Katowice, 12.06.2024 r.

Raport końcowy

Tutorino

**Skład sekcji:**

Artmański Tomasz

Cichosz Filip

Krzywoń Jakub

Legierski Sebastian

Miozga Marta **Prowadzący:**

Ulman Artur Mgr inż Marcin Wierzchanowski

1. **Wstępna Specyfikacja Systemu**

**Temat:**

„Tutorino - System dla Korepetytorów do zarządzania zajęciami, uczniami i płatnościami.”

**Zamysł projektu:**

Zamysłem projektu jest stworzenie systemu oraz aplikacji umożliwiającej Korepetytorom oraz Uczniom zarządzanie zajęciami, płatnościami i pomóc w organizacji.

**Funkcjonalności:**

* Wyświetlanie dostępnych w systemie korepetytorów i oferowanych przez nich korepetycji.
* Możliwość organizacji, umawiania, przenoszenia i odwoływania zajęć.
* Zarządzanie finansami per zajęcia bądź jako nadpłaty lub płatności skumulowane.
* Zarządzanie własnym kontem, jeżeli konto operatora też powiązanymi kontami uczniów

**Założenia:**

* Pomoc w znajdowaniu uczniów dla korepetytorów.
* Pomoc korepetytorom w znajdowaniu uczniów.
* Pomoc korepetytorom w zarządzaniu opłatami za zajęcia.
* Pomoc operatorom (rodzicom, uczniom samodzielnym finansowo) w zarządzaniu należnościami i prostym wykonywaniu płatności.

**Każdy użytkownik:**

Może edytować swój profil, dane, zdjęcie profilowe. Ma wgląd do swoich danych oraz statystyki pokroju: ilość przeprowadzonych / odbytych korepetycji (swoich, bądź w przypadku konta klasy operator, powiązanych kąt uczniów), oceny czy status konta.

**Korepetytorzy:**

Ponad możliwości wszystkich użytkowników, mogą również zmieniać deklarację korepetycji jakie oferują, które będą wyświetlane na ich profilu publicznie, aby pomóc się odnaleźć nowym uczniom. Korepetytor będzie miał również możliwość przełączyć swój profil na niepubliczny/prywatny, kiedy nie jest już zainteresowany udzielaniem korepetycji nowym uczniom bądź jego terminarz jest przepełniony. Korepetytor jest osobą zarządzającą w największym stopniu korepetycjami, które prowadzi. Korepetytor ma również dostęp do ewidencji płatności ze strony operatorów. Korepetytor może zarejestrować płatności per zajęcia lub jako nadpłata operatora wykonane gotówką lub poza systemem.

**Operator:**

Z zamysłu użytkownik typu operator jest tylko odpowiedzialny za finanse i ewentualnie umawianie zajęć dla podpiętego do danego operatora ucznia. Jeden operator może być powiązany z wieloma kontami klasy Uczeń, kiedy przykładowo rodzic wyprawia parę swoich dzieci na korepetycje. Operator ma wgląd w kalendarz powiązanych uczniów, widzi zaplanowane i odbyte zajęcia, to operatorzy mają możliwość wykonania płatności za zajęcia jednym kliknięciem prosto ze strony (lub z przekierowania do strony typu TPay).

**Uczeń:**

Uczeń w odróżnieniu do operatora nie ma wglądu do finansów, jedynie do swojego kalendarza, czyli widzi zaplanowane i odbyte korepetycje. Uczeń również widzi informacje powiązane z zajęciami jak tematyka, adres bądź link do spotkania online. System przewiduje, że jedno konto może posiadać wiele klas, nawet wszystkie naraz.

**MUST**

* System kont​
* Rejestracja​
* Logowanie​
* Zarządzanie Kontem​
* Sesja Logowania​
* Wystawianie Ofert Przez Korepetytorów​
* Przeglądanie ofert​​
* Umawianie Korepetycji​
* Anulowanie Korepetycji​
* Płacenie za zajęcia w systemie
* Szczegóły Zajęć​
* Online / kontaktowe​
* Address students / korepetytora​
* Link do spotkania / workspace​
* Możliwość udostępniania swojego profilu poprzez link

**SHOULD**

* Przenoszenie Korepetycji​
* System Ocen​
* Ocenianie Korepetytorów​
* Ocenianie Uczniów​
* Wyświetlane Ocen na profilu​
* Przypominanie o niezapłaconych zajęciach​​
* Panel Administracyjny​
* Wiele języków​
* Analiza Danych i Raportowanie​

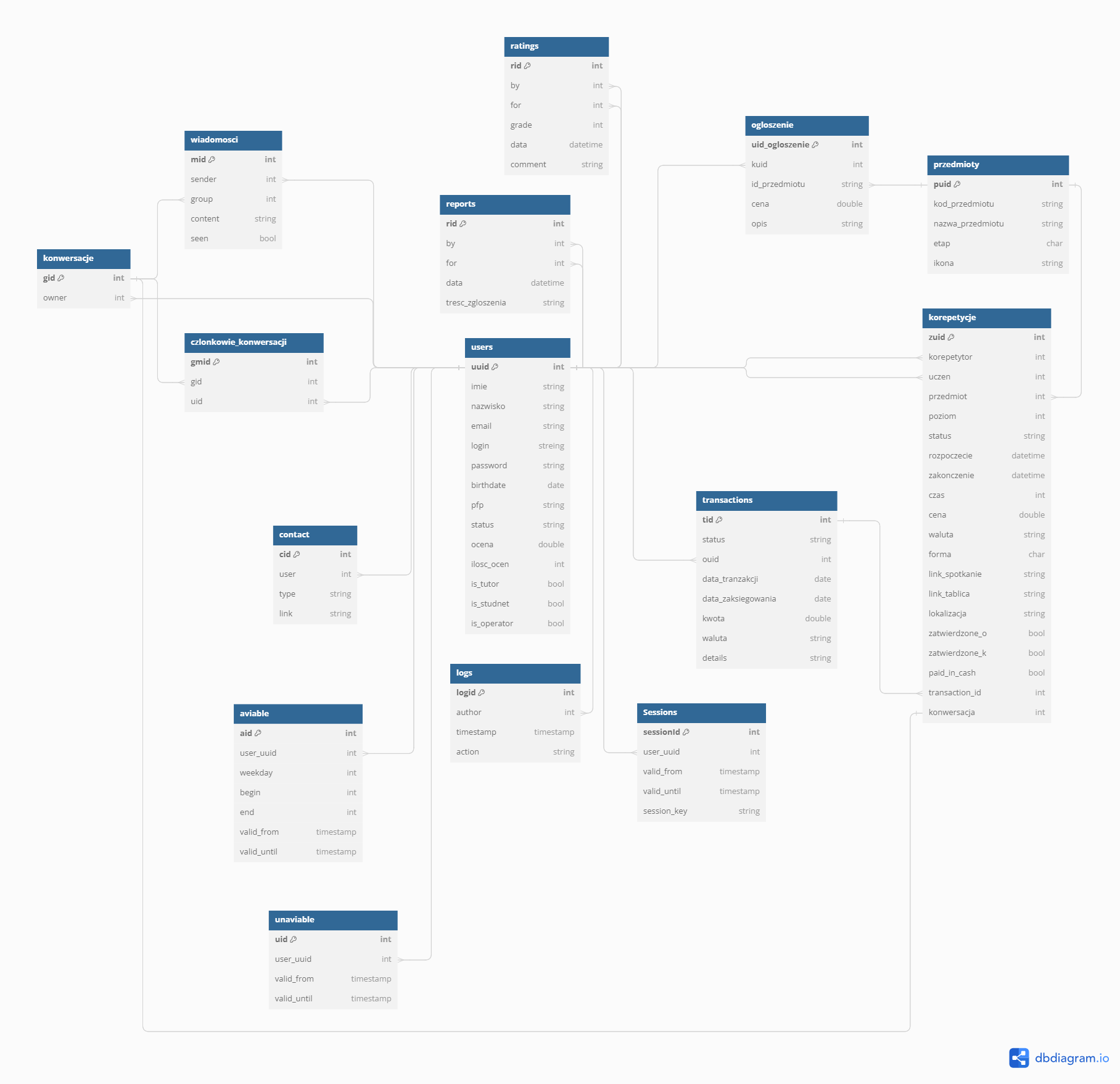
**COULD**

* Wyświetlanie Statystyk Konta​
* Powiadomienia Push​
* Zabezpieczenie API kluczem​
* Płatności za wiele zajęć naraz​
* Dark/Light Mode​
* Płatna Subskrypcja wyróżniająca oferty korepetytora​
* Sugerowanie podobnych ofert​​
* Integracja z Mediami społecznościowymi​
* System rekomendacji korepetytor - korepetytor​
* Integracja z systemem płatności mobilnych​
* Harmonogramowanie lekcji grupowych​
* Możliwość publikowania i udostępniania materiałów​

**WANT**

* Możliwość Zmiany Koloru UI​
* Płacenie za wielu uczniów na raz​
* Zbieranie punktów za odbyte zajęcia​
* Funkcja napiwku​
* Zgłaszanie Użytkowników za naruszenia​
* Komunikator – wbudowany system wiadomości​
* Automatyczne zakładanie spotkania Google Meets​

1. **Schemat bazy danych:**



1. **Implementacja i Testy:**

Implementacja projektu, opiera się na klasycznej trójwarstwowej architekturze, gdzie frontend, backend i baza danych są oddzielnie zaimplementowane.

Poniżej znajdziesz podsumowanie każdej z tych warstw

**Frontend (HTML, CSS, JS):**

* Frontend jest zrealizowany przy użyciu standardowych technologii webowych: HTML, CSS i JavaScript.
* HTML odpowiada za strukturę strony, definiując jej elementy i układ.
* CSS jest wykorzystywany do stylizacji, nadając estetyczny wygląd elementom HTML.
* JavaScript odpowiada za interakcję użytkownika i dynamiczne zmiany na stronie. Może komunikować się z backendem poprzez żądania HTTP do API.

**Backend ( C#):**

* API obsługuje żądania przychodzące z frontendu, wykonując operacje na bazie danych i zwracając odpowiedzi.
* Backend może również implementować autentykację i kontrolę dostępu, aby zapewnić bezpieczeństwo aplikacji.

**Baza Danych (SQL):**

* Baza danych, oparta na SQL, przechowuje dane używane przez aplikację.
* Schemat bazy danych jest dostosowany do potrzeb projektu, z uwzględnieniem relacji między różnymi tabelami.
* Backend wykonuje zapytania SQL do bazy danych w celu pobierania, aktualizacji lub usuwania danych.

**Serwer (C#, prywatny serwer):**

* Cała aplikacja jest hostowana na prywatnym serwerze, co oznacza, że jest dostępna w sieci, dla każdego użytkownika, który odpowiednio wywoła zapytanie.
* Serwer odpowiada za nasłuchiwanie na przychodzące żądania HTTP, przekazywanie ich do backendu, a następnie przekazywanie odpowiedzi z powrotem do frontendu.

**Testowanie (Python):**

* Do testowania API został przygotowany osobny skrypt napisany w języku Python.
* Testy te mogą obejmować sprawdzanie poprawności odpowiedzi API, obsługę błędów, a także testy jednostkowe funkcji backendu.
* Testowanie odgrywa kluczową rolę w zapewnianiu niezawodności i bezpieczeństwa aplikacji.

1. **Role projektowe:**

| **Artmański Tomasz** | Implementacja Bazy Danych, Zarządzanie serwerem, backend, konfiguracja bazy danych |
| --- | --- |
| **Filip Cichosz** | Backend, Projekt Bazy danych, Edycja testów Jednostkowych, dokumentacja/raport |
| **Marta Miozga** | Pomoc w backend, dokumentacja/ raport |
| **Sebastian Legierski** | Frontend, Prezentacje |
| **Jakub Krzywoń** | Testowanie, Backend, Prezentacje |
| **Artur Ulman** | Pomoc w backend, raport |

1. **Wnioski końcowe:**

Projekt grupowy nad aplikacją Tutorino został zrealizowany zgodnie z podstawowymi założeniami użytkowymi projektu. Udało się stworzyć kompleksowe rozwiązanie, które wspiera proces umawiania korepetycji przez serwis internetowy. Efektywna współpraca w zespole była kluczowym elementem projektu. Podział obowiązków i spotkania pomogły w utrzymaniu postępu prac. Wykorzystanie różnorodnych technologii, takich jak C# w API (Backend), HTML, CSS, JavaScript, JSON data (Frontend), oraz bazy danych, pozwoliło na stworzenie funkcjonalnych elementów aplikacji. Równocześnie było to wyzwanie związane z integracją tych komponentów. Przygotowanie założeń projektu oraz schematu bazy danych były kluczowymi narzędziami dokumentacyjnymi. Umożliwiły one zrozumienie architektury systemu oraz były przydatne w procesie komunikacji zespołowej. Proces testowania był istotny dla zapewnienia jakości aplikacji. Testy jednostkowe pozwoliły na identyfikację i naprawę błędów, co wpłynęło na działanie systemu. Realizacja projektu pozwoliła na zdobycie cennych doświadczeń związanych z zarządzaniem projektem.