2021

Projekt zespołowy – etap 2

SYSTEM WSPOMAGAJĄCY DZIAŁANIE FIRMY SPRZĄTAJĄCEJ

MARCIN OSIADACZ, KACPER KOWALSKI, TOBIASZ PIWOWARCZYK, WIKTOR DMITRUK

Spis treści

1. Przedstawienie koncepcji systemu

Główną rolą budowanego systemu jest wspomaganie działania firmy sprzątającej. Będzie on umożliwiał między innymi zarządzanie bazą klientów, zarządzanie zamówieniami i zleceniami pracy oraz wygodne wycenianie usług.

Podstawowe zadania realizowane przez aplikację:

- Zarządzanie klientami indywidualnymi oraz biznesowymi,
- Zarządzanie zamówieniami,
- Tworzenie zarówno jednorazowych jak i cyklicznych zleceń pracy,
- Zarządzanie jednoosobowymi zespołami sprzątającymi,
- Wygodne wycenianie z wykorzystaniem cennika.

System znacznie uprości realizację i obsługę zamówień w firmie, zarówno z perspektywy pracowników jak i kadry kierowniczej. Scentralizowana koordynacja zadań usprawni przydzielanie zleceń oraz zarządzanie czasem. W trakcie dokonywania zamówienia całkowity koszt za wykonanie usługi będzie wyznaczany w czasie rzeczywistym bez konieczności oczekiwania na odpowiedź ze strony serwera, co zapewni sprawne i wygodne złożenie nowego zlecenia. Wszelkie zamówienia będą udostępnione w postaci listy, co zapewni użytkownikowi łatwy dostęp do historii dokonanych zleceń. Możliwość tworzenia cyklicznych zleceń pracy znacznie przyspieszy procesy biznesowe w firmie, a funkcja wygodnego wyceniania z wykorzystaniem cennika pozytywnie wpłynie na szybkość i jakość obsługi klienta.

2. Specyfikacja funkcjonalna i niefunkcjonalna

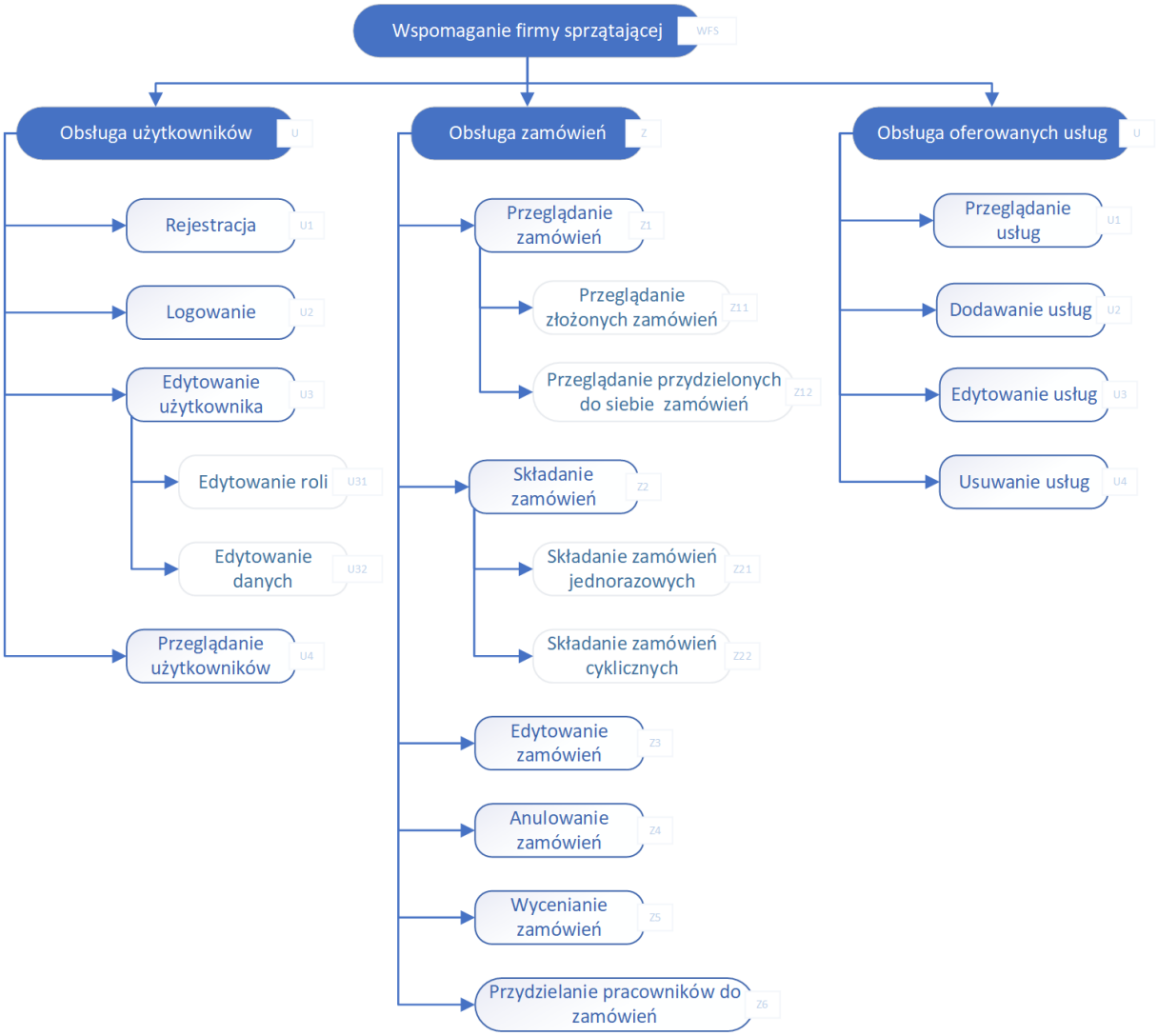
Rolą aplikacji jest wspomaganie działania firmy sprzątającej, tj. zarządzanie zamówieniami, klientami i pracownikami. Użytkownicy powinni mieć możliwość samodzielnego składania zamówień, a aplikacja powinna umożliwiać wygodne wycenianie usług.

2.1. Opis funkcji

Lp.	Opis wymagania funkcjonalnego
1	Niezalogowany użytkownik może zarejestrować nowe konto jako Klient
2	Niezalogowany 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

Lp.	Opis wymagania funkcjonalnego
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 sprzątanie
22	Pracownik może oznaczyć przydzielone do siebie zamówienie jako wykonane

2.2. Diagram hierarchii funkcji



Rysunek 1 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. Niezalogowany użytkownik – Gość

- Ma możliwość rejestracji konta i logowania na konto,
- Przeglądać ofertę,
- Dokonać wstępnej wyceny usługi.

2.4.2. Użytkownik końcowy systemu - Klient

1. Może robić to, co użytkownik niezalogowany (Gość)

2. Po zalogowaniu jako Klient może dodatkowo:

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

2.4.3. 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.

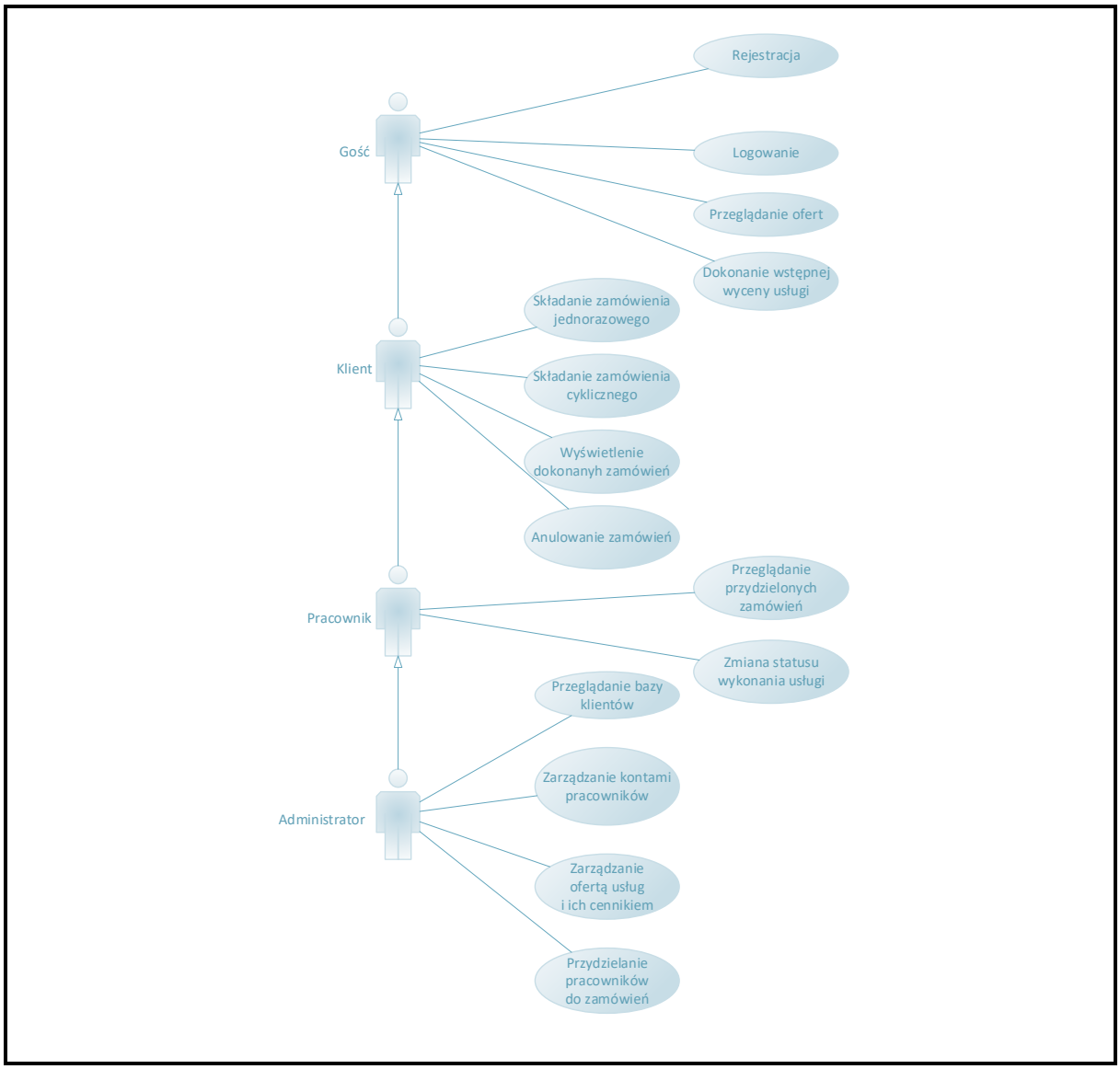
2.4.4. Administrator

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.

2.4.5. Diagramy przypadków użycia



Rysunek 2 Diagram przypadków użycia

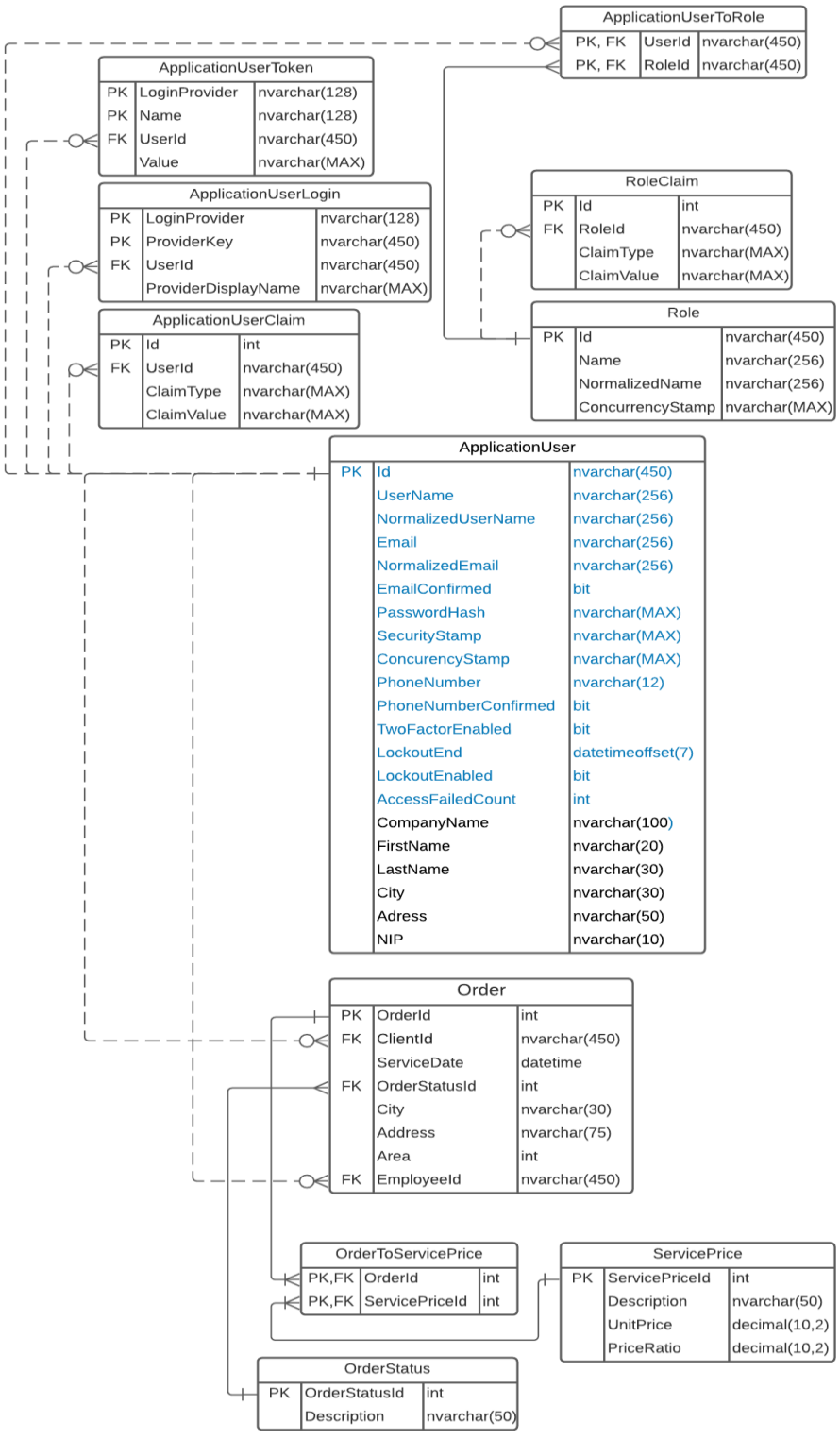
3. Model danych

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:

a. **AplicationUserToken, AplicationUserLogin, AplicationUserClaim, AplicationUserToRole, RoleClaim, Role -** dane systemowe, niezbędne do procesu rejestracji i logowania użytkowników;
b. **ApplicationUser** - klienci oraz użytkownicy systemu;
c. **Order** - zamówienia klientów;
d. **OrderStatus** - słownik statusu zamówień;
e. **ServicePrice** - słownik cennika usług;
f. **OrderToServicePrice -** usługi z cennika wybrane przez klienta do realizacji zamówienia.

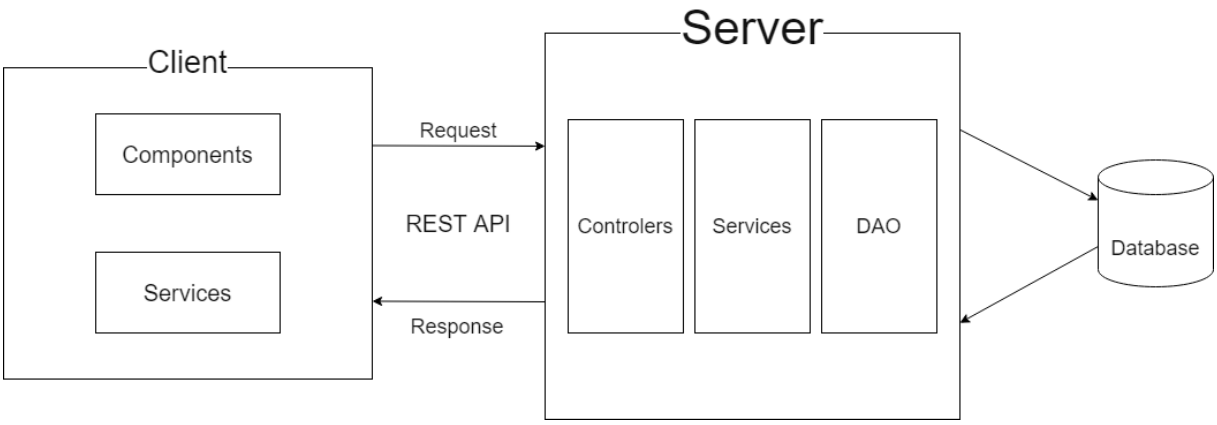
Graficzna reprezentacja zaimplementowanego modelu danych przedstawia poniższy diagram ERD (*Entity Relationship Diagram)* wykonany w notacji Martina, zwaną również notacją “kurzych łapek”.



Rysunek 3 Diagram ERD

4. Model architektury systemu

4.1. Diagram architektury systemu



Rysunek 4 Diagram architektury systemu

4.2. Opis i uzasadnienie wybranego modelu

Aplikacja budowana jest w architekturze wielowarstwowej. Rozróżnionymi warstwami są:

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

1 - Warstwa danych

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

2 - Warstwa serwerowa

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 obsługiwania 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.

3 - Warstwa kliencka

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 integracją 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.