

Projekt zespołowy – etap 2

SYSTEM WSPOMAGAJĄCY DZIAŁANIE FIRMY SPRZĄTAJĄCEJ

MARCIN OSIADACZ, KACPER KOWALSKI, TOBIASZ PIWOWARCZYK, WIKTOR DMITRUK

Spis treści

1. Przedstawienie koncepcji systemu.....	2
1.1. Opis i prezentacja systemu.....	2
2. Specyfikacja funkcjonalna i нефункционална.....	7
2.1. Opis funkcji	7
2.2. Diagram hierarchii funkcji	8
2.3. Zależności poza funkcjonalne	8
2.3.1. Wydajność	8
2.3.2. RODO i pliki Cookies	8
2.4. Określenie aktorów	8
2.4.1. Użytkownik końcowy systemu - Klient.....	8
2.4.2. Pracownik	9
2.4.3. Administrator	9
2.4.4. Diagram przypadków użycia.....	10
3. Model danych.....	10
3.1. Diagram ERD.....	11
3.2. Opis poszczególnych encji oraz relacji.....	12
3.2.1 ApplicationUser	12
3.2.2 Order	13
3.2.3 OrderToServicePrice.....	13
3.2.4 OrderStatus	14
3.2.5 ServicePrice	14
4. Model architektury systemu	15
4.1. Diagram architektury systemu	15
4.2. Opis i uzasadnienie wybranego modelu.....	15
4.2.1 Warstwa danych (1).....	15
4.2.2 Warstwa serwerowa (2)	15
4.2.3 Warstwa kliencka (3).....	16

1. Przedstawienie koncepcji systemu

1.1. Opis i prezentacja systemu

Interfejs użytkownika będzie składać się z kilku podstron. To pozwoli na ułatwione korzystanie z portalu ze względu na podzielenie różnych odpowiedzialności poszczególnych zadań.

Nagłówek strony internetowej zawierać będzie listę odnośników, za pomocą których użytkownik będzie mógł nawigować po aplikacji. W zależności od tego, czy użytkownik jest zalogowany do aplikacji, to lista dostępnych odnośników będzie się różniła. W przypadku, gdy użytkownik nie jest zalogowany do systemu będzie widnieć odnośnik do zalogowania się. W przypadku, gdy użytkownik jest zalogowany – nagłówek będzie wyświetlać odnośnik do strony z panelem do zarządzania zamówieniami, a także odnośnik do wylogowania się użytkownika.



Rysunek 1 - Wygląd nagłówka w przypadku, gdy użytkownik nie jest zalogowany



Rysunek 2 - Wygląd nagłówka w przypadku, gdy użytkownik jest zalogowany

Na głównej stronie będzie widoczny podstawowy formularz, w którym klient może dokonać wstępnej analizy kosztów wykonania usługi. Formularz ten będzie zawierał pola, w których możemy ustalić liczbę pomieszczeń w budynku, miejsce, w którym dana usługa będzie dokonywana oraz rodzaj usługi. Będą widoczne również podstawowe informacje o ofercie firmy sprzątającej. Każda oferta widoczna na stronie głównej zawierać będzie nazwę usługi, krótki opis oraz szacunkową cenę za wykonanie usługi. Po kliknięciu w jedną z nich system przekieruje nas na stronę, w której możemy dokonać zamówienia danej usługi.

NAGŁÓWEK STRONY

Dokonaj podstawowej analizy kosztów

Liczba pomieszczeń

—

00

+

Miasto


▼

Rodzaj usługi

▼

Analizuj


Nasze usługi



Usługa 1

100zł


Krótki opis usługi



Usługa 2

100zł

Krótki opis usługi



Usługa 3

100zł

Krótki opis usługi


STOPKA

Rysunek 3 - Ogólny schemat strony głównej aplikacji

System udostępni możliwość zalogowania się i rejestracji do systemu. W obu przypadkach będą utworzone pola w formularzu, do których należy wprowadzić wymagane dane. Wartości w każdej komórce będą poddawane walidacji. To pozwoli nam uniknąć problemów związanych z nieprawidłowymi danymi.

Strona logowania będzie zawierać panel, w którym będą widniały dwa pola: pole tekstowe, w którym należy wprowadzić nazwę użytkownika, oraz pole tekstowe z maskowaniem znaków, które będzie wykorzystywane do wprowadzenia hasła. Na samym dole tejże strony będzie umieszczony przycisk, który posłuży do zalogowania się użytkownika. Jeżeli logowanie zakończy się powodzeniem, wówczas

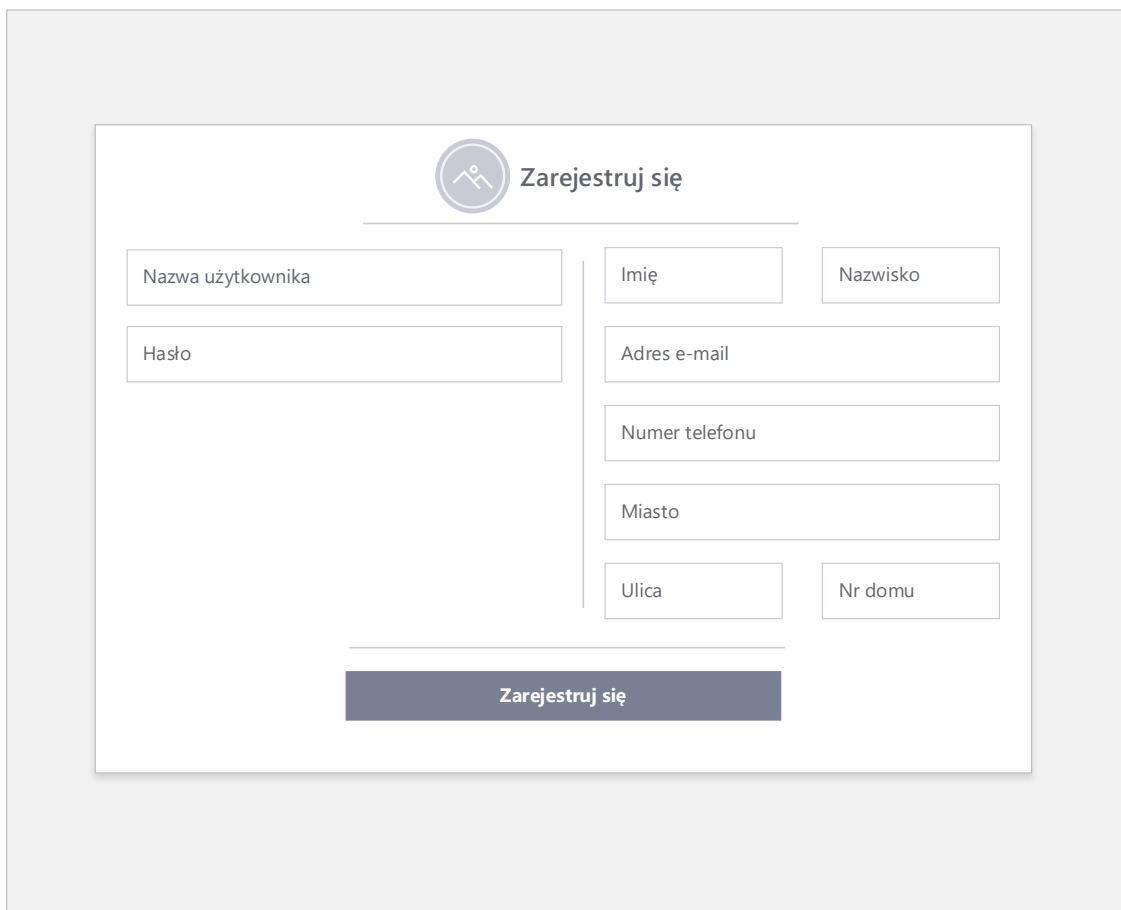
system przekieruje nas na stronę główną aplikacji. Pod danym przyciskiem umieszczony będzie tekst, który zasugeruje użytkownikowi zarejestrowanie się do aplikacji w przypadku, gdy konta jeszcze nie posiada.



The image shows a login form centered on a light gray background. The form is a white rectangle with a thin gray border. At the top left of the form is a circular icon containing a stylized person. To the right of the icon is the text "Zaloguj się". Below this header, there are two input fields: the first is labeled "Nazwa użytkownika" and the second is labeled "Hasło". Below the input fields is a dark gray button with the text "Zaloguj się" in white. At the bottom of the form, there is a line of text: "Jeżeli nie posiadasz konta, [Zarejestruj się](#)".

Rysunek 4 - Schemat przedstawiający wzór formularza logowania

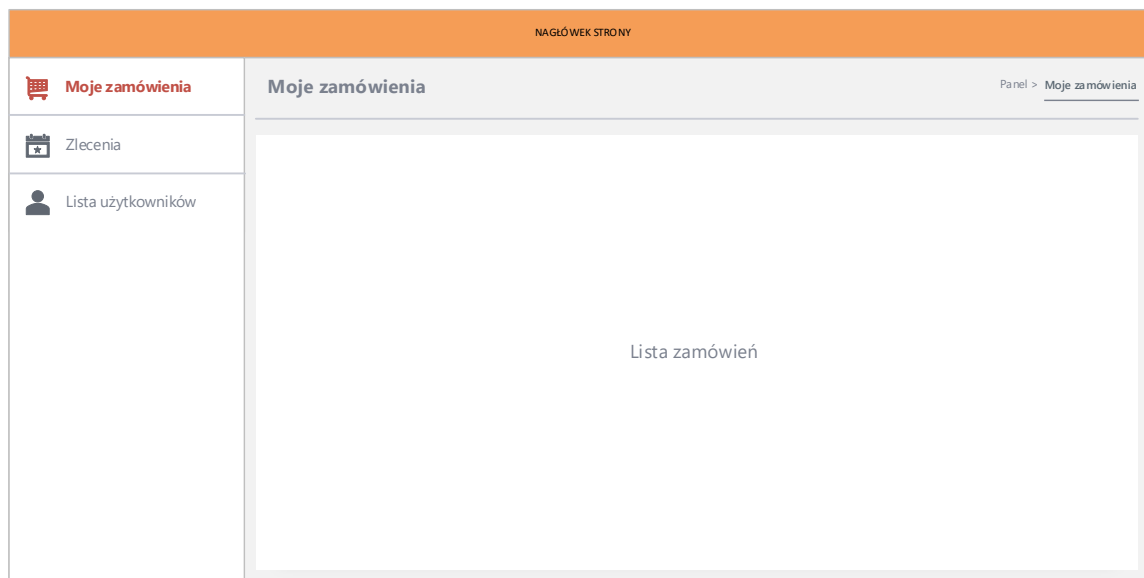
Strona rejestracji natomiast zawierać będzie panel z polami, w których należy wprowadzić następujące dane: nazwę użytkownika, hasło, imię, nazwisko, adres e-mail, numer telefonu oraz adres zamieszkania. Strona rejestracji będzie działała podobnie jak strona logowania, z tą różnicą, że będziemy mieli do dyspozycji zdecydowanie większą liczbę pól do wypełnienia. Po poprawnym zarejestrowaniu system automatycznie zaloguje użytkownika do systemu i przekieruje go na stronę główną portalu.



The image shows a registration form titled "Zarejestruj się" (Register) with a circular icon containing a person silhouette. The form is divided into two main columns. The left column contains two input fields: "Nazwa użytkownika" (Username) and "Hasło" (Password). The right column contains several input fields: "Imię" (First Name) and "Nazwisko" (Last Name) at the top, followed by "Adres e-mail", "Numer telefonu" (Phone Number), "Miasto" (City), "Ulica" (Street), and "Nr domu" (House Number). At the bottom of the form is a dark blue button labeled "Zarejestruj się".

Rysunek 5 - Schemat przedstawiający wzór formularza do rejestracji

Po zalogowaniu się (lub rejestracji) użytkownika użytkownik będzie mógł dokonać zamówienia. Po dokonaniu zamówienia użytkownik będzie mógł przeglądać swoje zamówienia w panelu klienta. Dostęp do panelu będzie możliwy po zalogowaniu się użytkownika. Panel ten będzie dostępny dla każdego zalogowanego użytkownika. W zależności od tego, jaką rolę posiada użytkownik, niektóre opcje nie będą w nim widoczne.



Rysunek 6 - Schemat przedstawiający ogólny schemat wyglądu panelu użytkownika

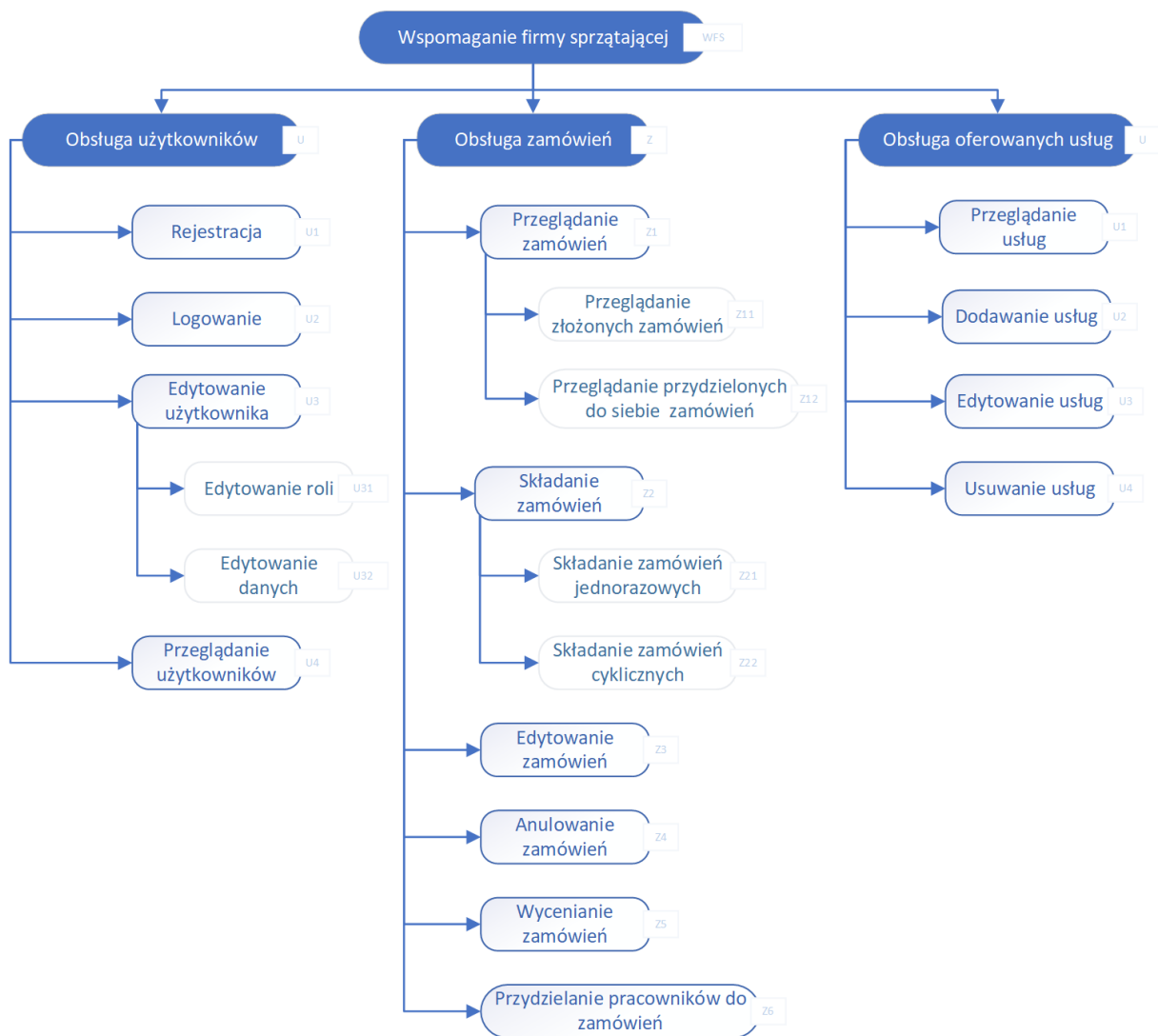
2. Specyfikacja funkcjonalna i нефункционаlna

Rolą aplikacji jest wspomaganie działania firmy sprzątającej, tj. zarządzanie zamówieniami, klientami i pracownikami. Użytkownicy powinni mieć możliwość samodzielnego składania zamówień, a aplikacja powinna umożliwiać wygodne wycenianie usług.

2.1. Opis funkcji

Lp.	Opis wymagania funkcjonalnego
1	Użytkownik może zarejestrować nowe konto jako Klient
2	Użytkownik może zalogować się na swoje konto
3	Niezalogowany użytkownik może przeglądać ofertę
4	Niezalogowany użytkownik może samodzielnie dokonać wstępnej wyceny usługi
5	Klient może wszystko to, co niezalogowany użytkownik
6	Klient może złożyć zamówienie na usługę jednorazową
7	Klient może złożyć zamówienie na usługę cykliczną
8	Klient może przeglądać swoje zamówienia
9	Klient może anulować złożone przez siebie zamówienie
10	Administrator może wszystko to, co użytkownik w roli Klient
11	Administrator może przeglądać bazę Klientów
12	Administrator może dodawać konta Pracowników
13	Administrator może usuwać konta Pracowników
14	Administrator może przeglądać zamówienia złożone przez Klientów
15	Administrator może przydzielić Pracownika do zamówienia, co jest równoznaczne z potwierdzeniem zamówienia
16	Administrator może przeglądać oferowane usługi
17	Administrator może edytować oferowane usługi i ich ceny
18	Administrator może dodawać nowe usługi
19	Administrator może usuwać usługi z oferty
20	Pracownik może wszystko to, co użytkownik w roli Klient
21	Pracownik może przeglądać przydzielone do siebie zamówienia na sprząatanie
22	Pracownik może oznaczyć przydzielone do siebie zamówienie jako wykonane

2.2. Diagram hierarchii funkcji



2.3. Zależności poza funkcjonalne

2.3.1. Wydajność

- Czas odpowiedzi serwera na żądanie http użytkownika nie powinien przekraczać 500ms.

2.3.2. RODO i pliki Cookies

- System powinien wyświetlać informację o przechowywaniu plików Cookies na urządzeniu użytkownika,
- System powinien wymagać akceptacji regulaminu i polityki prywatności podczas rejestracji nowego konta użytkownika.

2.4. Określenie aktorów

2.4.1. Użytkownik końcowy systemu - Klient

- Ma możliwość rejestracji konta i logowania na konto
- Bez logowania się do systemu może:

- Przeglądać ofertę,
- Dokonać wstępnej wyceny usługi.

3. Po zalogowaniu może dodatkowo:

- Złożyć zamówienie na jednorazowe sprzątanie,
- Złożyć zamówienie na cykliczne sprzątanie,
- Przeglądać złożone przez siebie zamówienia,
- Anulować złożone przez siebie zamówienie.

2.4.2. Pracownik

1. Może robić to, co użytkownik końcowy systemu (Klient)

2. Ponadto Pracownik może:

- Przeglądać przydzielone do siebie zamówienia na sprzątanie,
- Oznaczyć przydzielone do siebie zamówienie jako wykonane.

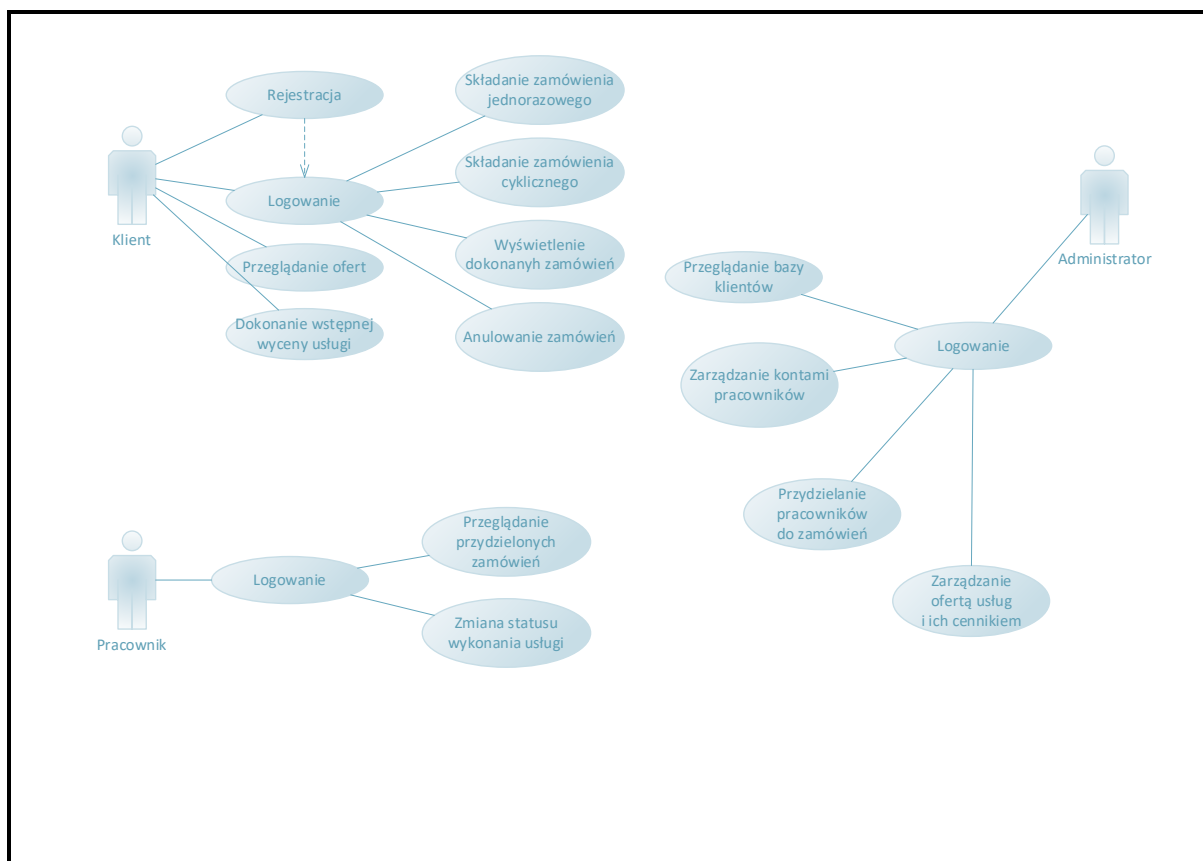
2.4.3. Administrator

1. Może robić to, co użytkownik końcowy systemu (Klient)

2. Ponadto Administrator może:

- Przeglądać bazę Klientów,
- Dodawać i usuwać konta Pracowników,
- Przeglądać wszystkie zamówienia,
- Przydzielać Pracowników do poszczególnych zamówień,
- Zarządzać ofertą usług i ich cennikiem.

2.4.4. Diagram przypadków użycia



Rysunek 7 Diagram przypadków użycia

3. Model danych

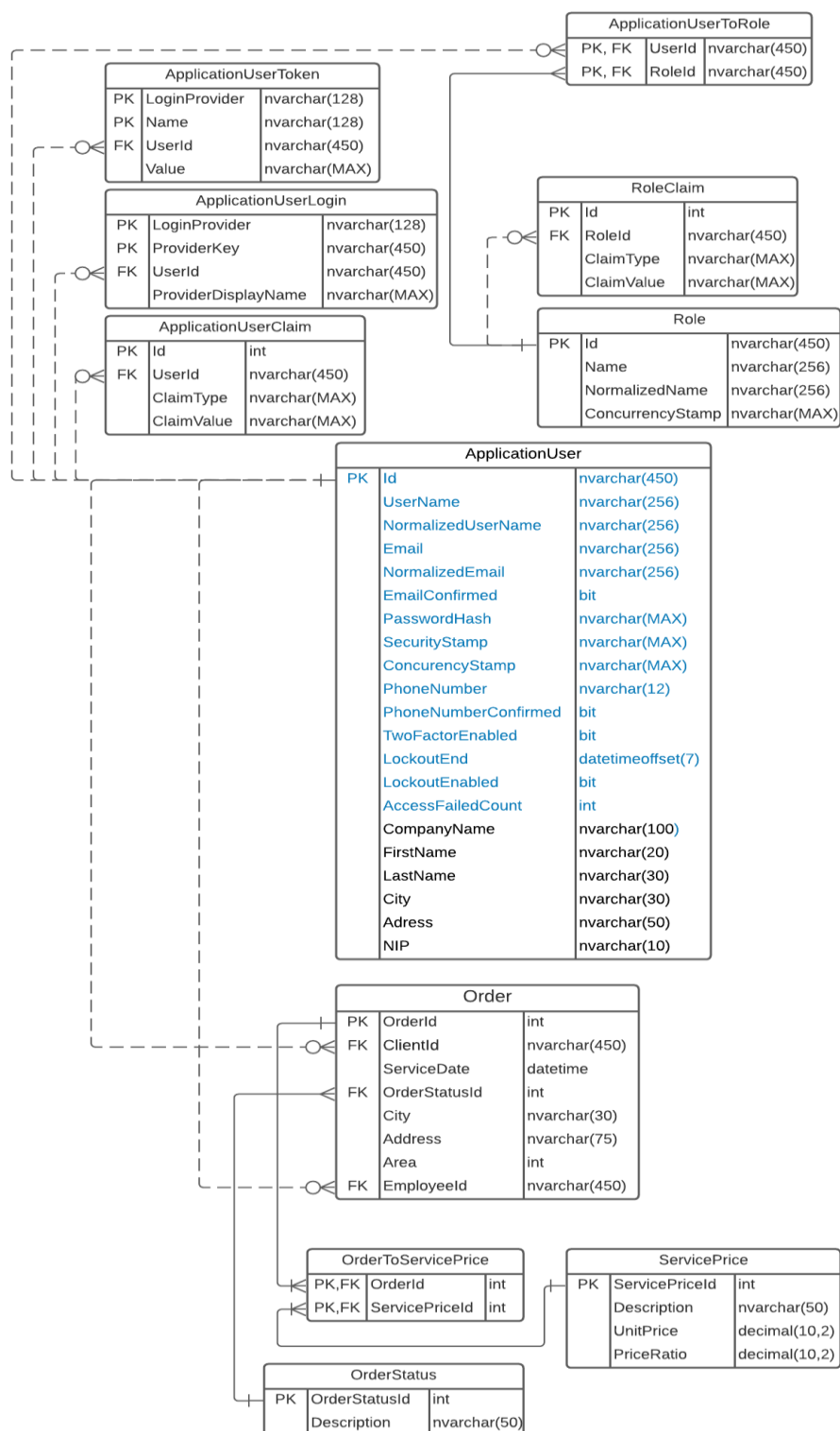
Na potrzeby realizacji projektu informacje przechowywane będą w jedenastu tabelach bazy danych. W pierwszych siedmiu tabelach będą znajdować się dane systemowe konieczne do zarządzania kontami użytkowników - w szczególności ich hasłami, rolami, danymi profilu, poświadczeniami oraz posiadanymi tokenami. Funkcjonalność ta zrealizowana zostanie poprzez implementację interfejsu API z biblioteki Identity frameworka ASP.NET Core. Dane główne, generowane w trakcie funkcjonowania systemu zbierane będą w pięciu tabelach, z których jedna (ApplicationUser) będzie wspólna zarówno dla klientów jak i użytkowników serwisu. W tej encji pola oznaczone kolorem niebieskim są polami dziedziczonymi encji ASPNetUsers.

Na model danych składają się następujące encje:

- ApplicationUserToken, ApplicationUserLogin, ApplicationUserClaim, ApplicationUserToRole, RoleClaim, Role** - dane systemowe, niezbędne do procesu rejestracji i logowania użytkowników;
- ApplicationUser** - klienci oraz użytkownicy systemu;
- Order** - zamówienia klientów;
- OrderStatus** - słownik statusu zamówień;
- ServicePrice** - słownik cennika usług;
- OrderToServicePrice** - usługi z cennika wybrane przez klienta do realizacji zamówienia.

3.1. Diagram ERD

Graficzną reprezentację zaimplementowanego modelu danych przedstawia poniższy diagram ERD (*Entity Relationship Diagram*) wykonany w notacji Martina, zwaną również notacją “kurzych łapek”.



Rysunek 8. Diagram ERD

3.2. Opis poszczególnych encji oraz relacji

Encje pochodzące z pakietu ASP.NET Core Identity wymienione w punkcie 3 a są niezmodyfikowane i charakteryzują się standardowymi, predefiniowanymi powiązaniami. Jedyną zmianą jest dostosowanie nazw do pozostałych encji modelu. Reasumując kwestie związane z zarządzaniem użytkownikiem należy wspomnieć, że encję ApplicationUser z encjami ApplicationUserClaim, ApplicationUserLogin i ApplicationUserToken wiąże opcjonalna relacja jeden do wielu (1:N). Ta sama encja tj. ApplicationUser wiązana jest z encją Role relacją wiele do wielu (N:N), gdzie encją pośredniczącą jest encja ApplicationUserToRole. Ponadto, każda z encji Role może charakteryzować się wieloma skojarzonymi encjami RoleClaim (1:N).

Istotniejszymi z punktu widzenia są pozostałe encje gromadzące dane użytkowników. W poniższych tabelach znajdują się definicje tych encji.

3.2.1 ApplicationUser

Encja z danymi klientów i pracowników.

Klucz podstawowy: Id

Klucze obce: brak

Relacje: opisane powyżej i poniżej

Pola	Typ danych	Czy puste	Opis
(..) pola odziedziczone z encji AspNetUser			
CompanyName	nvarchar(450)	TAK	Nazwa firmy, jeżeli jest to użytkownik firmowy
FirstName	nvarchar(20)	TAK	Imię, jeżeli jest to użytkownik prywatny
LastName	nvarchar(30)	TAK	Nazwisko, jeżeli jest to użytkownik prywatny
City	nvarchar(30)	TAK	Miasto pochodzenia
Address	nvarchar(50)	TAK	Adres użytkownika
NIP	nvarchar(10)	TAK	NIP firmy, jeżeli jest to użytkownik firmowy

3.2.2 Order

Encja z zamówieniami klientów

Klucz podstawowy: OrderId

Klucze obce: ClientId->ApplicationUser(Id),
EmployeeId->Applicationuser(Id)
OrderStatusId->OrderStatus(Id)

Relacje:

- **ApplicationUser (Id) – Order (ClientId) , ApplicationUser(Id) – Order (EmployeeId)** obie typu jeden do wielu (1:N), obie opcjonalne, gdyż nie każdy użytkownik musi mieć zleconą usługę sprzątnia i nie każdy pracownik musi mieć przypisane zlecenie sprzątnia.
- **Order(OrderStatusId) - OrderStatus(Id)** typu 1:N wiążąca zlecenie z jej statusem. Relacja ta jest jednoznaczna tzn, każde zlecenie musi mieć zdefiniowany status.

Pola	Typ danych	Czy puste	Opis
OrderId	int	NIE	Nazwa firmy, jeżeli jest to użytkownik firmowy
ClientId	int	NIE	Imię, jeżeli jest to użytkownik prywatny
ServiceDate	datetime	NIE	Nazwisko, jeżeli jest to użytkownik prywatny
OrderStatusId	int	NIE	Status zlecenia
City	nvarchar(30)	NIE	Miasto, wykonania usługi
Address	nvarchar(50)	NIE	Adres, lokalu do wykonania usługi
Area	int	NIE	Powierzchnia lokalu przeznaczonego do usługi
EmployeeId	int	TAK	Identyfikator pracownika przypisanego do zlecenia

3.2.3 OrderToServicePrice

Encja pośrednicząca pomiędzy zamówieniem a przedmiotem zamówienia

Klucz podstawowy: sprzężony z pól: OrderId i ServicePricId

Klucze obce: OrderId->Order (Id),
ServicePricId->ServicePrice(Id)

Relacje:

- **Order (Id) – OrderToServicePrice (OrderId) , ServicePrice(Id) – OrderToServicePrice (ServicePricId)** obie typu jeden do wielu (1:N), jednoznaczne, gdyż nie ma zlecenia bez wskazania jego zakresu. Encja ta jest encją pośredniczącą pomiędzy encjami Order i ServicePrice, które łączy relacja wiele do wielu (N:N).

Pola	Typ danych	Czy puste	Opis
OrderId	int	NIE	Identyfikator zlecenia
ServicePriceId	int	NIE	Identyfikator usługi słownikowej

3.2.4 OrderStatus

Encja słownikowa z możliwymi statusami zamówienia

Klucz podstawowy: OrderStatusId

Klucze obce: brak

Relacje: powyżej

Pola	Typ danych	Czy puste	Opis
OrderId	int	NIE	Identyfikator zlecenia
ServicePriceId	int	NIE	Identyfikator usługi słownikowej

3.2.5 ServicePrice

Encja słownikowa z cennikiem usług

Klucz podstawowy: ServicePriceId

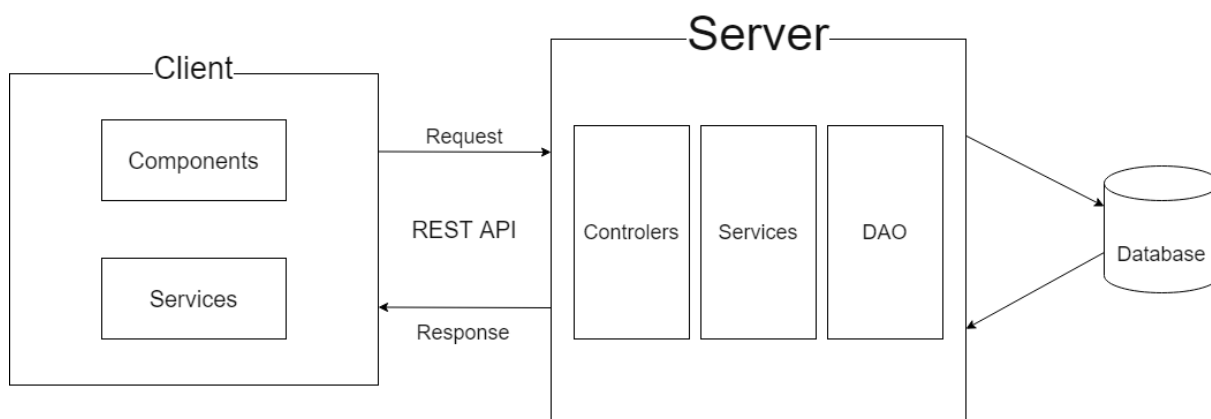
Klucze obce: brak

Relacje: powyżej

Pola	Typ danych	Czy puste	Opis
ServicePriceId	int	NIE	Identyfikator pozycji słownika
DescriptionPrice	nvarchar(50)	NIE	Nazwa usługi
UnitPrice	decimal(10,2)	NIE	Koszt jednostkowy usługi
PriceRatio	decimal(10,2)	NIE	Współczynnik wiążący cenę usługi z metrażem lokalu do sprzątnia

4. Model architektury systemu

4.1. Diagram architektury systemu



Rysunek 8 Diagram architektury systemu

4.2. Opis i uzasadnienie wybranego modelu

Aplikacja budowana jest w architekturze wielowarstwowej. Rozróżnionymi warstwami są:

1. Warstwa danych
2. Warstwa serwerowa
3. Warstwa kliencka

4.2.1 Warstwa danych (1)

Jest odpowiedzialna za przetwarzanie danych bezpośrednio w bazie danych. Komunikuje się z warstwą serwerową, w celu dodania nowych rekordów lub modyfikacji istniejących danych.

4.2.2 Warstwa serwerowa (2)

Jest to warstwa pośrednia pomiędzy warstwą danych (1) a warstwą kliencką (3). Odpowiada za przyjmowanie żądań od klienta (3) i po odpowiednim przetworzeniu, przekazywanie ich do bazy danych (3).

Warstwa serwerowa składa się z trzech głównych elementów:

1. DAO
2. Serwisy
3. Kontrolery

DAO – z punktu widzenia kodu jest to po prostu komponent dostarczający jednolity interfejs pozwalający na komunikację pomiędzy aplikacją a bazą danych. Zawiera w sobie logikę dostępu do bazy danych i mapowania danych na obiekty.

Serwisy – Są to klasy korzystające z DAO, a także innych serwisów w tej warstwie, w celu obsługiwanie logiki aplikacji. Stanowią wyższy poziom abstrakcji niż DAO i łączą w całość operacje z niższych warstw logicznych.

Kontrolery – Odpowiadają głównie za transfer danych. Dokonują odebrania danych od użytkownika, a następnie ich walidacji, po to, by przekazać je do serwisów.

Technologią wybraną do stworzenia tej warstwy jest ASP.NET Core i biblioteka Identity będąca częścią tego frameworku.

Architektura ASP.NET Core różni się znacząco od poprzednich wersji ASP.NET. Jedną z jej głównych zalet jest znacznie większa testowalność aplikacji zbudowanych na tym systemie. Wspiera również szerszą selekcję systemów operacyjnych (Windows, macOS i Linux) i jest po prostu szybsza od poprzedników.

Pakiet ASP.NET Core Identity zapewnia API pozwalające na bezproblemową obsługę logowania i rejestracji. Zarządza użytkownikami, hasłami, rolami i nie tylko. Pozwala również na łatwą integrację z zewnętrznymi dostawcami usług logowania, takimi jak: Google, Facebook czy Twitter.

4.2.3 Warstwa kliencka (3)

Technologią wybraną do zbudowania tej warstwy aplikacji jest React JS. Dzięki architekturze aplikacji jednostronicowej pozwala on nam uniknąć niepotrzebnego przeładowywania aplikacji, a szybkość systemu znacznie wzrasta dzięki wirtualnemu drzewu DOM będącemu jedną z kluczowych charakterystyk Reacta.

Kolejną zaletą tej biblioteki jest jednokierunkowy przepływ danych zwiększający stabilność aplikacji. Pozwala nam być pewnymi, że stan naszych komponentów nie ulegnie zmianie bez naszego bezpośredniego polecenia.

Poza komponentami napisanymi w React warstwa kliencka (3) składa się również z serwisów umożliwiających komunikację z warstwą serwerową (2) przez REST API. Do łatwiejszej obsługi żądań między warstwami wykorzystana jest biblioteka Axios.

Następną częścią części klienckiej (3) stanowi TypeScript. Ten nadzbiór języka JavaScript umożliwia łatwą integrację z funkcyjnymi komponentami Reacta i pozwala na uniknięcie wielu niepotrzebnych błędów pojawiających się podczas tworzenia aplikacji.

Ostatnimi elementami tej warstwy są biblioteki Jest i React Testing Library. Pozwalają one na łatwe tworzenie testów jednostkowych i integracyjnych do utworzonych komponentów. Głównymi zaletami tych bibliotek jest bardzo szeroka oferta opcji umożliwiających zachowanie użytkownika podczas interakcji z aplikacją.