i czas przerwy pomiędzy takimi spotkaniami. Jediną różnicą między dodawaniem terminu za pomocą przycisku a dodawaniem jednodniowego spotkania

za pomocą kalendarza jest fakt, że okno dodawania jednodniowego spotkania (Rys. 7.6) nie wymaga podawania daty, wystarczą same godziny. Natomiast okno dodawania dostępnego terminu za pomocą przycisku (Rys. 7.7) wymaga sprecyzowania zarówno dat, jak i godzin wydarzenia. Nałożona restrykcja na dostępne terminy uniemożliwia jednemu użytkownikowi dodanie dwóch dostępnych terminów nakładających się czasowo. Jest to ograniczenie mające na celu zapewnienia dostępności usługodawcy oraz zapobieganie tworzeniu nakładających się wydarzeń i kolizji terminów.

Rys. 7.6 Okienko dodawania dostępnego terminu z kalendarza (Opracowanie własne)

Rys. 7.7 Okienko dodawania dostępnego terminu z przycisku (Opracowanie własne)

7.6 Rezerwowanie i proponowanie terminu

Klienci mają możliwość zarezerwowania pożądanego terminu lub zaproponowania własnego. W przypadku proponowania pożądanego terminu sytuacja jest podobna do proponowania dostępnego terminu przez usługodawcę wystarczy, że użytkownik kliknie na kalendarz lub przycisk po prawej stronie powyżej kalendarza i poda zakres czasowy wydarzenia, jedyną różnicą jest brak możliwości podawania przedziałów czasowych pojedynczego spotkania przez klienta, zakłada się że klient proponuje pojedynczy i nie podzielny termin. Rezerwowanie terminów odbywa się natomiast poprzez kliknięcie wybranego dostępnego terminu, a po otwarciu się z okienka z wyborem (Rys. 7.8), wybrania czy chce się zarezerwować cały termin, czy tylko jego część. Rezerwowanie części terminu wygląda tak samo jak proponowanie rezerwacji, jedyną różnicą jest walidacja, która nie pozwala rezerwować terminu z poza zakresu czasowego proponowanego terminy, W przypadku gdy usługodawca podał przedział czasowe możliwe jest rezerwowanie tylko konkretnych dostępnych przedziałów (Rys. 7.9).

Usługodawca: Wioletka Wiolonczela

Termin: Czwartek, 18 Styczeń 10:00 - 12:00

Czy chcesz zaproponować rezerwację tego terminu?

☒ Zarezerwuj cały dostępny termin

☐ Zarezerwuj część dostępnego terminu

Rys. 7.8 Okienko wyboru opcji rezerwowania
(Opracowanie własne)

Usługodawca: Piotr Szumowski

Termin: Niedziela, 21 Styczeń 10:00 - 22:00

Czas wizyty: 2 godziny 45 minut

Czy chcesz zaproponować rezerwację tego terminu?

Dostępne terminy

☒ Zarezerwuj

Rys. 7.9 Rezerwowanie przedziału czasowego
(Opracowanie własne)

7.7 Potwierdzanie lub odrzucanie rezerwacji

Admin lub usługodawca będący właścicielem danego terminu mogą zatwierdzić lub usunąć propozycję rezerwacji dotyczącą spotkania z nimi (Rys. 7.11). Natomiast w przypadku gdy klient zaproponował swój własny termin usługodawca może tylko potwierdzić termin lub go nie przyjmować (Rys. 7.10). Admin natomiast może też usunąć lub edytować proponowany termin przez klienta.

Usługobiorca: Ambroży Kleks

Termin: Piątek, 12 Styczeń 11:00 - 12:00

Czy chcesz przyjąć ten proponowany termin?

☒ Tak, przyjmij ten termin

☒ Nie, nie przyjmuj tego terminu

Rys. 7.10 Potwierdzenie lub odrzucenie rezerwacji (opracowanie własne)

Usługobiorca: Ambroży Kleks

Usługodawca: Jan Kowalewski

Termin: Środa, 17 Styczeń 17:00 - 18:00

Czas wizyty: 1 godzina

Czy chcesz zatwierdzić rezerwację?

☒ Tak, zatwierdź rezerwację

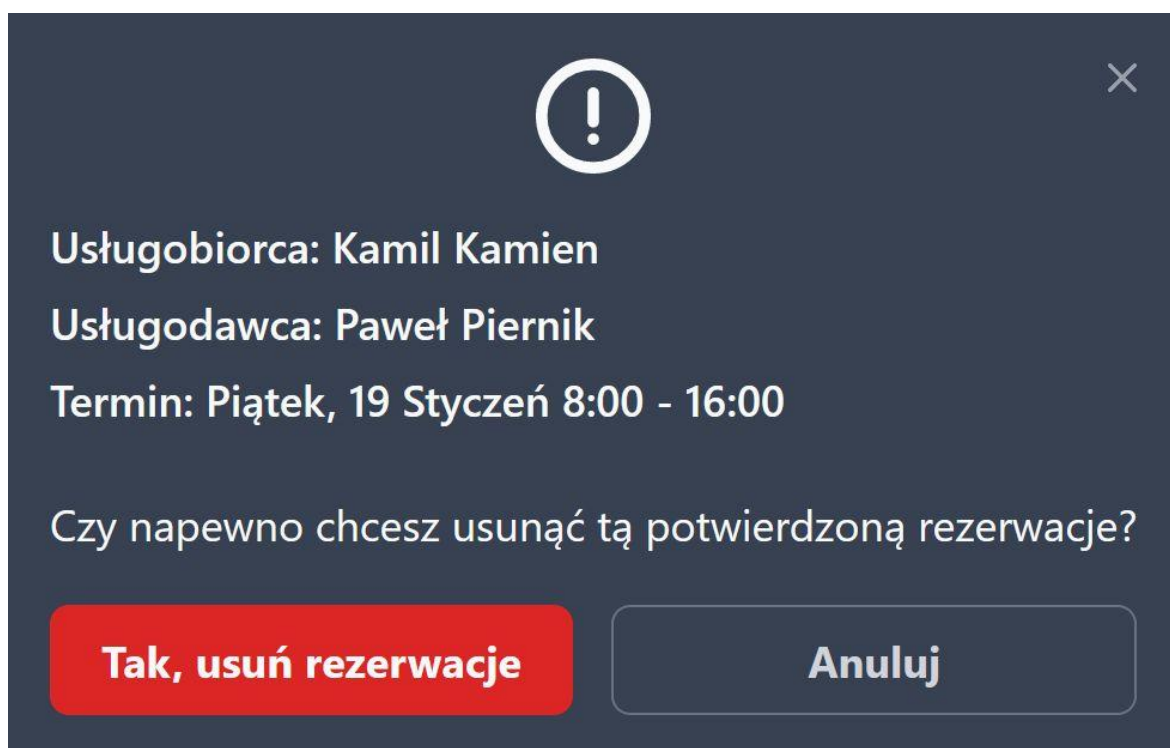
☒ Nie, usuń rezerwację

Rys. 7.11 Potwierdzenie lub usunięcie rezerwacji (opracowanie własne)

7.8 Edytowanie i usuwanie wydarzeń

Każdy użytkownik może edytować i usuwać swoje wydarzenia jeśli nie zostały potwierdzone przez obie strony i całkowicie zatwierdzone (takie wydarzenia są oznaczone

kolorem fioletowym). W przypadku potwierdzonych terminów użytkownik może tylko z niego zrezygnować lecz nie może go usunąć dla drugiej strony. Admin może edytować i usuwać terminy wszystkich użytkowników. Edytowanie i usuwanie wydarzenia odbywa się poprzez kliknięcie wydarzenia, a następnie wybrania opcji czy chcemy edytować, czy usunąć wydarzenie. Okienko bardzo podobne do tego z Rys. 7.8 tylko, że z opcjami edycji i usuwania. Wybranie opcji edytowania otwiera okienko Rys. 7.7 tylko, że z wypełnionymi aktualnymi danymi. Jeśli wybierzemy natomiast opcję usuwania otworzy się nowe okno z Rys. 7.12, operacja usuwania zakańcza się po wybraniu opcji lub zamknięciu okienka.



Rys. 7.12 Okienko usuwania (Opracowanie własne)

8. Podsumowanie

Praca zakładała stworzenie aplikacji webowej służącej do ustalania terminów spotkań. Aplikacja miała umożliwiać proponowanie terminów spotkań oraz rezerwacji przez klientów. Usługodawcy mieli mieć możliwość ogłaszania dostępnych terminów oraz zatwierdzania lub odrzucania propozycji rezerwacji. System miał umożliwiać dwustronne uzgadnianie terminów bez ingerencji pośredniczącej strony. Interfejs aplikacji powinien być przejrzysty, łatwy w obsłudze i dostępny w języku polskim. Główną zaletą aplikacji względem konkurencji miała być automatyczna walidacja terminów nie pozwalająca jednemu użytkownikowi na posiadanie kilku terminów w tym samym czasie, co chroniłoby przed występowaniem ewentualnych kolizji terminów. System miał być wdrożony na platformę hostingową umożliwiającą korzystanie z aplikacji dla szerszego grona użytkowników. Oprogramowanie zostało stworzone i wdrożone na platformie hostingowej PythonAnywhere. Adres *piotr.szumowski.bialystok.pl* został skonfigurowany w celu przekierowania do działającej aplikacji. Udało się spełnić wszystkie założenia projektowe, a działanie aplikacji jest stabilne oraz sprawne. Utworzona w ramach projektu aplikacja nadaje się do natychmiastowego użycia przez firmy oraz klientów tych firm.

Myśląc o przyszłym rozwoju aplikacji warto zastanowić się nad dodaniem funkcjonalności oferującej ustawianie spotkań z naciskiem na udział konkretnych osób lub wykluczeniem wybranych użytkowników. Możliwie przydatną funkcją byłoby również organizowanie spotkań wielostronnych, umożliwiających branie udziału w spotkaniu dla nieograniczonej liczby użytkowników. W przyszłości aplikacja mogłaby zostać urozmaicona o dodatkowe funkcjonalności pozwalające na integracje z innymi systemami przeznaczonymi do organizacji spotkań oraz rezerwacji terminów. Rozwój aplikacji o takie usługi wpłynąłby pozytywnie na uniwersalność systemu i zwiększyłby możliwości serwisu.

Bibliografia

1. Lutz M., *Learning Python, Fourth Edition*, O'Reilly Media 2009
2. DjangoProject. Witryna internetowa. <https://docs.djangoproject.com/en/5.0>, stan z 15.01.2024
3. Środowisko programistyczne Pycharm. Witryna internetowa <https://www.jetbrains.com/pycharm/features>, stan z 15.01.2024
4. Wikipedia. Wolna encyklopedia. <https://pl.wikipedia.org/wiki/Java>, stan z 15.01.2024
5. Albahari J., *C# 9.0 in a Nutshell*, O'Reilly Media 2021
6. Dąbrowski M., *Architektura warstwowa – sposób na organizację kodu*, <https://nullpointerexception.pl/architektura-warstwowa-sposob-na-organizacje-kodu>, stan z 16.01.2024
7. GeeksForGeeks. Witryna internetowa. <https://www.geeksforgeeks.org/django-project-mvt-structure>, stan z 16.01.2024
8. Skalska A., *Testowanie niezbędnym elementem tworzenia oprogramowania. Jakie są jego typy, rodzaje i poziomy?*, <https://udigroup.pl/blog/testowanie-oprogramowania-typy-rodzaje-pozioomy/#co-to-testowanie-oprogramowania>, stan z 16.01.2024

Spis rysunków

Rys. 2.1 Diagram przypadków użycia niezalogowanego użytkownika	5
Rys. 2.2 Diagram przypadków użycia klienta	7
Rys. 2.3 Diagram przypadków użycia usługodawcy	10
Rys. 2.4 Diagram przypadków użycia admina	13
Rys. 3.1 Kalendarz Google	17
Rys. 3.2 Kalendarz w aplikacji Microsoft Outlook Calendar	18
Rys. 3.3 Kalendarz w Doodle	19
Rys. 5.1 Schemat relacji bazy danych	31
Rys. 7.1 Widok głównej strony z kalendarzem	44
Rys. 7.2 Panel logowania.....	45
Rys. 7.3 Panel rejestracji konta.....	45
Rys. 7.4 Profil z włączoną edycją.....	46
Rys. 7.5 Tabela z wydarzeniami	47
Rys. 7.6 Okienko dodawania dostępnego terminu z kalendarza	48
Rys. 7.7 Okienko dodawania dostępnego terminu z przycisku	48
Rys. 7.8 Okienko wyboru opcji rezerwowania.....	49
Rys. 7.9 Rezerwowanie przedziału czasowego	49
Rys. 7.10 Potwierdzenie lub odrzucenie rezerwacji	49
Rys. 7.11 Potwierdzenie lub usunięcie rezerwacji	49
Rys. 7.12 Okienko usuwania	50

Spis kodów źródłowych

Kod źródłowy 6.1 Funkcja przesyłająca dane do backendu za pomocą AJAX	35
Kod źródłowy 6.2 Widok służący do zmiany statusu powiadomienia na odczytane	36
Kod źródłowy 6.3 Funkcja wysyłająca maile oraz klasa tworząca dodatkowy wątek	37
Kod źródłowy 6.4 Część szablonu logowania używająca formularza.....	38
Kod źródłowy 6.5 Formularz logowania	38
Kod źródłowy 6.6 Widok logowania	39
Kod źródłowy 6.7 Dekorator sprawdzający autentykację użytkowników	40