BAZY DANYCH - PROJEKT

DOKUMENTACJA DO ETAPU 2

Kowalczyk Emilia 272539

Julia Rojek 272529

1 WYBRANE TECHNOLOGIE

1.1 Backend

W backendzie zastosowano szereg technologii, które wspierają efektywne przetwarzanie danych, bezpieczeństwo oraz łatwą integrację z frontendem. Główne technologie użyte w tej części projektu to:

1. **Node.js** - środowisko uruchomieniowe JavaScript, które pozwala na realizację aplikacji serwerowych przy użyciu języka JavaScript. Dzięki swojej asynchroniczności i architekturze opartej na zdarzeniach, Node.js jest wydajny i umożliwia obsługę wielu jednoczesnych połączeń, co jest kluczowe dla skalowalności aplikacji.
2. **Express.js** - framework aplikacji webowych oparty na Node.js, który upraszcza tworzenie interfejsów API i zarządzanie trasami w aplikacji. Express.js pozwala na szybkie i elastyczne konfigurowanie endpointów, obsługę żądań HTTP oraz middleware, co ułatwia implementację.
3. **Multer** - middleware do obsługi przesyłania plików w aplikacji Express. Używany jest głównie do przesyłania obrazów i ich walidacji w backendzie (np. weryfikacji typów plików oraz obsługi błędów podczas przesyłania). W projekcie multer jest wykorzystywany do obsługi zdjęć użytkowników i domków, dzięki czemu pozwala na łatwe i bezpieczne przetwarzanie danych multimedialnych.
4. **Body-parser** - biblioteka umożliwiająca parsowanie treści żądań HTTP do formatu JSON, co jest niezbędne do poprawnej interpretacji danych wysyłanych przez klienta (frontend) w żądaniach POST i PUT.
5. **CORS** (Cross-Origin Resource Sharing) - middleware pozwalający na dostęp do API z różnych domen i portów, co jest przydatne w przypadku, gdy frontend i backend aplikacji są hostowane oddzielnie (np. na różnych portach w lokalnym środowisku). CORS w projekcie pozwala frontendowi na wykonywanie żądań do backendu, zachowując odpowiednie poziomy bezpieczeństwa.
6. **MySQL** - relacyjna baza danych użyta do przechowywania informacji o użytkownikach, rezerwacjach, domkach oraz innych istotnych danych. Zastosowanie relacyjnej bazy danych pozwala na wydajne zarządzanie i przetwarzanie danych z użyciem języka SQL.
7. **MySQL2** - biblioteka pozwalająca na połączenie Node.js z bazą danych MySQL. W projekcie biblioteka ta odpowiada za nawiązywanie połączenia z bazą, wykonywanie zapytań SQL oraz zwracanie wyników do aplikacji. Wersja MySQL2 jest zoptymalizowana pod kątem wydajności oraz lepszego wsparcia dla operacji asynchronicznych.
8. **Dotenv** - biblioteka do zarządzania zmiennymi środowiskowymi. Umożliwia przechowywanie poufnych danych (takich jak dane dostępowe do bazy danych) w pliku .env, co pozwala na bezpieczniejsze zarządzanie konfiguracją bez ujawniania danych w kodzie źródłowym.

9. **Path** - moduł wbudowany w Node.js, używany do zarządzania ścieżkami dostępu do plików i katalogów. Path jest wykorzystywany do serwowania statycznych zasobów, takich jak obrazy czy pliki, w sposób niezależny od systemu operacyjnego.
10. **Bcrypt** – biblioteka przeznaczona do bezpiecznego haszowania haseł. Bcrypt umożliwia szyfrowanie haseł użytkowników przed zapisaniem ich do bazy danych, co znacząco zwiększa bezpieczeństwo projektu i chroni dane użytkowników przed potencjalnym przechwyceniem.

1.2 Frontend

W projekcie frontend został zaprojektowany z wykorzystaniem nowoczesnych technologii i narzędzi, które wspierają dynamiczne działanie aplikacji oraz zapewniają czytelny i intuicyjny interfejs użytkownika. Poniżej znajdują się technologie wykorzystane w warstwie frontendowej:

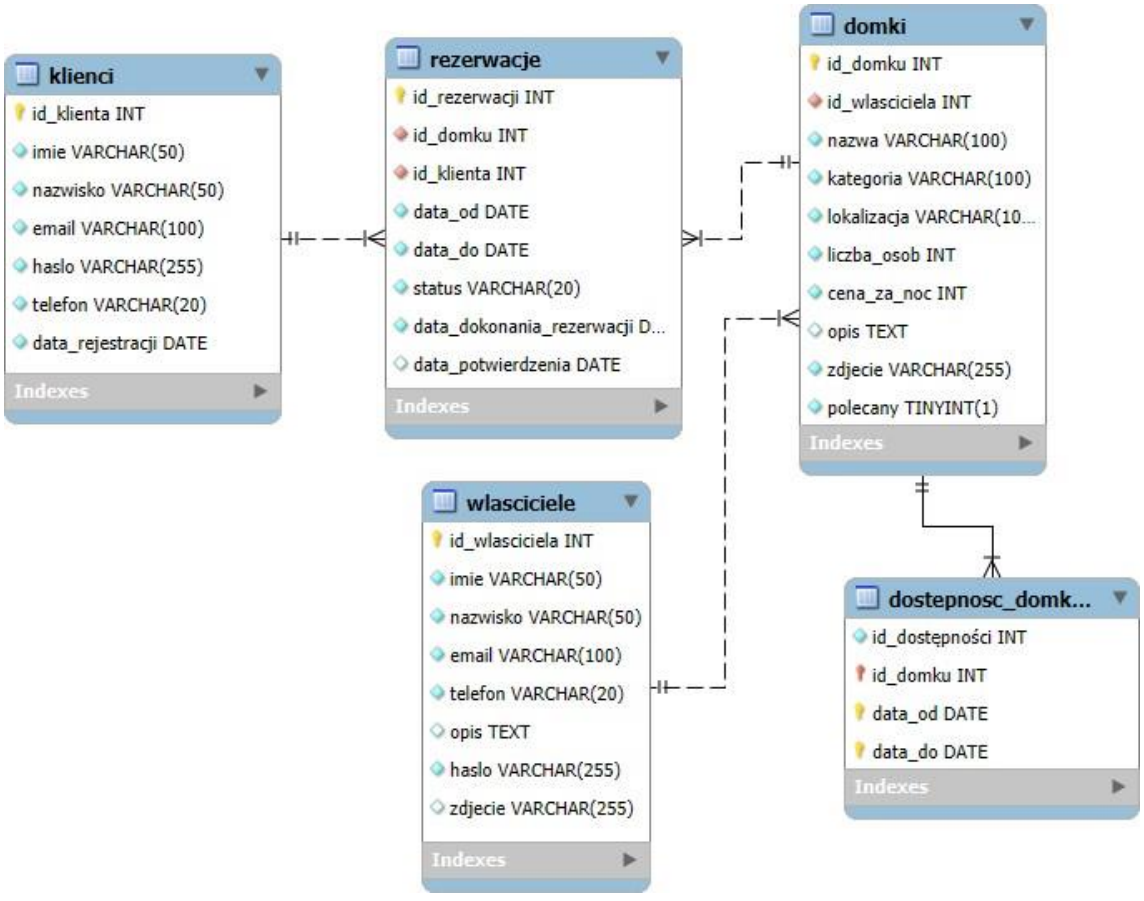
1. **React** - jest to biblioteka JavaScript, która umożliwia tworzenie komponentowych i dynamicznych aplikacji internetowych. Wykorzystuje **Virtual DOM** do efektywnego renderowania interfejsu użytkownika, co znacząco przyspiesza proces aktualizacji elementów strony bez konieczności przeładowywania całej aplikacji.
2. **React Router** - pozwala na nawigację pomiędzy różnymi widokami aplikacji bez przeładowania strony, co jest istotne w aplikacjach typu SPA (Single Page Application). W projekcie zastosowano na przykład komponenty takie jak Private_client_route oraz Private_owner_route, aby kontrolować dostęp do stron zarezerwowanych tylko dla klientów lub właścicieli, bazując na sesji użytkownika.
3. **React-datpicker i React-calendar** - pozwala na wygodne wybieranie zakresu dat, co jest widoczne w komponencie np. Search_form. Komponent umożliwia użytkownikom wybór dat w wygodny sposób oraz obsługę kalendarza, co podnosi użyteczność aplikacji.
4. **React Context API i React Hooks** - Narzędzia do zarządzania stanem aplikacji oraz uproszczenia operacji na komponentach funkcyjnych.
5. **Axios** - służy do wykonywania zapytań HTTP do serwera. Jest ona używana, na przykład, w komponentach Waiting oraz Confirmed, do komunikacji z backendem i pobierania danych o rezerwacjach. axios umożliwia łatwe przetwarzanie odpowiedzi serwera oraz obsługę błędów, co zapewnia spójność komunikacji między frontendem a backendem.
6. **Web Storage API** - Mechanizm przechowywania danych sesji użytkownika w przeglądarce.
7. **CSS i Sass** - stylowanie zostało zrealizowane z wykorzystaniem CSS, wspomaganego przez preprocesor Sass. Sass umożliwia bardziej modularne podejście do stylowania, dzięki wsparciu dla zmiennych, zagnieżdżania selektorów oraz dziedziczenia stylów.
8. **Darmowe bazy zdjęć** - możliwość korzystania z darmowych baz zdjęć, które oferują szeroki wybór materiałów wizualnych

2 ENCJE

2.1 Opis encji

1. **Klienci-** przechowuje dane klientów, którzy mogą dokonywać rezerwacji.
   - Atrybuty- id_klienta *(PK),* imię, nazwisko, email, telefon, data_rejestracji
2. **Rezerwacje-** Przechowuje dane o rezerwacjach dokonywanych przez klientów.
   - Atrybuty- *id_rezerwacji (PK), id_domku (FK), id_klienta (FK), data_od, data_do, status, data_dokonania_rezerwacji, data_potwierdzenia*
3. **Domki-** Przechowuje dane o dostępnych domkach do wynajmu.
   - Atrybuty- *id_domku (PK), id_wlasciciela (FK), nazwa, kategoria, lokalizacja, liczba_osob, cena_za_noc, opis, zdjecie, polecany*
4. **Dostępność domków-** Informacje o niedostępnych terminach dla poszczególnych domków.
   - Atrybuty- *id_dostępności (PK), id_domku (FK), data_od, data_do*
5. **Właściciele-** Przechowuje dane właścicieli, którzy posiadają domki do wynajęcia.
   - Atrybuty *id_wlasciciela (PK), imię, nazwisko, email, telefon, opis, haslo, zdjecie*

2.2 Graf encji



2.3 Normalizacja bazy danych

1. **Pierwsza forma normalna (1NF)**
    - ***Tabela klienci***: Przechowuje unikalne informacje o klientach (id_klienta, imie, nazwisko, email, telefon). Każdy rekord odpowiada jednemu klientowi, a kolumny przechowują tylko pojedyncze wartości.
    - ***Tabela domki***: Przechowuje unikalne informacje o każdym domku (id_domku, nazwa, kategoria, cena_za_noc). Każdy rekord odpowiada jednemu domkowi, a kolumny przechowują wyłącznie pojedyncze wartości.
    - ***Tabela rezerwacje:*** Przechowuje dane dotyczące każdej rezerwacji (id_rezerwacji, id_klienta, id_domku, data_od, data_do). Każdy rekord reprezentuje jedną rezerwację z jednoznacznie zidentyfikowanymi wartościami.
    - ***Tabela właściciele:*** Przechowuje unikalne informacje o właścicielach (id_właściciela, imie, nazwisko, email). Każdy rekord jest jednoznacznie zidentyfikowany przez id_właściciela i przechowuje pojedyncze wartości.
    - ***Tabela dostępność domków:*** Przechowuje dane o niedostępności domków (id_dostępności, id_domku, data_od, data_do). Każdy rekord odpowiada jednej niedostępności dla konkretnego domku i przechowuje jednoznaczne informacje.

Każda tabela jest w 1NF, ponieważ wszystkie kolumny zawierają tylko pojedyncze, niepodzielne wartości, a każdy rekord jest jednoznacznie identyfikowany przez klucz główny.

2. Druga Forma Normalna (2NF)
    - ***Tabela klienci:*** Kolumny jak imie, nazwisko, email, telefon są w pełni zależne od klucza głównego id_klienta.
    - ***Tabela domki:*** Kolumny jak nazwa, kategoria, cena_za_noc są w pełni zależne od klucza głównego id_domku.
    - ***Tabela rezerwacje:*** Kolumny jak data_od, data_do, status są zależne wyłącznie od id_rezerwacji, co zapewnia pełną zależność od klucza głównego.
    - ***Tabela właściciele***: Kolumny takie jak imię, nazwisko, email są zależne tylko od id_właściciela, co zapewnia pełną zależność od klucza głównego.
    - ***Tabela dostępność domków***: Kolumny takie jak data_od, data_do są zależne tylko od id_dostępności, co zapewnia pełną zależność od klucza głównego.

Każda tabela jest w 2NF, ponieważ wszystkie kolumny zależą wyłącznie od swojego klucza głównego i nie ma żadnych zależności częściowych.

3. Trzecia Forma Normalna (3NF)
    - ***Tabela klienci***: Kolumny jak email i telefon zależą tylko od id_klienta i nie są zależne od siebie nawzajem.
    - ***Tabela domki***: Kolumny jak cena_za_noc, kategoria są zależne wyłącznie od klucza głównego id_domku, a między kolumnami nie ma zależności, które mogłyby naruszać 3NF.
    - ***Tabela rezerwacje***: Kolumny takie jak data_od, data_do, status są niezależne od siebie i zależne tylko od id_rezerwacji.

- ***Tabela właściciele***: Kolumny jak imie, nazwisko, email są zależne wyłącznie od id_właściciela i nie mają między sobą zależności, które mogłyby naruszać 3NF.
- ***Tabela dostępność domków:*** Kolumny jak data_od, data_do są niezależne od siebie i zależne wyłącznie od id_dostępności.

Struktura bazy danych jest zoptymalizowana, a każda tabela przechowuje tylko niezbędne informacje, co minimalizuje redundancję

Dodatkowo przy ustawieniu opcji **CASCADE** dla kluczy obcych w tabelach **domki**, **rezerwacje** i **dostępność domków**, system zarządzania bazą danych automatycznie zarządza aktualizacjami i usuwaniem rekordów.

1. ***Tabela domki***
   - **On Update: CASCADE** - Kiedy wartość id_wlasciciela w tabeli właściciele zostanie zaktualizowana, system automatycznie zaktualizuje odpowiadające wartości id_wlasciciela we wszystkich rekordach tabeli domki.
   - **On Delete: CASCADE** - Kiedy właściciel zostanie usunięty z tabeli **właściciele**, wszystkie domki przypisane do tego właściciela zostaną również automatycznie usunięte z tabeli **domki**.
2. ***Tabela rezerwacje***
   - **On Update: CASCADE** (dla id_klienta i id_domku) - Kiedy wartość id_klienta lub id_domku zostanie zmieniona w tabelach **klienci** lub **domki**, odpowiednie rekordy w tabeli **rezerwacje** zostaną zaktualizowane, aby odzwierciedlać te zmiany. Jeśli ID klienta lub ID domku zostanie zaktualizowane, zmiana będzie także widoczna w tabeli **rezerwacje** we wszystkich rekordach, gdzie klient lub domek miał rezerwację.
   - **On Delete: CASCADE** - Kiedy klient lub domek zostanie usunięty, wszystkie rezerwacje powiązane z tym klientem lub domkiem zostaną automatycznie usunięte z tabeli **rezerwacje**. Przykładowo, jeśli usuniesz klienta, to wszystkie jego rezerwacje znikną; to samo dotyczy domku – usunięcie domku usunie wszystkie jego rezerwacje.
3. ***Tabela dostępność domków***
   - **On Update: CASCADE** (dla id_domku) - Jeśli id_domku zostanie zmienione w tabeli domki, wszystkie odpowiadające rekordy w tabeli **dostepnosc_domkow** zostaną automatycznie zaktualizowane.
   - **On Delete: CASCADE** - Jeśli domek zostanie usunięty, wszystkie wpisy w tabeli **dostepnosc_domkow** dotyczące tego domku zostaną automatycznie usunięte.

3 TRANSAKCJE

Opis możliwych transakcji:

1. ***Rejestracja klienta***
   - **Cel transakcji**: Umożliwienie nowemu klientowi utworzenia konta, aby mógł korzystać z systemu rezerwacji domków i zarządzać swoimi rezerwacjami.
   - **Kroki transakcji:**
     1. Klient podaje swoje dane (e-mail, hasło, imię i nazwisko)

2. System sprawdza, czy w tabeli klienci istnieje już rekord z tym samym adresem e-mail.
3. Jeśli e-mail jest unikalny, system zapisuje nowe konto klienta w tabeli klienci, dodając wiersz z podanymi danymi.

- **Warunki powodzenia:** Transakcja jest zakończona sukcesem, jeśli e-mail jest unikalny, a dane klienta zostały poprawnie zapisane w tabeli.

2. Logowanie klienta

- **Cel transakcji**: Umożliwienie klientowi dostępu do systemu, aby mógł zarządzać swoim kontem i rezerwacjami.
- **Kroki transakcji:**
    1. Klient wprowadza swój e-mail i hasło w celu zalogowania.
    2. System weryfikuje, czy podany e-mail i hasło są zgodne z danymi zapisanymi w tabeli klienci.
    3. Jeśli dane są zgodne, klient uzyskuje dostęp do swojego konta.
- **Warunki powodzenia**: Logowanie jest zakończone sukcesem, jeśli e-mail i hasło są zgodne z zapisanymi danymi w tabeli klienci.

3. Zarezerwowanie domku

- **Cel transakcji**: Umożliwienie klientowi dokonania rezerwacji wybranego domku na określony termin.
- **Kroki transakcji**:
    1. Klient wybiera domek, który chce zarezerwować, oraz podaje daty rezerwacji.
    2. System sprawdza dostępność domku na wybrany okres, przeszukując tabelę dostepnosc_domkow i tabelę rezerwacje.
    3. Jeśli domek jest dostępny, system zapisuje nową rezerwację w tabeli rezerwacje, dodając rekord z informacjami o kliencie, domku, dacie rozpoczęcia i zakończenia rezerwacji oraz ustawiając odpowiednio status rezerwacji.
- **Warunki powodzenia: Rezerwacja jest zakończona** sukcesem, jeśli domek jest dostępny na podany termin, a wszystkie dane zostały poprawnie zapisane.

4. Anulowanie rezerwacji przez klienta

- **Cel transakcji**: Umożliwienie klientowi anulowania swojej rezerwacji, jeśli zdecyduje się z niej zrezygnować.
- **Kroki transakcji**:
    1. Klient przegląda swoje aktywne rezerwacje i wybiera tę, którą chce anulować.
    2. System sprawdza, czy wybrana rezerwacja należy do zalogowanego klienta.
    3. System zmienia usuwa daną rezerwacje z tabeli rezerwacje.
- **Warunki powodzenia**: Transakcja jest zakończona sukcesem, jeśli klient jest właścicielem danej rezerwacji i domek został odblokowany.

5. Podgląd danych właściciela przez klienta

- **Cel transakcji**: Umożliwienie klientowi przeglądania danych kontaktowych właściciela, aby mógł się z nim skontaktować w razie potrzeby.
- **Kroki transakcji**:
    1. Klient wybiera domek, który zamierza zarezerwować lub z którym chce się zapoznać
    2. System pobiera dane właściciela z tabeli właściciele powiązanej z wybranym domkiem, takie jak imię, nazwisko, e-mail i numer telefonu. System wyświetla klientowi dane kontaktowe właściciela.

- **Warunki powodzenia**: Transakcja jest zakończona sukcesem, jeśli klient jest właścicielem danej rezerwacji i domek został odblokowany.

6. ***Filtrowanie domków***
   - **Cel transakcji**: Umożliwienie klientowi filtrowania dostępnych domków według wybranych kryteriów, aby znaleźć najlepiej dopasowaną ofertę.
   - **Kroki transakcji**:
     1. Klient wybiera kryteria filtrowania, takie jak lokalizacja, cena, liczba miejsc, kategoria i data dostępności.
     2. System pobiera listę domków z tabeli domki, tabelę dostepnosc_domkow i tabelę rezerwacje, które spełniają wybrane kryteria.
     3. System wyświetla klientowi listę domków, które są dostępne i spełniają kryteria filtrowania.
   - **Warunki powodzenia**: Transakcja kończy się sukcesem, jeśli system zwraca listę domków, które spełniają wszystkie wybrane kryteria filtrowania.
7. ***Sprawdzenie dostępności domku***
   - **Cel transakcji**: Umożliwienie klientowi sprawdzenia dostępności wybranego domku na określony termin.
   - **Kroki transakcji**:
     1. Klient wybiera domek i określa termin, w którym chce sprawdzić jego dostępność.
     2. System przeszukuje tabelę rezerwacje i dostepnosc_domkow, aby sprawdzić, czy domek jest dostępny na wybrany termin.
     3. System wyświetla klientowi informację o dostępności domku w wybranym okresie.
   - **Warunki powodzenia**: Sprawdzenie jest zakończone sukcesem, jeśli system poprawnie zwróci informacje o dostępności domku.
8. ***Przeglądanie ofert właściciela***
   - **Cel transakcji**: Umożliwienie właścicielowi przeglądania listy jego domków oraz bieżącego obłożenia rezerwacjami, aby mógł zarządzać swoimi ofertami.
   - **Kroki transakcji**:
     1. Właściciel loguje się do systemu i wybiera opcję przeglądania swoich ofert.
     2. System pobiera listę domków należących do właściciela z tabeli domki.
     3. System przeszukuje tabelę rezerwacje, aby wyświetlić aktualne rezerwacje dla każdego domku właściciela.
     4. System wyświetla właścicielowi listę jego domków i bieżące obłożenie rezerwacjami.
   - **Warunki powodzenia**: Transakcja kończy się sukcesem, jeśli wszystkie domki i powiązane rezerwacje zostały poprawnie wyświetlone.
9. ***Anulowanie rezerwacji przez właściciela***
   - **Cel transakcji**: Umożliwienie właścicielowi anulowania rezerwacji na prośbę klienta lub w wyjątkowych sytuacjach.
   - **Kroki transakcji**:
     1. Właściciel przegląda listę rezerwacji swoich domków i wybiera rezerwację do anulowania.
     2. System sprawdza, czy wybrana rezerwacja dotyczy domku należącego do właściciela.
     3. System zmienia status rezerwacji na „odrzucona” w tabeli rezerwacje.

- **Warunki powodzenia**: Anulowanie zakończy się sukcesem, jeśli właściciel jest właścicielem domku i rezerwacja została poprawnie anulowana.

4 PROGZNOZA

Prognoza charakteru poszczególnych encji:

1. ***Encja klienci***
    a. **Przewidywany sposób użycia:**
        - Głównie **odczyt**, podczas logowania oraz wyświetlania danych klienta.
        - Sporadyczne operacje **zapisu**, gdy nowy klient zakłada konto lub aktualizuje swoje dane.
    b. **Przewidywana zmienność**:
        - **Dodawanie**: Nowi klienci będą dodawani podczas rejestracji, więc dodawanie będzie występować regularnie, ale niezbyt często.
        - **Edycja**: Rzadko, głównie jeśli klient zdecyduje się na aktualizację swoich danych (np. zmiana e-maila czy hasła).
        - **Usuwanie**: Sporadyczne, np. w przypadku usunięcia konta przez klienta.
    c. **Przewidywana liczba wystąpień obiektów**:
        - Liczba rekordów będzie rosnąć wraz z rejestracją nowych klientów. Można przewidzieć umiarkowaną liczbę, np. setki do tysięcy rekordów, zależnie od popularności systemu.
2. ***Encja właściciele***
    a. **Przewidywany sposób użycia**:
        - Głównie **odczyt**, gdy klient przegląda dane kontaktowe właściciela.
        - Sporadyczne operacje **zapisu**, gdy właściciel aktualizuje swoje dane.
    b. **Przewidywana zmienność**:
        - **Dodawanie**: Dodawanie będzie rzadkie, gdyż liczba właścicieli jest ograniczona i nie zmienia się tak dynamicznie jak liczba klientów.
        - **Edycja**: Rzadko, głównie jeśli właściciel chce zmienić swoje dane kontaktowe
        - **Usuwanie**: Bardzo rzadkie, gdy właściciel decyduje się na usunięcie konta.
    c. **Przewidywana liczba wystąpień obiektów**:
        - Liczba rekordów będzie dużą, prawdopodobnie od dziesiątek do setek.
3. ***Encja domki***
    a. **Przewidywany sposób użycia**:
        - **Odczyt**: Bardzo częsty, gdy klienci przeglądają dostępne domki, sprawdzają ich szczegóły lub dokonują rezerwacji
        - **Zapis**: Rzadki, głównie podczas dodawania nowego domku lub aktualizacji danych domku przez właściciela.
    b. **Przewidywana zmienność**:
        - **Dodawanie**: Będzie sporadyczne, gdy właściciel doda nowy domek do oferty.
        - **Edycja**: Bardzo rzadko, gdy właściciel aktualizuje informacje o domku.
        - **Usuwanie**: Bardzo rzadkie, np. w przypadku wycofania domku z oferty.
    c. **Przewidywana liczba wystąpień obiektów**:
        - Zależnie od liczby właścicieli i ich oferty, może sięgać dziesiątek do setek rekordów.

4. ***Encja rezerwacje***
    a. **Przewidywany sposób użycia**:
        - **Odczyt**: Częsty, gdy klienci lub właściciele przeglądają aktywne rezerwacje.
        - **Zapis**: Również częsty, ponieważ każdy proces rezerwacji, anulowania rezerwacji czy aktualizacji terminu wiąże się z zapisami w tej tabeli.
    b. **Przewidywana zmienność**:
        - **Dodawanie**: Często, ponieważ każda nowa rezerwacja dodaje nowy rekord.
        - **Edycja**: Możliwa, np. przy zmianie statusu rezerwacji na „odrzucona" lub „potwierdzona".
        - **Usuwanie**: Sporadyczne, możliwe w przypadku anulacji rezerwacji
    c. **Przewidywana liczba wystąpień obiektów**:
        - Liczba rekordów może być znaczna, zależnie od liczby aktywnych klientów, może sięgać setek, a nawet tysięcy.

5. ***Encja dostępność domków***
    a. **Przewidywany sposób użycia**:
        - **Odczyt**: Bardzo częsty, ponieważ system musi sprawdzać dostępność domków na różne terminy przy każdej próbie rezerwacji lub przeglądania ofert.
        - **Zapis**: Sporadyczny, tylko w przypadku dodania rekordu przez admina na prośbę właściciela.
    b. **Przewidywana zmienność**:
        - **Dodawanie**: Sporadyczne, tylko na szczególną prośbę właściciela
        - **Edycja**: Sporadyczna, gdy właściciel zdecyduje się zmienić okresy dostępności lub je dostosować
        - **Usuwanie**: Bardzo rzadka, usunięcie jest uzasadnione tylko w wyjątkowych sytuacjach np. gdy właściciel chce usunąć domek ze swojej oferty.
    c. **Przewidywana liczba wystąpień obiektów**:
        - Raczej niska, rekordy w encji **dostępność domków** są tworzone sporadycznie, na prośbę właściciela, i rzadko usuwane. Oznacza to, że baza nie będzie szybko rosnąć.