1. Opis projektu - założenia i funkcje

Aplikacja ma za zadanie ułatwienie pracy recepcjonistom pracującym w małych salonach branży beauty ze szczególnym uwzględnieniem branży kosmetologicznej. Jej funkcje obejmować będą:

- dodawanie klientów do bazy,
- edycję danych klientów,
- usuwanie klientów z bazy,
- wprowadzanie zabiegów na listę,
- edycję danych zabiegów,
- usuwanie zapisanych zabiegów,
- rejestrację wizyty klientów w salonie,
- możliwość zmiany danych wizyty,
- usuwanie wizyty z kalendarza.

2. Analiza systemowa

a) przypadki użycia

Przypadek I

Identyfikator: I

Nazwa: Dodanie klienta do bazy.

Opis: Zapewnienie recepcjoniście możliwości dodania klienta do bazy.

Stan początkowy: Klienta nie ma w bazie danych.

Stan końcowy: Klient jest zarejestrowany w bazie klientów.

Aktorzy pierwszoplanowi: Recepcja

Aktorzy drugoplanowi: Klient

Podstawowy przebieg zdarzeń:

1. Przypadek użycia zaczyna się, kiedy klient zgłasza się do recepcji.
2. Recepcjonista uruchamia aplikację.
3. Recepcjonista klika “ustawienia”.
4. Recepcjonista wpisuje kod dostępu do bazy danych i klika “ok”.
5. Recepcjonista klika przycisk “klienci”.
6. Recepcjonista klika przycisk “nowy klient”.
7. Recepcjonista wprowadza dane klienta - imię, nazwisko, numer telefonu.
8. Recepcjonista zatwierdza dane klienta klikając przycisk “zapisz”.
9. Przypadek użycia kończy się w chwili dodania klienta do bazy.

Alternatywny przebieg zdarzeń: W trakcie wpisywania danych przez recepcjonistę klient rezygnuje z dodania go do bazy.

A8. Recepcjonista nie wprowadza klienta do bazy klikając przycisk “powrót”.

A9. Przypadek użycia kończy się brakiem wprowadzenia nowego klienta do bazy.

Przypadek II
Identyfikator: II
Nazwa: Edycja danych klienta.
Opis: Zapewnienie recepcjoniście możliwości edycji danych klienta.
Stan początkowy: Dane klienta w bazie klientów są nieaktualne.
Stan końcowy: Dane klienta są aktualne.
Aktorzy pierwszoplanowi: Recepcjonista.
Aktorzy drugoplanowi: Klient.
Podstawowy przebieg zdarzeń:
1. Przypadek użycia zaczyna się w chwili zgłoszenia nieaktualnych danych przez klienta.
2. Recepcjonista uruchamia aplikację.
3. Recepcjonista klika “ustawienia”.
4. Recepcjonista wpisuje kod dostępu do bazy danych i klika “ok”.
5. Recepcjonista klika przycisk “klienci”.
6. Recepcjonista wyszukuje klienta z bazy.
7. Recepcjonista klika w miejscu danych wybranego klienta.
8. Recepcjonista zmienia dane klienta.
9. Recepcjonista zatwierdza nowe dane klienta klikając przycisk “zapisz”.
10. Przypadek użycia kończy się w chwili wyświetlania edytowanych danych klienta.

Alternatywny przebieg zdarzeń: W trakcie wpisywania edytowanych danych przez recepcjonistę klient rezygnuje z wprowadzenia zmian.
A8. Recepcjonista nie zmienia danych klienta w bazie klikając przycisk “powrót”.
A9. Przypadek użycia kończy się brakiem edycji danych klienta.

Przypadek III
Identyfikator: III
Nazwa: Usunięcie klienta z bazy.
Opis: Zapewnienie recepcjoniście możliwości usunięcia danych klienta z bazy.
Stan początkowy: Dane klienta są zapisane w bazie.
Stan końcowy: Brak klienta w bazie.
Aktorzy pierwszoplanowi: Recepcjonista.
Aktorzy drugoplanowi: Klient.
Podstawowy przebieg zdarzeń:
1. Przypadek użycia zaczyna się w chwili zgłoszenia przez klienta żądania usunięcia jego danych z bazy.

2. Recepcjonista uruchamia aplikację.
3. Recepcjonista klika “ustawienia”.
4. Recepcjonista wpisuje kod dostępu do bazy danych i klika “ok”.
5. Recepcjonista klika przycisk “klienci”.
6. Recepcjonista wyszukuje klienta z bazy.
7. Recepcjonista klika w miejscu danych wybranego klienta.
8. Recepcjonista klika przycisk “usuń”.
9. Przypadek użycia kończy się w chwili usunięcia klienta z bazy.

Alternatywny przebieg zdarzeń: W trakcie usuwania danych przez recepcjonistę klient chce dalej figurować na liście klientów.
A8. Recepcjonista nie usuwa danych klienta z bazy klikając przycisk “powrót”.
A9. Przypadek użycia kończy się dalszą obecnością danych klienta w bazie.

Przypadek IV
Identyfikator: IV
Nazwa: Dodanie zabiegu do oferty.
Opis: Zapewnienie recepcjoniście możliwości dodania zabiegu do oferty.
Stan początkowy: Brak określonego zabiegu w bazie.
Stan końcowy: Zabieg figuruje w bazie.
Aktorzy pierwszoplanowi: Recepcjonista
Aktorzy drugoplanowi: -
Podstawowy przebieg zdarzeń:
1. Przypadek użycia zaczyna się, kiedy trzeba dodać nowy zabieg do bazy.
2. Recepcjonista uruchamia aplikację.
3. Recepcjonista klika “ustawienia”.
4. Recepcjonista wpisuje kod dostępu do bazy danych i klika “ok”.
5. Recepcjonista klika przycisk “zabiegi”.
6. Recepcjonista klika przycisk “nowy zabieg”.
7. Recepcjonista wprowadza dane zabiegu - nazwę i czasochłonność, opcjonalnie może wprowadzić opis.
8. Recepcjonista zatwierdza dane zabiegu klikając przycisk “zapisz”.
9. Przypadek użycia kończy się w chwili dodania zabiegu do bazy.

Alternatywny przebieg zdarzeń: W trakcie wpisywania nowego zabiegu recepcjonista rezygnuje z dodania go do bazy.
A7. Recepcjonista nie wprowadza nowego zabiegu do bazy klikając przycisk “powrót”.
A8. Przypadek użycia kończy się brakiem dodania nowego zabiegu.

Przypadek V
Identyfikator: V
Nazwa: Edycja danych zabiegu.
Opis: Zapewnienie recepcjoniście możliwości edycji danych zabiegu.
Stan początkowy: Dane zabiegu w bazie są nieaktualne.
Stan końcowy: Dane zabiegu są aktualne.
Aktorzy pierwszoplanowi: Recepcjonista.
Aktorzy drugoplanowi: -
Podstawowy przebieg zdarzeń:
1. Przypadek użycia zaczyna się w chwili potrzeby zmiany nieaktualnych danych zabiegu.
2. Recepcjonista uruchamia aplikację.
3. Recepcjonista klika “ustawienia”.
4. Recepcjonista wpisuje kod dostępu do bazy danych i klika “ok”.
5. Recepcjonista klika przycisk “zabiegi”.
6. Recepcjonista wyszukuje zabieg z bazy.
7. Recepcjonista klika w miejscu danych wybranego zabiegu.
8. Recepcjonista zmienia dane zabiegu.
9. Recepcjonista zatwierdza nowe dane zabiegu klikając przycisk “zapisz”.
10. Przypadek użycia kończy się w chwili wyświetlania edytowanych danych zabiegu.

Alternatywny przebieg zdarzeń: Recepcjonista rezygnuje z edycji danych zabiegu.
A9. Recepcjonista nie zmienia danych zabiegu w bazie klikając przycisk “powrót”.
A10. Przypadek użycia kończy się brakiem edycji danych zabiegu.

Przypadek VI
Identyfikator VI
Nazwa: Usunięcie zabiegu z bazy.
Opis: Zapewnienie recepcjoniście możliwości usunięcia danych zabiegu z oferty.
Stan początkowy: Dane zabiegu są zapisane w bazie.
Stan końcowy: Brak zabiegu w bazie.
Aktorzy pierwszoplanowi: Recepcjonista.
Aktorzy drugoplanowi: -
Podstawowy przebieg zdarzeń:
1. Przypadek użycia zaczyna się w chwili potrzeby usunięcia określonego zabiegu z bazy.
2. Recepcjonista uruchamia aplikację.
3. Recepcjonista klika “ustawienia”.
4. Recepcjonista wpisuje kod dostępu do bazy danych i klika “ok”.
5. Recepcjonista klika przycisk “zabiegi”.
6. Recepcjonista wyszukuje zabieg z bazy.
7. Recepcjonista klika w miejscu danych wybranego zabiegu.

8. Recepcjonista klika przycisk “usuń”.
9. Przypadek użycia kończy się w chwili usunięcia zabiegu z bazy.

Alternatywny przebieg zdarzeń: Recepcjonista rezygnuje z usunięcia zabiegu z bazy.
A8. Recepcjonista nie usuwa danych zabiegu z bazy klikając przycisk “powrót”.
A9. Przypadek użycia kończy się dalszą obecnością danych zabiegu w bazie.

Przypadek VII
Identyfikator: VII
Nazwa: Dodanie wizyty.
Opis: Zapewnia recepcjoniście możliwość dodania wizyty.
Stan początkowy: Klient chce umówić się na wizytę.
Stan końcowy: Klient jest umówiony na wizytę.
Aktorzy pierwszoplanowi: Recepcjonista
Aktorzy drugoplanowi: Klient
Podstawowy przebieg zdarzeń:
1. Przypadek użycia zaczyna się, kiedy klient zgłasza się do recepcji z zamiarem umówienia wizyty.
2. Recepcjonista uruchamia aplikację.
3. Recepcjonista klika “ustawienia”.
4. Recepcjonista wpisuje kod dostępu do bazy danych i klika “ok”.
5. Recepcjonista klika przycisk “lista wizyt”.
6. Recepcjonista klika przycisk “nowa wizyta”.
7. Recepcjonista wprowadza dane wizyty - wybiera datę i godzinę z kalendarza, wyszukuje klienta oraz zabieg z listy.
8. Recepcjonista zatwierdza dane wizyty klikając przycisk “zapisz”.
9. Przypadek użycia kończy się w chwili dodania zabiegu.

Alternatywny przebieg zdarzeń: W trakcie wpisywania danych wizyty przez recepcjonistę klient rezygnuje z usługi.
A8. Recepcjonista nie wprowadza wizyty klikając przycisk “powrót”.
A9. Przypadek użycia kończy się brakiem wprowadzenia nowej wizyty.

Przypadek VIII
Identyfikator VIII
Nazwa: Usunięcie wizyty.
Opis: Zapewnienie recepcjoniście możliwości usunięcia danych wizyty.
Stan początkowy: Klient ma umówioną wizytę i chce z niej zrezygnować.
Stan końcowy: Usunięcie umówionej wizyty.
Aktorzy pierwszoplanowi: Recepcjonista
Aktorzy drugoplanowi: Klient

Podstawowy przebieg zdarzeń:
1. Przypadek użycia zaczyna się w chwili zgłoszenia przez klienta rezygnacji z wizyty.
2. Recepcjonista uruchamia aplikację.
3. Recepcjonista klika “ustawienia”.
4. Recepcjonista wpisuje kod dostępu do bazy danych i klika “ok”.
5. Recepcjonista klika przycisk “lista wizyt”.
6. Recepcjonista wyszukuje wizytę na liście.
7. Recepcjonista klika w miejscu danych wybranej wizyty.
8. Recepcjonista klika przycisk “usuń”.
9. Przypadek użycia kończy się w chwili usunięcia wizyty.

Alternatywny przebieg zdarzeń: W trakcie usuwania danych wizyty przez recepcjonistę, klient zmienia zdanie.
A8. Recepcjonista nie usuwa danych wizyty klikając przycisk “powrót”.
A9. Przypadek użycia kończy się dalszą obecnością danych wizyty na liście..

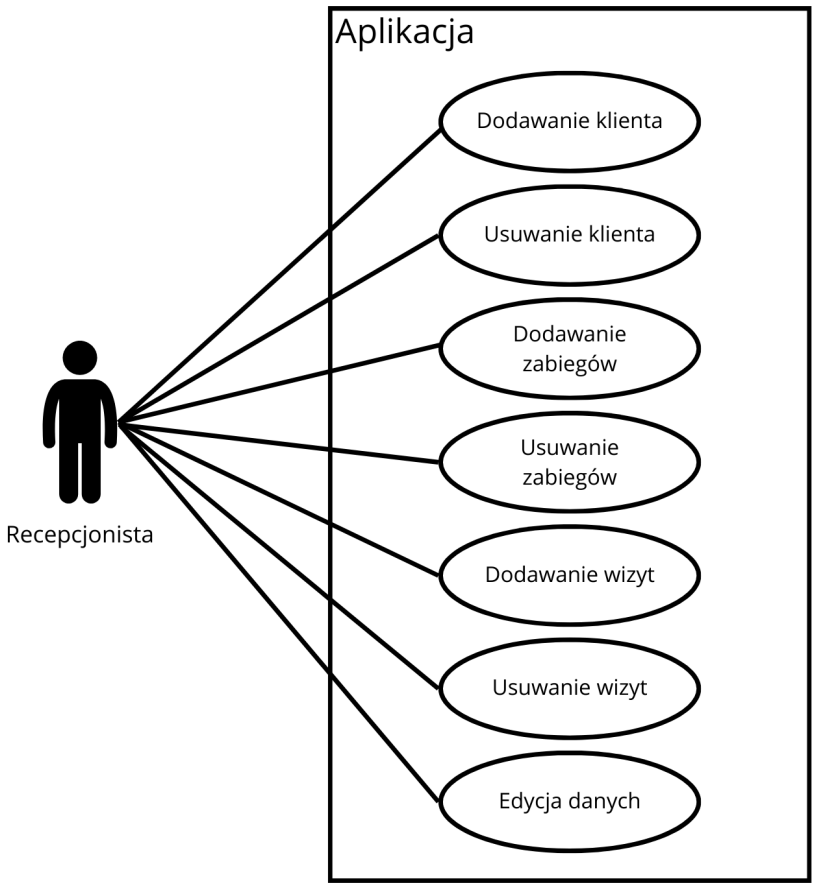
b) wymagania funkcjonalne i niefunkcjonalne
Wymagania funkcjonalne:

1) Menu główne:
    - przyciski do przejścia do sekcji: Klienci, Zabiegi, Lista wizyt, Ustawienia.
2) Zarządzanie klientami:
    - dodawanie nowego klienta (imię, nazwisko, telefon),
    - edycja danych istniejącego klienta,
    - usuwanie klienta,
    - wyświetlanie listy wszystkich klientów,
    - wyświetlanie szczegółowych informacji o kliencie.
3) Zarządzanie zabiegami:
    - dodawanie nowego zabiegu (nazwa, czas trwania, opis),
    - edycja danych istniejącego zabiegu,
    - usuwanie zabiegu,
    - wyświetlanie listy wszystkich zabiegów,
    - wyświetlanie szczegółowych informacji o zabiegu.
4) Zarządzanie wizytami:
    - dodawanie nowej wizyty (wybór klienta, zabiegu, daty i godziny),
    - usuwanie wizyt,
    - wyświetlanie listy wszystkich wizyt,
    - wyświetlanie szczegółowych informacji o wizycie.
5) Ustawienia:
    - wprowadzenie hasła dostępu do bazy danych

Wymagania niefunkcjonalne:

1) Użyteczność:
    - intuicyjny interface użytkownika,
    - łatwość nawigacji między sekcjami aplikacji,
2) Wydajność:
    - płynne działanie aplikacji.
3) Bezpieczeństwo:
    - szyfrowanie klucza API do bazy danych,
    - przesyłanie danych do bazy przy pomocy protokołu HTTPS,
    - ochrona przed nieautoryzowanym dostępem do danych.
4) Skalowalność:
    - możliwość dodawania nowych funkcjonalności w przyszłości.
5) Zgodność:
    - Aplikacja powinna działać poprawnie w popularnych przeglądarkach internetowych (Chrome, Firefox, Opera, Edge…).

c) diagram przypadków użycia



d) osoby uczestniczące w projekcie:
Aleksandra Olejarz
Magdalena Goszkowska

3. Projekt architektury
a) wybór technologii
Aplikacja będzie korzystała z:

- baza danych - internetowa baza danych typu NoSQL, wykorzystująca komunikację REST API (https://restdb.io).

W realizacji aplikacji zostaną wykorzystane:

- HTML - do strukturyzacji treści aplikacji.
- CSS - do określenia wyglądu elementów aplikacji.
- JavaScript - do obsługi zdarzeń, zapewnienia dynamiki oraz interaktywności aplikacji.
- Git - kontrola wersji.
- GitHub Pages - hosting strony.

b) projekt bazy danych

1) Klienci:
    - _id (unikalny identyfikator, generowany automatycznie przez bazę danych),
    - imię (tekst),
    - nazwisko (tekst),
    - tel (tekst).
2) Zabiegi:
    - _id (unikalny identyfikator, generowany automatycznie przez bazę danych),
    - nazwa (tekst),
    - czas (liczba),
    - opis tekst, (opcjonalny).
3) Zamówienia (wizyty):
    - _id (unikalny identyfikator, generowany automatycznie przez bazę danych),
    - idKlienta (tekst, odwołanie do id_klienta),
    - idZabiegu (tekst, odwołanie do id_zabiegu),
    - termin (data i czas).

Relacje:

- Jeden klient może zamówić wiele wizyt.
- Jeden zabieg może być częścią wielu wizyt.

c) przypadki użycia w modułach

Moduł konta klienta.

I Dodanie klienta do bazy.
II Edycja danych klienta.
III Usunięcie klienta z bazy.
IV Dodanie zabiegu do oferty.
V Edycja danych zabiegu.
VI Usunięcie zabiegu z bazy.
VII Dodanie wizyty do kalendarza.
VIII Edycja danych zapisanej wizyty.
IX Usunięcie wizyty z kalendarza.

4. Implementacja i testowanie aplikacji

a) Testy frontend

Wykorzystano testy end-to-end przy pomocy Selenium. Test skupia się na weryfikacji poprawności działania kluczowych funkcjonalności aplikacji z perspektywy użytkownika końcowego.

Zakres testów:

- test strony głównej - sprawdza poprawność wyświetlania strony głównej oraz dostępności przycisku “klienci”,
- test dodawania nowego klienta - weryfikuje proces dodawania nowego klienta do bazy danych, w tym poprawność działania formularza oraz wyświetlanie informacji o dodanym kliencie.

Narzędzia:

- Selenium - narzędzie do automatyzacji przeglądarek internetowych,
- ChromeDriver - sterownik przeglądarki Chrome dla Selenium,
- node.js - środowisko uruchomienia testów,
- assert - moduł node.js do sprawdzenia poprawności (tworzenia asercji).

Scenariusze testowe:

a1) Test strony głównej:

- otwórz przeglądarkę Chrome,
- przejdź na stronę główną aplikacji: https://aolejarz-57990.github.io,
- sprawdź, czy tytuł strony to “Aplikacja wielowarstwowa”,
- Sprawdź, czy przycisk “klienci” jest widoczny i da się go kliknąć.

a2) Test dodawania nowego klienta:

- otwórz przeglądarkę Chrome,
- przejdź na stronę główną aplikacji: https://aolejarz-57990.github.io,
- kliknij przycisk “klienci”
- poczekaj, aż przycisk “dodaj klienta” stanie się widoczny,

- kliknij przycisk “dodaj klienta”,
- wypełnij formularz danymi klienta:
  - Imię: Jan,
  - Nazwisko: Kowalski,
  - Telefon: 123456ABC,
- kliknij przycisk “zapisz”,
- poczekaj aż pojawi się przycisk “nowy klient”, co sygnalizuje zakończenie dodawania klienta

Oczekiwane rezultaty:
- strona główna wyświetla prawidłowy tytuł,
- formularz dodawania klienta działa poprawnie,
- nowy klient pojawia się na liście klientów.

Wykonanie testów:
- otworzenie terminala w Visual Studio Code,
- uruchomienie testu poleceniem: node./tests/frontend/testNowyKlinet.js

b) Testy backend
Wykorzystano testy jednostkowe skupiające się na weryfikacji poprawności działania funkcji odpowiedzialnych za zarządzanie klientami w module “zarządzanie_danymi.js”.

Zakres testów:
Testy jednostkowe obejmują funkcje:
- pobieranie listy klientów,
- dodawanie nowego klienta,
- pobieranie szczegółów klienta o ID ‘123’,
- aktualizację danych klienta,
- usunięcie klienta o ID ‘123’.

Narzędzia:
- jest - narzędzie do testowania,
- node.js - środowisko uruchomienia testów.

Scenariusze testowe:
b1) Test pobierania listy klientów:
- stwórz symulację i zażądaj pobrania listy klientów,
- sprawdź, czy żądanie zostało wywołane raz,
- sprawdź, czy żądanie zostało wywołane z poprawnymi parametrami: URL: https://aplikacja-ac0d.restdb.io/rest/klienci, metoda: ‘GET’.

b2) Test dodawania klienta:

- stwórz symulację i zażądaj dodania klienta z danymi testowymi do bazy,
- sprawdź, czy żądanie zostało wywołane raz,
- sprawdź, czy żądanie zostało wywołane z poprawnymi parametrami: URL: https://aplikacja-ac0d.restdb.io/rest/klienci, metoda: ‘POST’, dane: JSON.stringify(klient).

b3) Test pobierania danego klienta:

- stwórz symulację i zażądaj pobrania danych klienta o id ‘123’,
- sprawdź, czy czy żądanie zostało wywołane raz,
- sprawdź, czy żądanie zostało wywołane z poprawnymi parametrami: URL: https://aplikacja-ac0d.restdb.io/rest/klienci/123, metoda: ‘GET’.

b4) Test aktualizacji danych klienta:

- stwórz symulację i utwórz klienta z danymi testowymi,
- zażądaj aktualizacji danych klienta,
- sprawdź, czy czy żądanie zostało wywołane raz,
- sprawdź, czy żądanie zostało wywołane z poprawnymi parametrami: URL: https://aplikacja-ac0d.restdb.io/rest/klienci/undefined, metoda: ‘PUT’, dane: JSON.stringify(klient).

b5) Test usuwania klienta:

- stwórz symulację i zażądaj usunięcia klienta o id ‘123’
- sprawdź, czy żądanie zostało wywołane raz,
- sprawdź, czy żądanie zostało wywołane z poprawnymi parametrami: URL: https://aplikacja-ac0d.restdb.io/rest/klienci/123, metoda: ‘DELETE’.

Oczekiwane rezultaty:

- Test pobierania listy klientów: żądanie powinno zostać wywołane raz, żądanie powinno zostać wywołane z parametrami: URL: https://aplikacja-ac0d.restdb.io/rest/klienci, metoda: ‘GET’,
- Test dodawania klienta: żądanie powinno zostać wywołane raz, żądanie powinno zostać wywołane z parametrami: URL: https://aplikacja-ac0d.restdb.io/rest/klienci, metoda: ‘POST’, dane: JSON.stringify(klient),
- Test pobierania danego klienta: żądanie powinno zostać wywołane raz, żądanie powinno zostać wywołane z parametrami:
- URL: https://aplikacja-ac0d.restdb.io/rest/klienci/123, metoda: ‘GET’,
- Test aktualizacji danych klienta: żądanie powinno zostać wywołane raz, żądanie powinno zostać wywołane z parametrami:
- URL: https://aplikacja-ac0d.restdb.io/rest/klienci/undefined, metoda: ‘PUT’, dane: JSON.stringify(klient),
- Test usuwania klienta: żądanie powinno zostać wywołane raz, żądanie powinno zostać wywołane z parametrami:

URL: https://aplikacja-ac0d.restdb.io/rest/klienci/123, metoda: ‘DELETE’.

Wykonanie testów:

- otworzenie terminala w Visual Studio Code,
- uruchomienie testu poleceniem:npm test.

Raport testów:

npm test

test

jest

✔ Przykładowy test (0.760542ms)
✔ Testy zarządzania klientami (0.542042ms)
PASS tests/backend/zarzadzenie_danymi.test.js
✓ 2+2 powinno dać 4 (2 ms)
✓ pobierz klientów (1 ms)
✓ dodaj klienta
✓ pobierz klienta (2 ms)
✓ aktualizuj klienta
✓ usuń klienta (1 ms)

Test Suites: 1 passed, 1 total
Tests: 6 passed, 6 total
Snapshots: 0 total
Time: 0.139 s, estimated 1 s
Ran all test suites.
ℹ tests 0
ℹ suites 2
ℹ pass 0
ℹ fail 0
ℹ cancelled 0
ℹ skipped 0
ℹ todo 0
ℹ duration_ms 14.458791