Dokumentacja Projektu: Strona Internetowa dla Salonu Tatuaży

Etap 1

Skład zespołu: Kateryna Fedoruk i Maciej Grzyb

Październik 2024

Wstęp

Celem projektu jest stworzenie strony internetowej dla salonu tatuażu, która będzie pełnić funkcję zarówno informacyjną, jak i wspierającą rezerwacje wizyt. Strona umożliwi klientom przeglądanie portfolio tatuaży, zapoznanie się z profilami artystów oraz samodzielne rezerwowanie terminów wizyt. W projekcie uwzględniono specyfikację wymagań zgodnie z praktykami stosowanymi w komercyjnym oprogramowaniu, obejmującą user stories, przypadki użycia, wstępne makiety oraz diagramy UML i BPMN.

Etap ten obejmuje analizę biznesową, której celem jest określenie zakresu funkcjonalności i wymagań projektu od strony użytkownika. Dokumentacja powstała w formacie PDF przy użyciu systemu składu LaTeX. Niniejszy dokument zawiera szczegółowe wymagania funkcjonalne, user stories, przypadki użycia, diagramy BPMN oraz UML, a także wstępne makiety, które wspierają zrozumienie funkcji i przepływu w systemie.

Wymagania Funkcjonalne

Gość (Użytkownik niezalogowany)

- **Rejestracja jako klient** – system powinien umożliwiać użytkownikowi niezalogowanemu na stworzenie konta poprzez wypełnienie formularza, w którym wprowadzane są imię, nazwisko, numer telefonu, adres email oraz hasło.
- **Logowanie się jako klient** – system po wybraniu opcji logowania jako klient ma umożliwić zalogowanie się po podaniu adresu email oraz poprawnego hasła.
- **Logowanie się jako kierownik** – system po wybraniu opcji logowania jako kierownik ma umożliwić zalogowanie się po podaniu adresu email oraz poprawnego hasła.
- **Przeglądanie portfolio** – Gość powinien posiadać możliwość przeglądania dostępnych stylów tatuaży w zakładce "Portfolio". Po wybraniu konkretnego stylu (np. tatuaż na plecach, na łydce), gość powinien zobaczyć galerię zdjęć tatuaży wykonanych w wybranym stylu.
- **Przeglądanie artystów** – Gość powinien mieć możliwość przeglądania listy artystów w zakładce "Artyści". Po kliknięciu na konkretnego artystę, gość powinien zobaczyć krótką informację o artyście oraz jego portfolio w formie zdjęć.
- **Przeglądanie cennika** – Gość powinien mieć możliwość przeglądania informacji dotyczących cen tatuaży w zakładce "Cena tatuażu".
- **Kontakt z salonem** – Gość powinien mieć dostęp do zakładki "Kontakt", gdzie znajdzie dane kontaktowe do salonu, takie jak numer telefonu, adres e-mail oraz formularz kontaktowy.

Klient

- **Umówienie terminu** – Zalogowany użytkownik powinien mieć możliwość rezerwacji terminu tatuażu w zakładce „Umów wizytę”, gdzie po wybraniu artysty i terminu może dokonać rezerwacji. System sprawdza dostępność terminu i potwierdza rezerwację lub wyświetla komunikat o niepowodzeniu.
- **Wylogowanie się** – Zalogowany użytkownik powinien mieć możliwość wylogowania się z konta, aby zakończyć sesję.

Menedżer

- **Zarządzanie portfolio** – Menedżer powinien mieć możliwość dodawania i usuwania zdjęć w zakładce „Portfolio”, aby aktualizować prezentowane style tatuaży oraz zarządzać zdjęciami przypisanymi do poszczególnych artystów, prezentując ich aktualne prace.
- **Zarządzanie artystami** – Menedżer powinien mieć możliwość dodawania i usuwania artystów z listy artystów oraz dodawania i usuwania dostępnych terminów pracy artystów, co umożliwia bieżącą kontrolę nad ich dostępnością.
- **Zarządzanie treściami na stronie** – Administrator powinien mieć możliwość edytowania opisów w zakładkach takich jak "Artyści", „Cennik” itp, co pozwala na bieżącą aktualizację informacji.
- **Pobranie danych użytkownika** – Administrator powinien mieć możliwość uzyskania danych dotyczących użytkownika, który dokonał potwierdzonej rezerwacji.
- **Wylogowywanie** – po wybraniu opcji wyloguj menedżer zostaje wylogowany i powraca do poziomu uprawnień gościa.

User Stories

User stories opisują funkcjonalności systemu z perspektywy różnych typów użytkowników, takich jak Gość, Klient oraz Menedżer. Każda historia użytkownika przedstawia krótki scenariusz interakcji, który ma na celu osiągnięcie określonego celu przez danego użytkownika. W ten sposób definiujemy wymagania systemu i funkcje, które powinien spełniać, aby odpowiedzieć na potrzeby wszystkich grup użytkowników.

User Story dla Gościa:

- Gość przegląda portfolio, aby zobaczyć przykłady tatuaży dostępnych w salonie.
  Kryteria akceptacji:
  - Gość przegląda galerię tatuaży, widząc miniatury zdjęć.
- Gość przegląda profile artystów, aby poznać ich styl pracy.
  Kryteria akceptacji:
  - Gość ma dostęp do listy artystów i może kliknąć na nazwisko, aby zobaczyć opis i przykłady prac.
- Gość przegląda cennik, aby dowiedzieć się o kosztach tatuaży.
  Kryteria akceptacji:
  - Gość widzi tabelę z orientacyjnymi cenami dla różnych tatuaży i sesji.
- Gość rejestruje się, aby móc umawiać się na tatuaż.
  Kryteria akceptacji:
  - Użytkownik tworzy konto, ustalając nazwę użytkownika, numer telefonu oraz hasło.

User Story dla Klienta:

- Klient loguje się na swoje konto, aby móc umawiać się na tatuaż.
  Kryteria akceptacji:
  - ◇ Klient podaje nazwę użytkownika i hasło.
  - ◇ System wyświetla komunikat o błędzie przy błędnych danych.
- Klient rezerwuje termin tatuażu u wybranego artysty.
  Kryteria akceptacji:
  - ◇ Klient wybiera opcję „Umów wizytę" w profilu artysty, przechodzi do wyboru terminu.
  - ◇ System potwierdza rezerwację lub wyświetla komunikat o braku dostępności.
- Klient ma możliwość zmiany lub anulowania rezerwacji.
  Kryteria akceptacji:
  - ◇ Klient anuluje rezerwację, kontaktując się z salonem (telefon, e-mail, formularz).
  - ◇ Klient anuluje starą rezerwację i rezerwuje nowy termin w celu zmiany.

User Story dla Menedżera:

- Menedżer zarządza portfolio, aby aktualizować zdjęcia tatuaży.
  Kryteria akceptacji:
  - ◇ Menedżer dodaje, usuwa zdjęcia oraz zmienia opisy.
- Menedżer zarządza profilem artystów, aby aktualizować informacje o zespole.
  Kryteria akceptacji:
  - ◇ Menedżer dodaje, usuwa artystów i aktualizuje ich profile.
- Menedżer zarządza kalendarzem artystów, aby kontrolować ich dostępność.
  Kryteria akceptacji:
  - ◇ Menedżer dodaje, edytuje i usuwa dostępne terminy artystów.
- Menedżer ma możliwość odwoływania wizyt klientów w razie nieprzewidzianych okoliczności.
  Kryteria akceptacji:
  - ◇ Menedżer identyfikuje klientów po numerze telefonu, kontaktuje się, informując o anulacji.

Use Cases

Przypadki użycia (Use Cases) szczegółowo opisują konkretne funkcjonalności systemu oraz interakcje pomiędzy użytkownikami a systemem. Każdy przypadek użycia definiuje przebieg główny (oraz alternatywny) danego procesu, wskazując na wymagania funkcjonalne oraz logikę działania systemu. Dzięki temu możliwe jest prześledzenie pełnego przepływu poszczególnych operacji, takich jak rejestracja, logowanie czy rezerwacja wizyty.

Use Case 1: Przeglądanie Portfolio

- **Opis:** Gość przegląda portfolio tatuaży dostępnych w salonie.
- **Aktorzy:** Gość
- **Warunki początkowe:** Gość otwiera stronę główną aplikacji.
- **Warunki końcowe:** Gość zapoznaje się ze szczegółami wybranego tatuażu.
- **Przebieg główny:**
  1. Gość wybiera zakładkę „Portfolio”.
  2. System wyświetla miniatury tatuaży.
  3. Gość klika na zdjęcie tatuażu, aby powiększyć je i zobaczyć szczegóły.
  4. System wyświetla powiększone zdjęcie z informacjami (artysta).

Use Case 2: Przeglądanie Profili Artystów

- **Opis:** Gość przegląda profile artystów, aby poznać ich opis i prace.
- **Aktorzy:** Gość
- **Warunki początkowe:** Gość otwiera stronę główną aplikacji.
- **Warunki końcowe:** Gość zapoznaje się z pracami wybranego artysty.
- **Przebieg główny:**
  1. Gość wybiera zakładkę „Artyści”.
  2. System wyświetla profile artystów (opis, galeria zdjęć).

Use Case 3: Rejestracja Konta Klienta

- **Opis:** Gość rejestruje konto, aby umawiać się na tatuaż.
- **Aktorzy:** Gość
- **Warunki początkowe:** Gość otwiera stronę rejestracji.
- **Warunki końcowe:** Konto Gościa jest aktywne.
- **Przebieg główny:**
  1. Gość wypełnia formularz rejestracyjny, podając wymagane dane.
  2. System sprawdza dostępność nazwy użytkownika.
  3. Konto zostaje aktywowane.
- **Przebieg alternatywny:**
  1. Jeśli podana nazwa użytkownika nie jest dostępna, system wyświetla komunikat o błędzie.

Use Case 4: Logowanie Klienta

- **Opis:** Klient loguje się do swojego konta.
- **Aktorzy:** Klient
- **Warunki początkowe:** Klient posiada konto w systemie.
- **Warunki końcowe:** Klient jest zalogowany.
- **Przebieg główny:**
  1. Klient podaje nazwę użytkownika i hasło.
  2. System weryfikuje dane i loguje Klienta.
- **Przebieg alternatywny:**
  1. Jeśli dane są błędne, system wyświetla komunikat o błędzie.

Use Case 5: Przeglądanie Dostępnych Terminów

- **Opis:** Klient przegląda dostępne terminy, aby umówić się na tatuaż.
- **Aktorzy:** Klient
- **Warunki początkowe:** Klient jest zalogowany.
- **Warunki końcowe:** Klient dokonuje rezerwacji terminu.
- **Przebieg główny:**
  1. System wyświetla artystów i ich dostępne terminy.
  2. Klient wybiera termin i potwierdza rezerwację.
  3. System weryfikuje dostępność wybranego terminu i blokuje wybrany termin.
- **Przebieg alternatywny:**
  1. Jeśli termin nie jest dostępny, system wyświetla komunikat o błędzie.

Use Case 6: Kontakt z Salonem

- **Opis:** Gość lub Klient wysyła zapytanie lub wiadomość do salonu przez formularz kontaktowy.
- **Aktorzy:** Gość, Klient
- **Warunki początkowe:** Użytkownik otwiera stronę „Kontakt”.
- **Warunki końcowe:** Wiadomość jest wysłana, a użytkownik otrzymuje potwierdzenie wysłania.
- **Przebieg główny:**
  1. Użytkownik wypełnia formularz kontaktowy.
  2. System wysyła wiadomość do salonu i wyświetla potwierdzenie wysłania.

Use Case 7: Zarządzanie Portfolio przez Menedżera

- **Opis:** Menedżer dodaje lub usuwa zdjęcia w portfolio.
- **Aktorzy:** Menedżer
- **Warunki początkowe:** Menedżer jest zalogowany.
- **Warunki końcowe:** Portfolio zostaje zaktualizowane.
- **Przebieg główny:**
  1. Menedżer wybiera opcję zarządzania portfolio.
  2. System wyświetla istniejące zdjęcia.
  3. Menedżer dodaje nowe zdjęcie lub usuwa wybrane.
  4. System zapisuje zmiany i aktualizuje portfolio.

Use Case 8: Zarządzanie Profilami Artystów przez Menedżera

- **Opis:** Menedżer edytuje profile artystów.
- **Aktorzy:** Menedżer
- **Warunki początkowe:** Menedżer jest zalogowany.
- **Warunki końcowe:** Profil artysty zostaje zaktualizowany.
- **Przebieg główny:**
  1. Menedżer wybiera profil artysty do edycji.
  2. System wyświetla dane profilu.
  3. Menedżer dokonuje zmian (opis, zdjęcia, itp.).
  4. System zapisuje zmiany i aktualizuje profil artysty.

Use Case 9: Zarządzanie Dostępnymi Terminami

- **Opis:** Menedżer dodaje, edytuje lub usuwa dostępne terminy artystów.
- **Aktorzy:** Menedżer
- **Warunki początkowe:** Menedżer jest zalogowany.
- **Warunki końcowe:** Terminy artysty są zaktualizowane.
- **Przebieg główny:**
  1. Menedżer wybiera opcję zarządzania terminami artystów.
  2. System wyświetla dostępne terminy artystów.
  3. Menedżer dodaje, edytuje lub usuwa wybrane terminy.
  4. System zapisuje zmiany i aktualizuje dostępność w kalendarzu.

Opis diagramów przypadków użycia i procesów BPMN

W tej sekcji przedstawiono diagramy przypadków użycia i BPMN, które ilustrują kluczowe procesy w systemie strony internetowej salonu tatuaży. System ten oferuje różnorodne funkcjonalności, od rejestracji i logowania użytkowników, przez przeglądanie portfolio artystów, aż po zarządzanie profilami artystów i rezerwacją wizyt.

Diagram UML - przypadki użycia

Diagram przypadków użycia przedstawia funkcjonalności dostępne dla różnych typów użytkowników: Gościa, Klienta oraz Menedżera. Gość, jako niezalogowany użytkownik, może przeglądać portfolio, przeglądać profil artysty oraz zarejestrować się i zalogować. Klient, po zalogowaniu, zyskuje możliwość dokonania rezerwacji wizyty, obejmującą wybór terminu oraz potwierdzenie rezerwacji. Menedżer, jako użytkownik z uprawnieniami administracyjnymi, może zarządzać portfolio i profilami artystów, co obejmuje dodawanie i usuwanie prac z portfolio, tworzenie oraz edytowanie profili artystów, a także zarządzanie dostępnością terminów. Relacje *include* między przypadkami użycia ilustrują zależności i wymagane kroki w poszczególnych procesach.

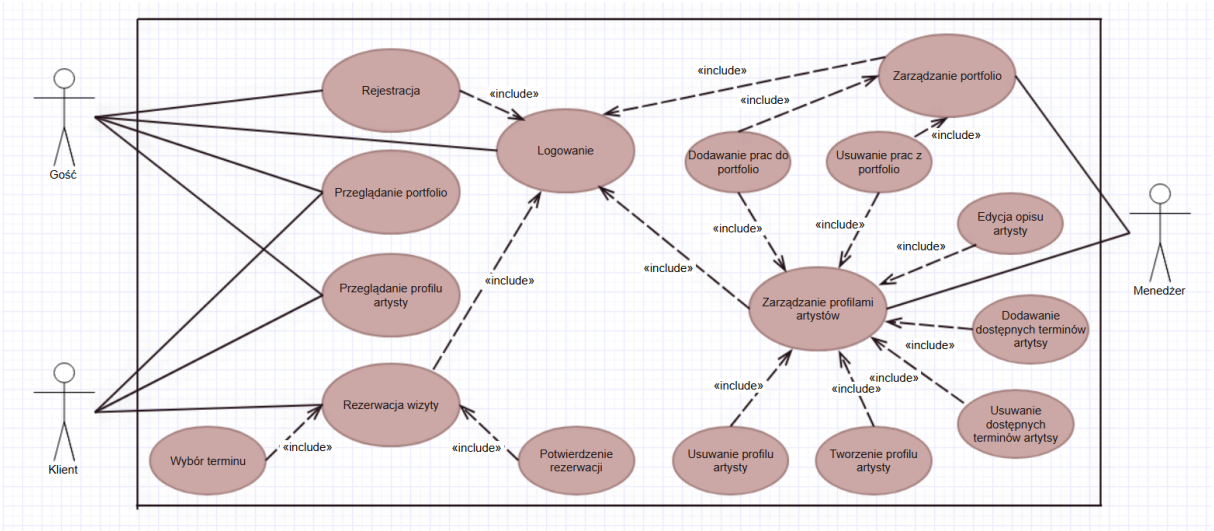

Figure 1: Diagram przypadków użycia

Diagram BPMN - Proces rejestracji użytkownika

Diagram BPMN dla procesu rejestracji użytkownika opisuje kroki niezbędne, aby Gość mógł założyć konto w systemie. Proces rozpoczyna się od wejścia użytkownika na stronę rejestracji. Następnie użytkownik wypełnia formularz rejestracyjny, po czym system sprawdza dostępność wybranej nazwy użytkownika. Jeśli nazwa użytkownika jest zajęta, system wyświetla odpowiednią informację, umożliwiając wybór innej nazwy. W przypadku dostępności nazwy, system zapisuje dane użytkownika i kończy proces rejestracji sukcesem. Diagram ten jasno przedstawia logiczny przepływ działań i decyzji podejmowanych podczas rejestracji.

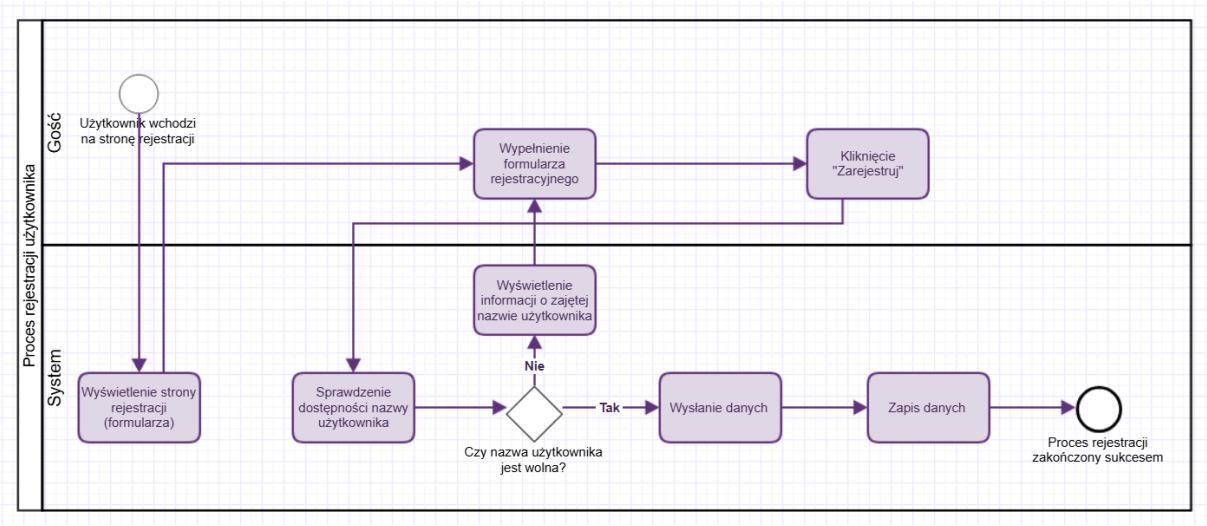

Figure 2: Diagram BPMN - Proces rejestracji użytkownika

Diagram BPMN - Proces logowania użytkownika

Diagram BPMN przedstawia proces logowania użytkownika, który umożliwia uzyskanie dostępu do dodatkowych funkcjonalności strony. Użytkownik, będący Gościem, wchodzi na stronę logowania i wprowadza swoje dane logowania. System weryfikuje poprawność wprowadzonych danych. W przypadku niepoprawnych danych system wyświetla komunikat o błędzie, umożliwiając ponowne wprowadzenie danych. Gdy dane są prawidłowe, użytkownik zostaje zalogowany i uzyskuje pełny dostęp do swojego konta oraz funkcji, takich jak rezerwacja wizyt.

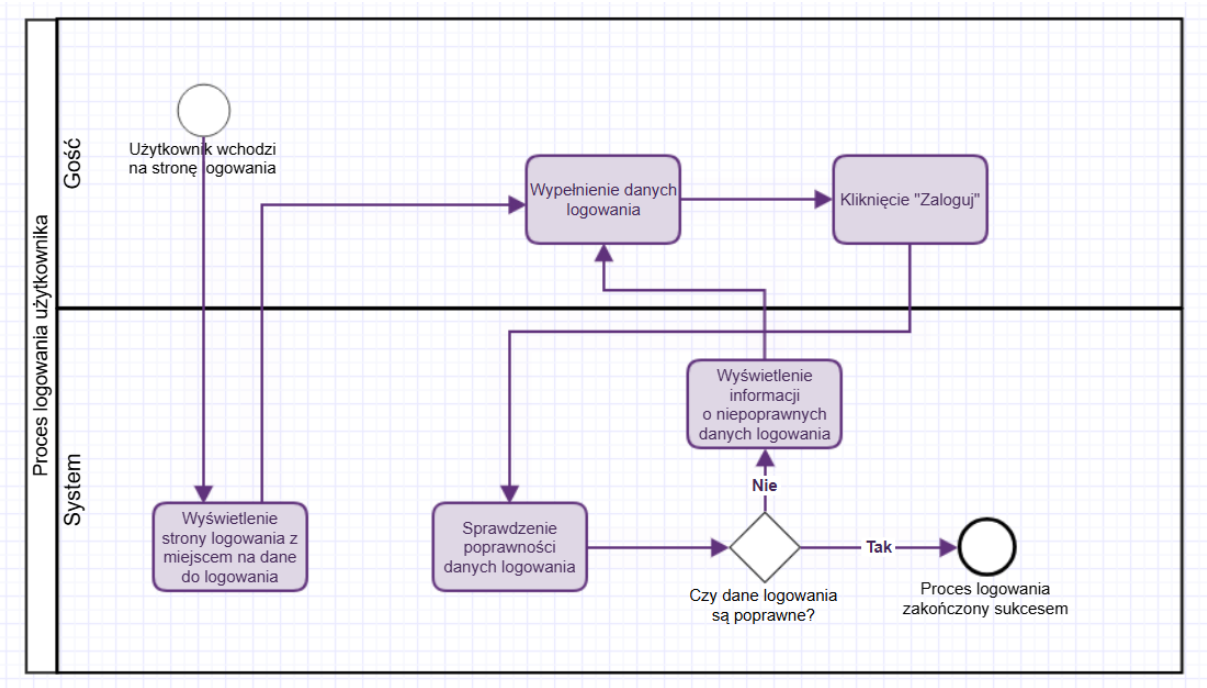

Figure 3: Diagram BPMN - Proces logowania użytkownika

Diagram BPMN - Proces rezerwacji wizyty

Diagram BPMN dla procesu rezerwacji wizyty przedstawia kroki wykonywane przez Klienta, który chce umówić wizytę w salonie tatuaży. Proces rozpoczyna się od kliknięcia przycisku „Umów wizytę", po czym system wyświetla dostępne terminy i artystów. Użytkownik wybiera preferowanego artystę i termin, a system sprawdza dostępność wybranego terminu. Jeśli termin jest zajęty, użytkownik jest informowany o niepowodzeniu rezerwacji. W przypadku dostępności terminu system zapisuje rezerwację, kończąc proces sukcesem.

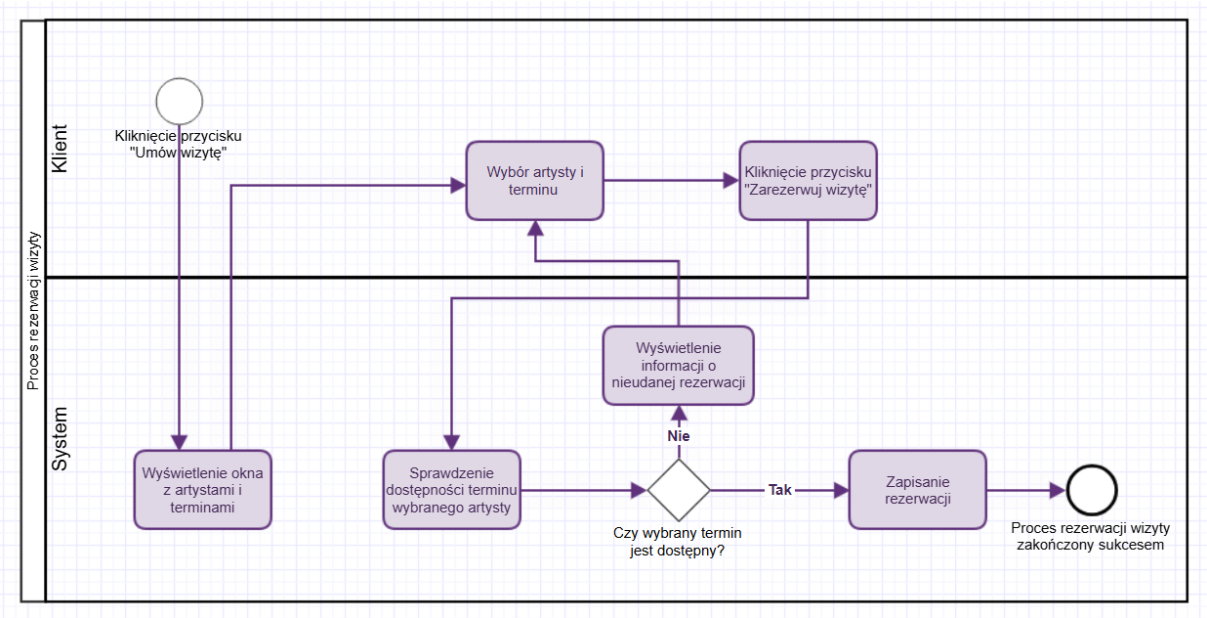

Figure 4: Diagram BPMN - Proces rezerwacji wizyty

Wybrane technologie

Typ aplikacji:

- **Aplikacja webowa:** Aplikacja dostępna przez przeglądarkę internetową, umożliwiająca interakcję zarówno dla użytkowników niezalogowanych, jak i zalogowanych.

Typ bazy danych:

- **H2 Database:** Wbudowana baza danych w Javie, wspierająca SQL.

Frontend:

- **Struktura aplikacji:** HTML5
- **Style/Wygląd aplikacji:** CSS3
- **Interakcje i dynamika aplikacji:** JavaScript

Backend:

- **Główny język programowania:** Java
- **Framework:** Spring Boot
- **System budowania projektu:** Maven

Projekt bazy danych

Opis projektu bazy danych:

Projekt bazy danych został opracowany zgodnie z zasadami normalizacji oraz optymalnego projektowania, aby zapewnić wydajność i spójność przechowywanych danych. Baza danych składa się z następujących encji i relacji, które odzwierciedlają strukturę systemu strony internetowej dla salonu tatuaży.

Encje i ich atrybuty:

- **UŻYTKOWNICY:** Przechowuje dane użytkowników systemu.
  - **Atrybuty:** ID_user, phone, password, username.
- **WIZYTY:** Przechowuje informacje o zarezerwowanych wizytach.
  - **Atrybuty:** ID_appointment, ID_user, ID_artist, date, time.
- **PRACOWNICY:** Przechowuje dane pracowników salonu (artystów tatuażu).
  - **Atrybuty:** ID_artist, name, photo_URL.
- **PORTFOLIO:** Przechowuje informacje o pracach artystów.
  - **Atrybuty:** ID_photo, ID_artist, description.
- **PHOTO:** Przechowuje szczegóły dotyczące plików graficznych w portfolio.
  - **Atrybuty:** ID_photo, photo_URL.
- **DOSTĘPNOŚĆ:** Reprezentuje dostępność artystów w określonych terminach.
  - **Atrybuty:** ID_availability, ID_artist, time, date.
- **KONTAKT:** Przechowuje wiadomości przesłane przez klientów i gości za pomocą formularza kontaktowego.
  - **Atrybuty:** ID_message, message, phone.

Relacje między encjami:

Relacje pomiędzy encjami zostały zaprojektowane zgodnie z zasadą kluczy obcych, aby zachować spójność danych:

- **UŻYTKOWNICY → WIZYTY:** Relacja 1:N (każdy użytkownik może mieć wiele wizyt, każda wizyta przypisana jest do jednego użytkownika).
- **PRACOWNICY → WIZYTY:** Relacja 1:N (każdy pracownik może obsługiwać wiele wizyt, każda wizyta jest przypisana do jednego pracownika).
- **PRACOWNICY → PORTFOLIO:** Relacja 1:N (każdy pracownik może mieć wiele prac w portfolio).
- **PORTFOLIO → PHOTO:** Relacja 1:1 (każde zdjęcie w portfolio odnosi się do jednego rekordu w tabeli PHOTO).
- **PRACOWNICY → DOSTĘPNOŚĆ:** Relacja 1:N (każdy pracownik ma przypisaną dostępność w kalendarzu).

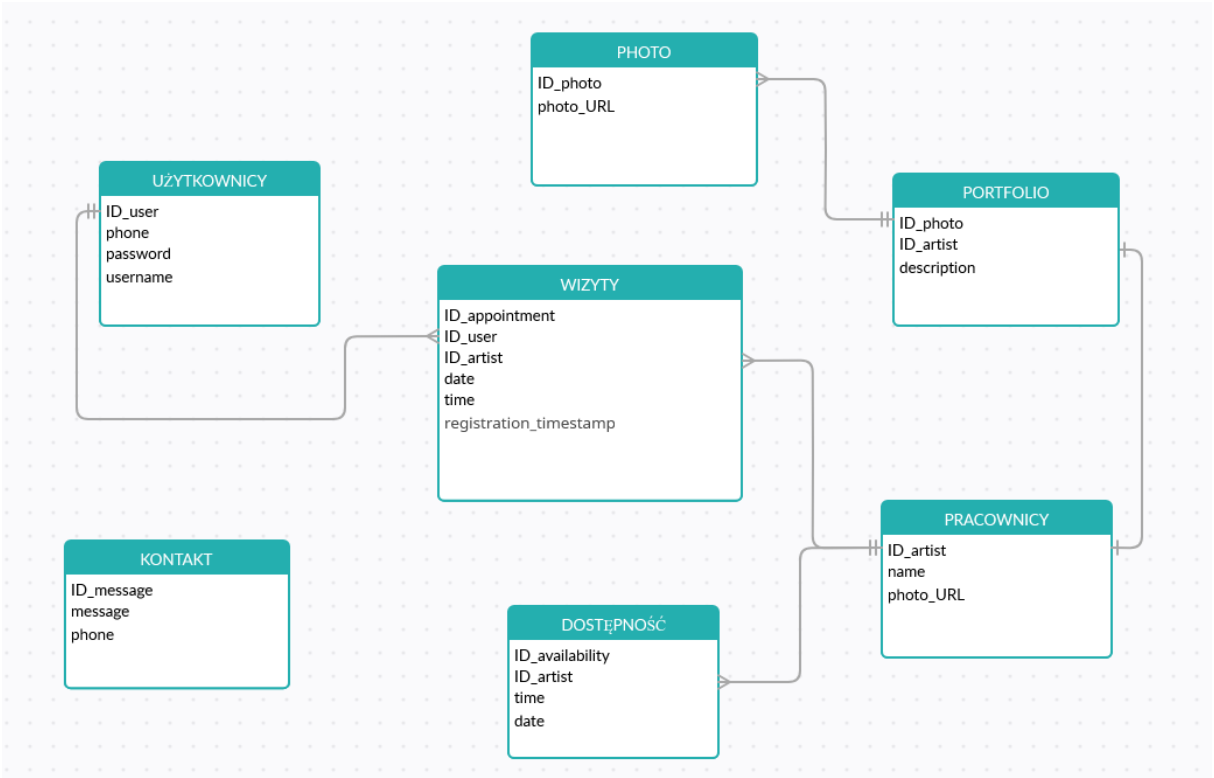

Figure 5: Diagram relacji encji (ERD) dla bazy danych systemu.

Opis transakcji w systemie

Zapis:

- **Dodanie nowego użytkownika (rejestracja):** Po przesłaniu formularza rejestracyjnego dane użytkownika (nazwa użytkownika, hasło, numer telefonu) są zapisywane w tabeli *Użytkownicy*. Weryfikowane są unikalność nazwy użytkownika oraz zgodność formatu hasła.
- **Złożenie rezerwacji:** Po wybraniu artysty i terminu, system zapisuje dane wizyty (ID użytkownika, ID artysty, data, godzina, godzina rezerwacji) w tabeli *Wizyty*. Weryfikowana jest dostępność wybranego terminu w tabeli *Dostępność*.
- **Dodanie nowego artysty:** Administrator zapisuje dane nowego artysty (imię, nazwisko, zdjęcie) w tabeli *Pracownicy*. Dane są następnie powiązywane z tabelą *Portfolio*.
- **Dodanie zdjęcia do portfolio:** Po przesłaniu zdjęcia i przypisaniu go do artysty, informacje o zdjęciu (ścieżka pliku, ID artysty) są zapisywane w tabeli *PHOTO* i powiązane z tabelą *Portfolio*.
- **Wysłanie wiadomości przez formularz kontaktowy:** Dane przesłane w formularzu (wiadomość, numer telefonu) są zapisywane w tabeli *Kontakt*. Każda wiadomość otrzymuje unikalny identyfikator.

Modyfikacja:

- **Aktualizacja portfolio:** Administrator edytuje dane w tabeli *Portfolio*, modyfikując opisy lub zamieniając zdjęcia. Transakcje aktualizują odpowiednie wpisy w tabelach *Portfolio* i *PHOTO*.
- **Aktualizacja dostępności artystów:** Harmonogram pracy artystów jest aktualizowany poprzez dodanie lub usunięcie terminów w tabeli *Dostępność*. Weryfikowane jest powiązanie terminu z ID artysty.

Odczyt:

- **Przeglądanie portfolio:** System pobiera dane z tabeli *Portfolio*, w tym opisy i ścieżki do zdjęć, a następnie wyświetla je użytkownikom.
- **Wyświetlenie dostępnych artystów:** Dane o pracownikach są pobierane z tabeli *Pracownicy* i łączone z danymi z tabeli *Portfolio*, aby przedstawić ich prace.
- **Wyświetlenie rezerwacji użytkownika:** System odczytuje z tabeli *Wizyty* wizyty powiązane z ID użytkownika i wyświetla szczegóły, takie jak data, godzina i artysta.
- **Wyświetlenie kalendarza dostępności artystów:** Terminy dostępności artystów są pobierane z tabeli *Dostępność* na podstawie wybranego ID artysty.
- **Wyświetlenie wiadomości:** Administrator odczytuje dane z tabeli *Kontakt*, aby zarządzać wiadomościami przesłanymi przez formularz.

Usuwanie:

- **Usunięcie konta użytkownika:** Na żądanie klienta administrator usuwa dane użytkownika z tabeli *Użytkownicy*. Dane w tabeli *Wizyty* mogą być utrzymywane w celach archiwalnych, ale są anonimizowane.
- **Anulowanie rezerwacji:** Rezerwacja jest usuwana z tabeli *Wizyty*, a powiązany termin w tabeli *Dostępność* zostaje oznaczony jako dostępny.
- **Usunięcie zdjęcia z portfolio:** Administrator usuwa wpis w tabeli *PHOTO* oraz powiązany wpis w tabeli *Portfolio*.
- **Usunięcie artysty:** Po usunięciu artysty z tabeli *Pracownicy* wszystkie powiązane dane (np. dostępność i portfolio) są również usuwane lub archiwizowane.

Prognoza występowania poszczególnych encji

Użytkownicy:

- **Sposób użycia:** Sporadyczny zapis podczas rejestracji nowych użytkowników (klientów). Odczyt w celu logowania użytkowników.
- **Przewidywana zmienność:** Niska – tylko dodawanie nowych użytkowników. Brak aktualizacji danych osobowych.
- **Przewidywana liczba wystąpień:** Początkowo kilka encji (1-2 menedżerów, 10-50 klientów). Liczba będzie rosła proporcjonalnie do popularności serwisu.

Wizyty:

- **Sposób użycia:** Częsty zapis podczas rezerwacji nowych wizyt przez klientów. Odczyt w celu przeglądania harmonogramu wizyt przez klientów i menedżera.
- **Przewidywana zmienność:** Wysoka – ciągłe dodawanie nowych wizyt i sporadyczne usuwanie.
- **Przewidywana liczba wystąpień:** Zależna od liczby klientów i artystów. Na początek przewiduje się kilkadziesiąt wizyt miesięcznie.

Pracownicy:

- **Sposób użycia:** Częsty odczyt przez gości i klientów w celu przeglądania listy artystów i ich profili. Sporadyczny zapis i modyfikacja przez menedżera.
- **Przewidywana zmienność:** Niska – zmiany zachodzą przy aktualizacji zespołu pracowników.
- **Przewidywana liczba wystąpień:** Stała, zależna od liczby pracowników salonu (np. 5-10 artystów).

Dostępność:

- **Sposób użycia:** Częsty odczyt w celu sprawdzenia dostępnych terminów artystów. Sporadyczny zapis lub modyfikacja przez menedżera podczas aktualizacji harmonogramu.
- **Przewidywana zmienność:** Umiarkowana – zależna od zmian w grafiku pracy artystów.
- **Przewidywana liczba wystąpień:** Zależna od liczby artystów i godzin pracy. Możliwe kilkadziesiąt terminów miesięcznie.

Portfolio:

- **Sposób użycia:** Częsty odczyt w celu przeglądania zdjęć prac artystów. Sporadyczny zapis nowych prac i modyfikacja przez menedżera.
- **Przewidywana zmienność:** Umiarkowana – aktualizacje są zależne od nowych projektów artystów.
- **Przewidywana liczba wystąpień:** Liczba zdjęć proporcjonalna do liczby artystów i ich prac – przewiduje się kilkadziesiąt encji na początek.

Photo:

- **Sposób użycia:** Powiązane z encją PORTFOLIO – używane do przechowywania szczegółowych danych o zdjęciach. Częsty odczyt przez użytkowników.
- **Przewidywana zmienność:** Umiarkowana – dodawanie nowych zdjęć lub usuwanie istniejących.
- **Przewidywana liczba wystąpień:** Kilkadziesiąt do kilkuset zdjęć w zależności od liczby artystów.

Kontakt:

- **Sposób użycia:** Częsty zapis wiadomości wysyłanych przez formularz kontaktowy przez gości i klientów. Sporadyczny odczyt przez menedżera.
- **Przewidywana zmienność:** Umiarkowana – wiadomości są przetwarzane na bieżąco, a starsze mogą być usuwane.
- **Przewidywana liczba wystąpień:** Kilkanaście do kilkudziesięciu wiadomości miesięcznie.

Backend aplikacji

Wprowadzenie

Dokument opisujący endpointy backendu aplikacji webowej. Każdy endpoint zawiera opis jego funkcji, metod HTTP oraz parametrów.

Repozytorium GitHub

Link do repozytorium: https://github.com/267131-2023/Tattoo-FeatherLink

Lista Endpointów

/user-panel/delete-account

Metoda: `POST`
Opis: Usuwa konto użytkownika wraz ze wszystkimi jego wizytami i przywraca dostępne terminy.

/user-panel/book

Metoda: `POST`
Opis: Pozwala użytkownikowi zarezerwować wizytę, wybierając dostępny termin.
Parametry:

[noitemsep]`availabilityId` (`query`, `integer`) - ID dostępnego terminu (wymagany).

/register

Metody: `GET`, `POST`
Opis:

[noitemsep]**GET:** Wyświetla formularz rejestracji, umożliwiający użytkownikowi wprowadzenie danych i utworzenie konta. **POST:** Rejestruje nowego użytkownika, sprawdzając unikalność nazwy użytkownika oraz walidując dane formularza.

Parametry dla POST:

[noitemsep]`user` (`query`, obiekt `User`) - Obiekt reprezentujący dane użytkownika (wymagany).

/authenticateTheUser

Metoda: `POST`
Opis: Uwierzytelnia użytkownika na podstawie wprowadzonej nazwy użytkownika i hasła. Jeśli dane są niepoprawne, wyświetla błąd.
Parametry:

[noitemsep]`username` (`query`, `string`) - Nazwa użytkownika (wymagany). `password` (`query`, `string`) - Hasło użytkownika (wymagany).

/authenticateContact

Metoda: `POST`
Opis: Weryfikuje dane kontaktowe (np. numer telefonu) i zapisuje je w systemie. Jeżeli numer telefonu jest pusty, wyświetli błąd.
Parametry:

[noitemsep]`contact` (`query`, `obiekt Contact`) - Obiekt reprezentujący dane kontaktowe (wymagany).

/admin-panel/delete-portfolio

Metoda: `POST`
Opis: Usuwa portfolio na podstawie jego ID.
Parametry:

[noitemsep]`portfolioId` (`query`, `integer`) - ID portfolio (wymagany).

/admin-panel/delete-message

Metoda: `POST`
Opis: Usuwa wiadomość kontaktową na podstawie jej ID.
Parametry:

[noitemsep]`messageId` (`query`, `integer`) - ID wiadomości (wymagany).

/admin-panel/delete-availability

Metoda: `POST`
Opis: Usuwa dostępność artysty na podstawie ID dostępności.
Parametry:

[noitemsep]`availabilityId` (`query`, `integer`) - ID dostępności (wymagany).

/admin-panel/delete-artist

Metoda: `POST`
Opis: Usuwa artystę na podstawie jego nazwy.
Parametry:

[noitemsep]`artistName` (`query`, `string`) - Nazwa artysty (wymagany).

/admin-panel/delete-appointment

Metoda: `POST`
Opis: Usuwa wizytę na podstawie jej ID.
Parametry:

[noitemsep]`appointmentId` (`query`, `integer`) - ID wizyty (wymagany).

/admin-panel/add-portfolio

Metoda: `POST`
Opis: Dodaje nowe portfolio dla artysty.
Parametry:

[noitemsep]`photoURL` (`query`, `string`) - URL zdjęcia portfolio (wymagany). `artistId` (`query`, `integer`) - ID artysty (wymagany).

/admin-panel/add-availability

Metoda: `POST`
Opis: "Dodaje nową dostępność artysty w systemie..
Parametry:

[noitemsep]`date` (`query`, `string`) - Data dostępności (wymagany). `time` (`query`, `string`) - Czas dostępności (wymagany). `artistId` (`query`, `integer`) - ID artysty (wymagany).

/admin-panel/add-artist

Metoda: `POST`
Opis: Dodaje nowego artystę do systemu.
Parametry:

[noitemsep]`name` (`query`, `string`) - Imię artysty (wymagany). `photoURL` (`query`, `string`) - URL zdjęcia artysty (wymagany). `description` (`query`, `string`) - Opis artysty (wymagany).

/admin-panel

Metoda: `GET`
Opis: Wyświetla panel administracyjny tylko dla użytkowników z rolą admin.

/admin-panel/portfolio

Metoda: `GET`
Opis: Wyświetla wszystkie portfolia artystów.

/admin-panel/contact

Metoda: `GET`
Opis: Wyświetla wszystkie wiadomości kontaktowe.

/admin-panel/availability

Metoda: `GET`
Opis: Wyświetla dostępność wszystkich artystów.

/admin-panel/artist

Metoda: `GET`
Opis: Wyświetla wszystkich artystów w systemie.

/admin-panel/appointment

Metoda: `GET`
Opis: Wyświetla wszystkie zaplanowane wizyty.

/

Metoda: `GET`
Opis: Wyświetla stronę główną aplikacji.

/works

Metoda: `GET`
Opis: Wyświetla stronę z pracami (portfolio).

/price

Metoda: `GET`
Opis: Wyświetla stronę z cennikiem usług.

/user-panel

Metoda: `GET`
Opis: Wyświetla panel użytkownika, w którym można zobaczyć swoje umówione wizyty oraz dostępne terminy.

/portfolio

Metoda: `GET`
Opis: Wyświetla wszystkie dostępne zdjęcia lub projekty w portfolio.

/logout

Metoda: `GET`
Opis: Wylogowuje użytkownika, usuwając jego sesję.

/login

Metoda: `GET`
Opis: Wyświetla stronę logowania dla użytkownika, umożliwiając mu wprowadzenie danych uwierzytelniających.

/contact

Metoda: `GET`
Opis: Wyświetla formularz kontaktowy, który umożliwia użytkownikowi wypełnienie swoich danych kontaktowych.

/artists

Metoda: `GET`
Opis: Wyświetla wszystkich artystów w systemie.

/appointments

Metoda: `GET`
Opis: Wyświetla wszystkie zaplanowane wizyty użytkowników.

Frontend Aplikacji

Technologie

Frontend aplikacji opiera się na:

- **HTML** - struktura strony.
- **CSS** - stylizacja i układ wizualny.
- **Thymeleaf** - silnik szablonów dla integracji frontendu z backendem (Spring Boot).

HTML

- Podzielony na logiczne sekcje za pomocą tagów, takich jak `<header>`, `<main>`, `<footer>` oraz kontenerów `<div>`.
- Formularze (np. rejestracja, logowanie) obsługują dane użytkownika.
- Wykorzystują walidację po stronie serwera i klienta, aby zapewnić poprawność wprowadzonych danych.

CSS

- Pliki: `user.css` (dla użytkowników) i `admin.css` (dla administratorów).
- Zapewniają responsywność i estetykę (np. style przycisków, tabele, layout galerii).

Przykłady:

- **Przyciski:** Estetyczne style z efektami `hover`.
- **Układ galerii:** Zdjęcia w rzędach, automatycznie dopasowywane do szerokości strony.

Thymeleaf

- Integruje dane z backendu w czasie generowania strony.
- Umożliwia dynamiczne wyświetlanie treści:
  - Iteracje: `th:each` (np. lista artystów).
  - Powiązanie danych: `th:field`, `th:text` (np. dane użytkownika w formularzach).
  - Dynamiczne linki: `th:href`.

Walidacja formularzy

- Spring Boot + Thymeleaf obsługują walidację pól wejściowych.
- Atrybuty Thymeleaf pozwalają na dynamiczne wyświetlanie błędów:
  - `th:field` - powiązanie pola formularza z danymi użytkownika.
  - `fields.hasErrors` - sprawdza, czy pole zawiera błędy.
  - `th:errors` - wyświetla komunikaty o błędach.

Działanie:

- Jeśli użytkownik wprowadzi niepoprawne dane (np. puste pole, zły format hasła), `@Valid` oraz `BindingResult` sprawdzą poprawność.
- Jeśli wystąpią błędy, użytkownik zostaje odesłany z powrotem do formularza, a błędy pojawiają się obok pól.
- Jeśli dane są poprawne, użytkownik zostaje zapisany do bazy i przekierowany na stronę sukcesu.