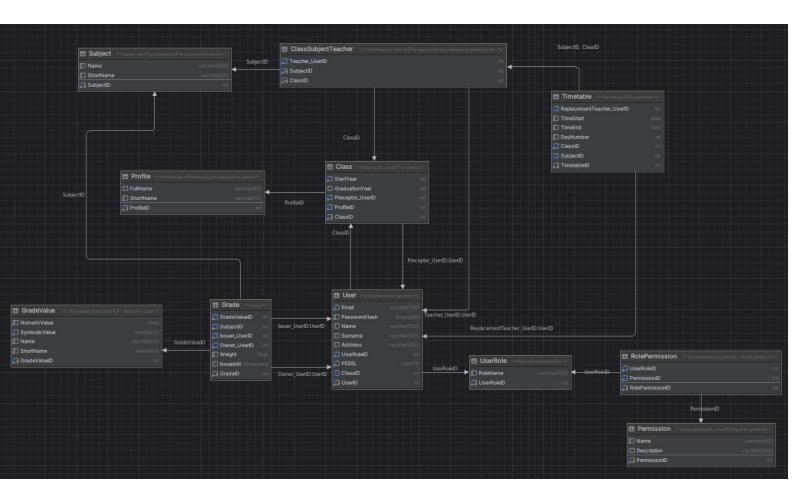
| Wydział | Imię i nazwisko | Projekt nr 1 | Kieurnek |
|---|-------------------------------------|--------------|------------------------|
| WIMilP | 1. Maciej Adamus | SQL | Informatyka techniczna |
| | 2. Kacper Bielak | | |
| Temat projektu: System bazodanowy do obsługi dziennika elektronicznego | | | |
| Data wykonania | Przedmiot: Grupa | | |
| 14.05.2024r. | Zawansowane architektury baz danych | | GL01 |

1. Wstęp i cel projektu

Celem pierwszego projektu na przedmiot Zawansowane architektury baz danych było zaimplementowanie systemu bazodanowego do obsługi dziennika elektronicznego w szkole. Baza danych stworzona została w języku programowania SQL w systemie do zarządzania relacyjnymi bazami danych MySQL. System ma na celu umożliwienie zarządzanie cyklem nauczania w szkole podstawowej, które może obejmować szereg różnych operacji, m.in. operację na uczniach, nauczycielach, klasach, przedmiotach, zajęciach czy ocenach.

2. Struktura systemu bazodanowego



Struktura zaimplementowanego systemu bazodanowego przedstawiona została na powyższym diagramie ERD.

Poniżej znajduję się opis wszystkich tabel wchodzących w skład systemu i widocznych na powyższym diagramie ERD:

- *User* reprezentuję użytkowników w systemie takich jak np. uczniowie, nauczyciele czy dyrektor, którzy są odróżniani dzięki relacji z tabelą *UserRole*, w przypadku ucznia posiada uzupełnione pole *ClassID* wskazujące na przynależność do danej klasy;
- UserRole wskazuję role użytkownika, ma do siebie przypisany obiekt z tabeli RolePermission;
- RolePermission zawiera zestaw pozwoleń (obiektów Permission) dla danej roli;
- *Permission* pojedyncza rola;
- Class reprezentuję klasę, posiada uczniów oraz wychowawcę, profil klasy wskazuje relacja z klasą Profile;
- *Profile* reprezentuję profile klas;
- Timetable plan zajęć dla poszczególnych przedmiotów prowadzonych przez danego nauczyciela (relacja z tabelą *ClassSubjectTeacher*), zawiera również pole odpowiadające za zastępstwo nauczyciela (jeśli takie jest wymagane);
- ClassSubjectTeacher tabela asocjacyjna wiążąca ze sobą przedmiot, nauczyciela i klasę, wykorzystana w głównej mierze w planie zajęć (*Timetable*);
- Subject reprezentuję przedmiot;
- Grade reprezentuje ocenę dla danego ucznia z danego przedmiotu wystawioną przez danego nauczyciela;
- *GradeValue* reprezentuje stopnie, jest w relacji z Grade i została od niej odseparowana ze względów na możliwość uzyskania ocen takich jak np. 5-.

W ramach systemu stworzony został również szereg funkcji, procedur, wyzwalaczy i widoków, które zostały opisane poniżej.

Procedury

- Procedura nr 1: pozwala na sprawdzenie czy dana klasa ukończyła już szkołę wykorzystana w wyzwalaczu nr 1, 4.
- 2. Procedura nr 2: Sprawdza czy dana klasa w danym dniu i danej godzinie ma już zajęcia wykorzystana w wyzwalaczu nr 2.

- 3. Procedura nr 3: wyliczenie średniej ocen klasy.
- 4. Procedura nr 4: wyznaczenie planu lekcji dla klasy.
- 5. Procedura nr 5: wyliczenie średniej ocen ucznia.
- 6. Procedura nr 6: wyliczenie średniej ocen ucznia dla poszczególnych przedmiotów.

<u>Funkcje</u>

- 1. Funkcja nr 1: weryfikuje czy dany użytkownik jest studentem wykorzystana w wyzwalaczu nr 6, 7, 8, 9.
- 2. Funkcja nr 2: weryfikuje czy dany użytkownik jest nauczycielem i czy ma już wychowawstwo wykorzystana w wyzwalaczu nr 12, 13.
- 3. Funkcja nr 3: wyznacza liczbę uczniów w klasie wykorzystana w wyzwalaczu nr 14, 15.

Wyzwalacze (trigery)

- Triger nr 1 before add user (uczen): weryfikuje czy klasa do której ma trafić uczeń nie zakończyła już edukacji.
- 2. Triger nr 2 before add timetable: weryfikuje czy nowe zajęcia nie zachodzą na już istniejące.
- 3. Triger nr 3 after add class: wyznacza rok ukończenia szkoły dla tej klasy.
- 4. Triger nr 4 before update user (uczen): weryfikuje czy klasa do której ma trafić uczeń nie zakończyła już edukacji.
- 5. Triger nr 5 before update class: wyznacza rok ukończenia szkoły dla tej klasy.
- 6. Triger nr 6 before add grade: weryfikuje czy user dodający ocenę na pewno nie jest uczniem.
- 7. Triger nr 7 before update grade: weryfikuje czy user dodający ocenę na pewno nie jest uczniem.
- 8. Triger nr 8 before add grade: weryfikuje czy user otrzymujący ocenę na pewno jest uczniem.
- 9. Triger nr 9 before update grade: weryfikuje czy user otrzymujący ocenę na pewno jest uczniem.
- 10. Triger nr 10 before add grade : weryfikuje czy nauczyciel wystawiający ocenę na pewno naucza ucznia otrzymującego ocenę przedmiotu z którego ocena jest wystawiana.

- 11. Triger nr 11 before update grade : weryfikuje czy nauczyciel wystawiający ocenę na pewno naucza ucznia otrzymującego ocenę przedmiotu z którego ocena jest wystawiana.
- 12. Triger nr 12 before add class: weryfikuję czy użytkownik, który ma być wychować na pewno jest nauczycielem i czy nie ma już wychowawstwa.
- 13. Triger nr 13 before update class: weryfikuję czy użytkownik, który ma być wychować na pewno jest nauczycielem i czy nie ma już wychowawstwa.
- 14. Triger nr 14 before add user (uczen): weryfikuje czy klasa do której ma trafić uczeń nie jest pełna.
- 15. Triger nr 15 before update user (uczen): weryfikuje czy klasa do której ma trafić uczeń nie jest pełna.

Widoki

- 1. Widok nr 1: lista ocen danego ucznia.
- 2. Widok nr 2: lista ocen klasy z danego przedmiotu (dziennik).
- 3. Widok nr 3: plan lekcji dla danego użytkownika.
- 4. Widok nr 4: top 10 uczniów z najlepszą w szkole średnią łączną ze swoich przedmiotów.
- 5. Widok nr 5: top 10 klas z najlepszą łączną średnią ocen w szkole.
- 6. Widok nr 6: lista uczniów klasy, którym wychodzą zagrożenia, z listą przedmiotów zagrożonych.
- 7. Widok nr 7: lista uczniów kwalifikujących się do stypendium/świadectwa z paskiem za średnią ocen.
- 8. Widok nr 8: lista klas które uczy dany nauczyciel, z przedmiotami których tam uczy.
- 9. Widok nr 9: lista klas z wychowawcami i rokiem szkoły.
- 10. Widok nr 10: lista wszystkich ocen z wartościami i nauczycielem wystawiającym ocenę.

3. Implementacja systemu – kod źródłowy, wykorzystane narzędzia

System bazodanowy zaimplementowany został w relacyjne bazie danych MySQL w języku SQL. Przy implementacji wykorzystano system kontroli wersji GitHub w którym znajduje się całościowy kod źródłowy systemu wraz z jego opisem. Ponadto przy implementacji wykorzystano IDE Microsoft Visual Studio Code oraz gotowe narzędzie do zarządzania bazą danych Workbench. Link do repozytorium z implementacją znajduje się poniżej:

Repozytorium GitHub

Proces uruchomieniowy systemu został zautomatyzowano dzięki wykorzystaniu oprogramowania Docker, które jest open-source'owe i służy do konteneryzacji. Do tego celu stworzono plik *docker-compose.yml* zawierający konfigurację serwera bazodanowego i służący do jego uruchomienia na podstawie zbudowanego obrazu. Sam obraz natomiast jest tworzony dzięki plikowi *Dockerfile*, który odpowiada za stworzenie obrazu na podstawie obrazu mysql. Ponadto tworzy on zbiorcze pliki zawierające wszystkie instrukcje SQL'owe tworzące system. Są one przechowywane w katalogach *database* oraz *database/import*.

4. Część zaimplementowana do prezentacji systemu

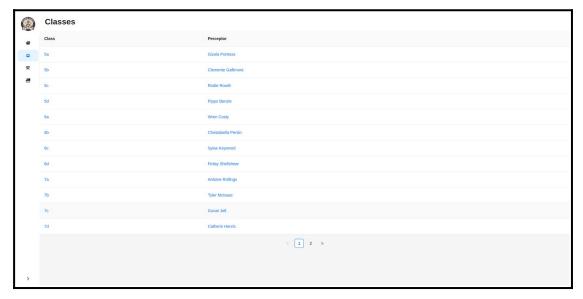
W ramach prezentacji stworzonego systemu bazodanowego zaimplementowane aplikacje webową w język TypeScript z wykorzystaniem frameworka React. Powstałe oprogramowanie pozwala przeprowadzić przykładowe operację na bazie danych, a także umożliwia prezentację danych wraz z uwzględnieniem zależności między nimi.

Poniżej przedstawiono zrzuty ekranu z aplikacji wraz z opisem funkcjonalności:

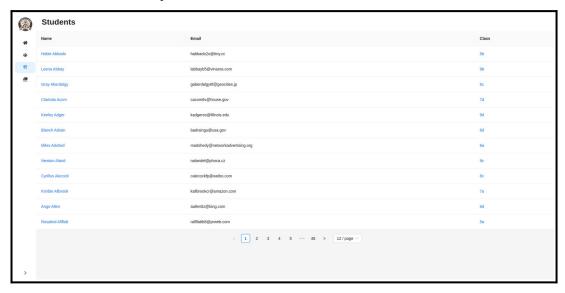
 Główny widok z listą top 10 uczniów, top 10 klas, uczniów zagrożonych i uczniów starających się o stypendium



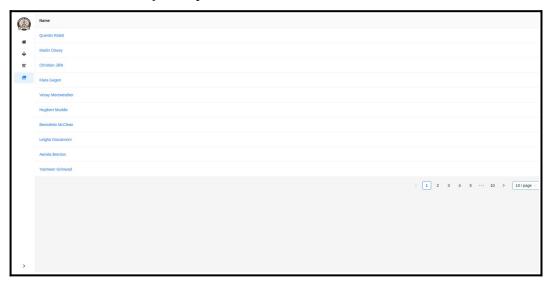
• Widok z listą klas:



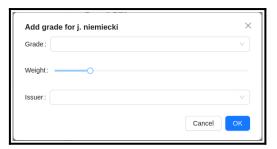
• Widok z listą studentów:

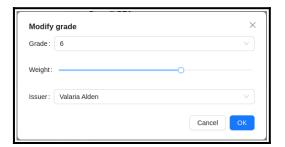


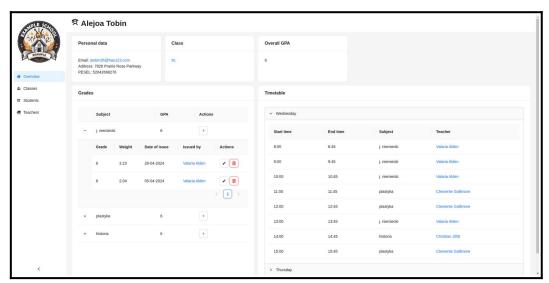
• Widok z listą nauczycieli:



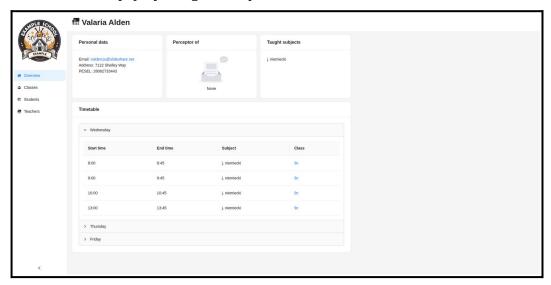
 Widok pojedynczego studenta oraz możliwość dodania nowej oceny, usunięcia lub edycji istniejącej:



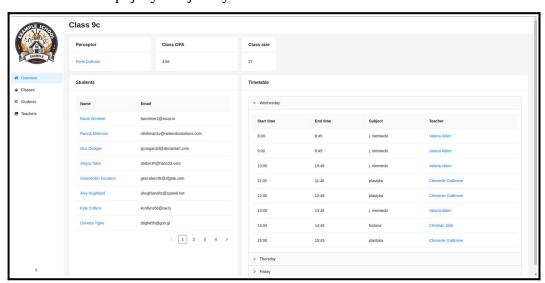




• Widok pojedynczego nauczyciela:



Widok pojedynczej klasy:



5. Wnioski i możliwości dalszego rozwoju

Zaimplementowany system bazodanowy do obsługi dziennika elektronicznego w szkole gwarantuje możliwość zarządzania cyklem nauczania. Stworzone tabele pozwalają przechowywać dane o uczniach, nauczycielach, klasach, planie zajęć czy ocenach. Zaimplementowane procedury i funkcje zdecydowanie mogą pomóc przy implementacji całościowego systemu pełniącego rolę dziennika elektronicznego (np. aplikacja webowa czy mobilna). Wyzwalacze, które zostały stworzone pozwolą natomiast na uniknięcie potencjalnych błędów przy wykonywaniu różnych operacji na bazie danych, np. zbyt duże klasy, błędne wystawianie ocen itp. Ponadto możliwość korzystania z zapisanych w systemie widokach zdecydowanie przyśpieszy implementację, a także i samo działanie wykonywania kolejnych zapytań wymaganych w dzienniku elektronicznym.

Wybór relacyjnej bazy danych zamiast nierelacyjnej był dobrym krokiem przy jej projektowaniu. Z racji dużej ilości powiązań w projekcie dziennika elektronicznego nierelacyjny silnik bazodanowy mógłby okazać się mniej efektywny. Ponadto w przypadku samej implementacji systemu w przyszłości ten wybór może zdecydowanie przyśpieszyć i ułatwić ten proces. Dokonując wybór tego typu bazy kierowano się również możliwością tworzenia wyzwalaczy i widoków co może zmniejszyć ilość pracy przy implementacji końcowego systemu.

Zaimplementowana część do prezentacji napisana w języku TypeScript z wykorzystaniem frameworka React pomaga w zobrazowaniu działania stworzonego systemu bazodanowego. Operację, która ona oferuje (wyświetlania, dodawanie, modyfikacje czy usuwanie danych) pomagają w lepszy sposób przedstawić jego strukturę i przykładowe użycie.

Jak już wspomniano wcześniej, stworzony system bazodanowy może zostać wykorzystany w końcowym systemie pełniącym rolę dziennika elektronicznego. Ponadto zaimplementowana część do prezentacji może być jego punktem wejściowym. Jeśli chodzi o samą bazę danych to może okazać się przydatne jego rozszerzenie o funkcjonalność sprawdzanie obecności uczniów. Do tego celu należałoby wprowadzić nową tabelę przechowująca te dane i będącą w relacji z tabelami *User* oraz *Timetable*.