

POLITECHNIKA RZESZOWSKA
IM. IGNACEGO ŁUKASIEWICZA

WYDZIAŁ MATEMATYKI I FIZYKI STOSOWANEJ

Bazy Danych | *Projekt*
System organizacji studenta

Albert Gawin

Patryk Kępa

Vitalii Morskyi

L4 | 2FS-DI | 166731

Rzeszów, 2021

Spis treści

1 Dokumentacja wstępna	2
1.1 Opis projektu	2
1.2 Prace nad botem i systemem zarządzającym	2
1.3 Planowane widoki	2
2 Projekt konceptualny	4
2.1 Sformułowanie zadania projektowego	4
2.2 Analiza stanu wyjściowego	4
2.3 Analiza wymagań użytkownika	4
2.4 Określenie scenariuszy użycia	5
2.5 Identyfikacja funkcji	7
2.6 Propozycja encji, ich atrybutów i powiązań	7
3 Projekt logiczny	10
3.1 Przejście na model relacyjny	10
3.2 Analiza zależności funkcyjnych	10
3.2.1 Terms	11
3.2.2 Courses	12
3.2.3 Group types	13
3.2.4 USOS units	14
3.2.5 Unit groups	15
3.2.6 Rooms	16
3.2.7 Chats	17
3.2.8 Permissions	18
3.2.9 Teachers	19
3.2.10 Activities	20
3.2.11 Users	21
3.2.12 Study programmes	22
3.2.13 Log activities	23
3.2.14 Log tests	24
3.2.15 Log homeworks	25
3.2.16 Course manager (tabela asocjacyjna)	26
3.2.17 Group teacher (tabela asocjacyjna)	26
3.2.18 Activities per test (tabela asocjacyjna)	27
3.2.19 Users groups (tabela asocjacyjna)	27
3.2.20 User programme (tabela asocjacyjna)	28
3.3 Model relacyjny po normalizacji	28
3.4 Operacje na danych w notacji algebry relacyjnej	28
4 Projekt implementacyjny	32
4.1 Tworzenie tabel	32
4.2 Uzupełnianie tabel danymi	32
4.3 Tworzenie kwerend w języku PL/pgSQL	33
4.4 Przykłady działania aplikacji bazodanowej	39
4.5 Doświadczenia, wynikające z realizacji projektu	41
5 Bibliografia	42

1 Dokumentacja wstępna

1.1 Opis projektu

Celem projektu jest stworzenie systemu umożliwiającego studentom na wymianę między sobą informacji na temat zajęć np. notatek, przewidywanych zaliczeń itp.

Aplikacja ułatwi użytkownikom dzielenie się materiałami, przy pomocy bezpośredniej integracji z popularną media społecznościową – Telegram [6]. Dzięki temu będzie można ominąć problemy związane z wprowadzaniem serwisu w formie strony internetowej bądź aplikacji mobilnej – dodatkowe kroki, które mogą sprawić, że użytkownik może być mniej chętny korzystać z takiego rozwiązania.

1.2 Prace nad botem i systemem zarządzającym

W obrębie tego zadania:

- Przygotowanie bota na platformie Telegram, który będzie posiadał następującą funkcjonalność:
 - rejestracja użytkownika w BD;
 - weryfikacja uczelni użytkownika;
 - regulacja uprawnień poszczególnych użytkowników przez administratorów;
 - dodanie nowych spotkań, zadań domowych i testów (dalej - elementów kształcenia) przez użytkowników do BD;
 - weryfikacja nowych elementów kształcenia przez administratorów;
 - wyświetlenie elementów kształcenia i rozkładu zajęć na najbliższy dzień, tydzień, miesiąc oraz na cały semestr;
 - możliwość komunikacji z użytkownikami zarówno jak przez prywatny czat, tak i w grupie;
 - zasoby ubezpieczenia przed spamowaniem oraz atakami DDoS;
 - pobieranie i zapisywanie danych o częstotliwości użycia bota, jakości doświadczenia użytkownika, zrozumiałości interfejsu wizualnego.
- Przygotowanie infrastruktury na platformie Telegram, a mianowicie:
 - czat dla każdej grupy wykładowej, ćwiczeniowej i laboratoryjnej (projektowej).
 - czaty dla administratorów i wykładowców.

1.3 Planowane widoki

- Widok **administratora** serwisu: zarządzanie uprawnieniami użytkowników, podejmowanie decyzji przy dodaniu nowych elementów kształcenia, możliwość edycji już stworzonych elementów kształcenia, możliwość edycji rozkładu zajęć, podglądy i raporty dotyczące jakości doświadczenia użytkowników.

- Widok **moderatora** serwisu: podejmowanie decyzji przy dodaniu nowych elementów kształcenia, możliwość edycji już stworzonych elementów kształcenia.
- Widok **studenta**: możliwość dodania elementów kształcenia, podgląd rozkładu zajęć i elementów kształcenia swojej grupy.
- Widok **wykładowcy**: możliwość dodania elementów kształcenia, podgląd swojego rozkładu zajęć i elementów kształcenia przypisanych do danego wykładowcy.

Wymienione widoki są dostępne za pośrednictwem bota w Telegramie.

2 Projekt konceptualny

2.1 Sformułowanie zadania projektowego

System oferujący studentom możliwość wymiany informacji na temat ich studiów. Całość interakcji użytkownika przebiega poprzez wykorzystanie platformy Telegram [6].

Każdy użytkownik jest weryfikowany za pomocą systemu USOS [8]. Administrator przydziela odpowiednie role użytkownikom, np. student, moderator etc. Ma on także uprawnienia do zmiany danych o zajęciach.

Student ma dostęp do rozkładu zajęć oraz terminarzu egzaminów, które są moderowane przez administratorów (np. starostów). Każdy użytkownik ma prawo zaproponować informację i notatki o zajęciach lub egzaminach.

Zaprojektowanie bazy danych (dalej – BD), która będzie zawierać informację o:

- studentach (użytkownikach);
- rozkładzie zajęć;
- spotkaniach zajęciowych oraz ponadwymiarowych;
- zadaniach domowych;
- testach (egzaminy, kolokwia, wejściówki).

2.2 Analiza stanu wyjściowego

Wprowadzenie naszego systemu może być prowadzone na parę różnych sposobów. W związku z oferowaniem usług bezpośrednio studentom model biznesowy może zostać oparty na małych dotacjach w celu utrzymania serwisu. Jako że pula użytkowników ogranicza się do ilości studentów, prawdopodobnie możliwe jest utrzymywanie usługi w bardzo niskich kosztach, do momentu aż nasze rozwiązanie nie napotkałoby możliwości rozszerzenia się o większą ilością systemów, bądź użytkowników – wykraczając poza konkretne roczniki, bądź uczelnie. W takiej sytuacji, w związku z oczywistym powiązaniem edukacyjnym, możliwa jest współpraca z uczelnią w celu korzystania z uczelnianej przestrzeni serwerowej, aby móc utrzymać aplikacje na obecnej puli użytkowników, oraz móc rozwinąć się w celu dalszego skalowania naszej usługi.

2.3 Analiza wymagań użytkownika

Użytkownik korzystający z naszej usługi, będzie miał możliwość z interakcją z “botem” przez komunikator *Telegram*. Za pomocą określonych komend, wysyłanych do naszej aplikacji, będzie mógł następująco:

- dodawać notatki do zajęć;
- przeglądać materiały dodane przez innych użytkowników;
- sprawdzać swoje obecne zajęcia;
- przeglądać kalendarz wydarzeń.

Oczywiście, zanim użytkownik będzie mieć dostęp do tych funkcji, wymagana będzie rejestracja polegający na połączeniu identyfikatora USOS wraz z kontem na Telegramie, realizowane za pomocą protokołu OAuth 1.0.

Będzie także wyznaczona konkretna pula użytkowników, mających przywileje moderatorskie i administracyjne – dając im możliwość zatwierdzania dodanych materiałów (w celu obrony przed nadużywaniem serwisu), monitorowania działań użytkowników, oraz ich blokowania w razie łamania regulaminu użycia usługi.

2.4 Określenie scenariuszy użycia

Student:

- rejestracja w serwisie;
- aktywacja konta;
- dodawanie notatek do odbytych zajęć;
- przegląd materiałów dodanych do zajęć;
- przegląd wydarzeń;
- przegląd swojego planu zajęć.

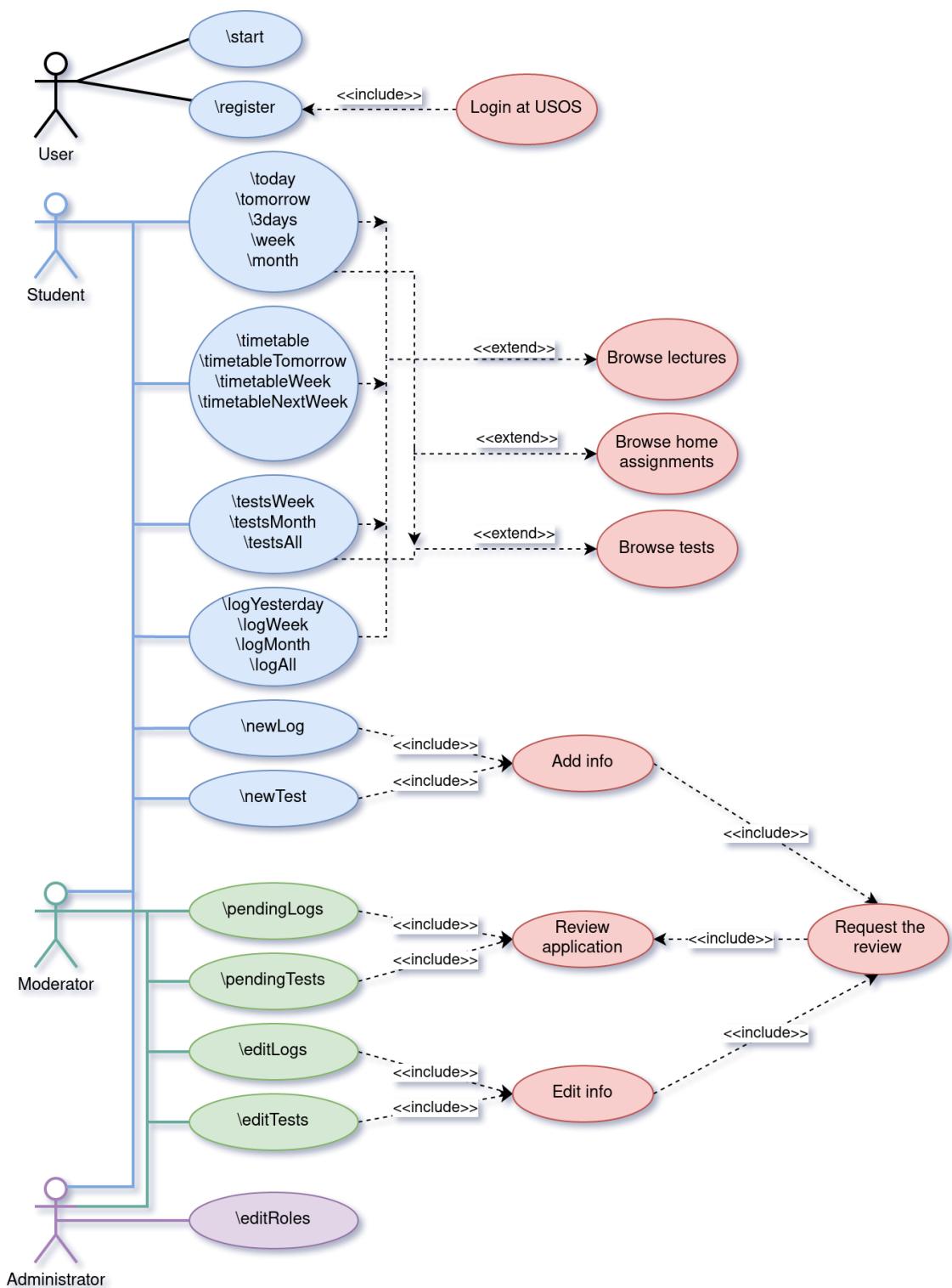
Moderator:

- wszystkie uprawnienia studenta;
- zatwierdzanie lub edycja notatek użytkowników;
- przegląd informacji o użytkownikach.

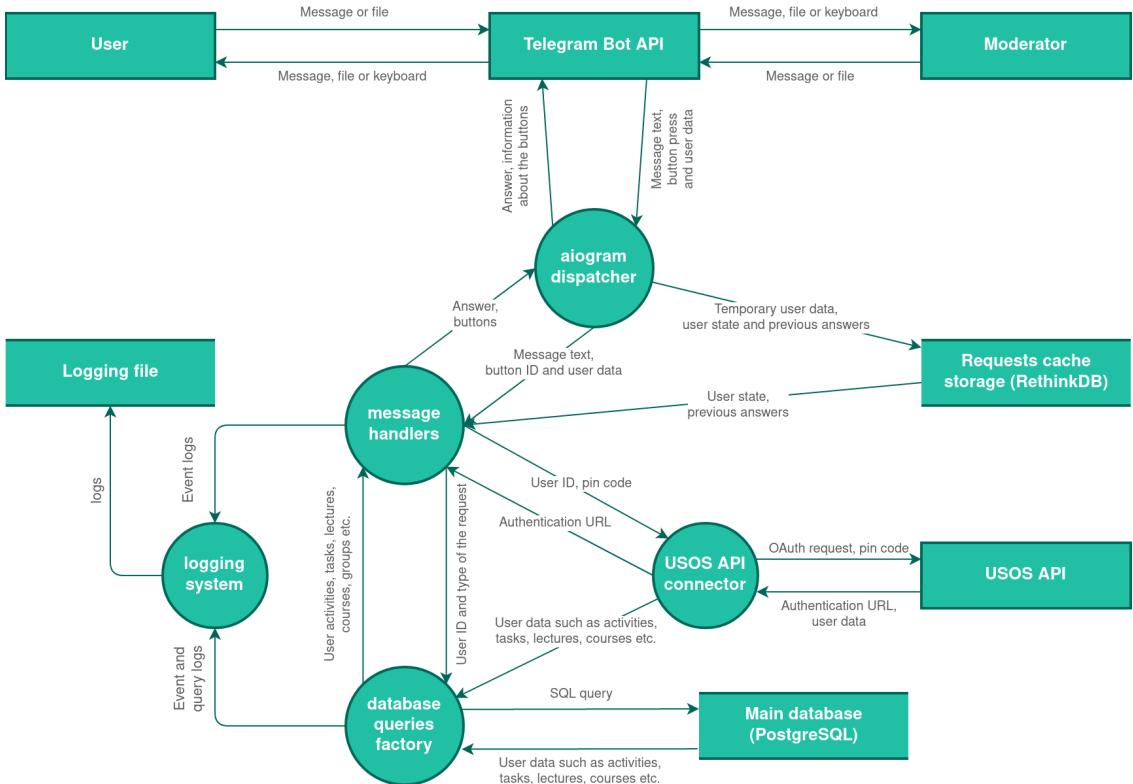
Administrator:

- wszystkie uprawnienia moderatora;
- dodawanie wydarzeń;
- moderacja użytkowników (np. blokowanie ich);
- przydzielanie użytkownikom dodatkowych przywileju;
- przeglądanie logów oraz aktualnego stanu serwera.

UML diagram scenariuszy użycia został podany na rysunku 1.



Rysunek 1: Diagram scenariuszy użycia.



Rysunek 2: Diagram przepływu danych.

2.5 Identyfikacja funkcji

Zostały wyróżnione następujące funkcje, niezbędne dla optymalnej działalności systemu:

- **aiogram dispatcher** – pozwoli na łatwą komunikację z serwerami Telegramu [1][7];
- **message handlers** – zbiór funkcji, które będą obrabiać wszystkie dane wejściowe od użytkownika i przygotowywać dane aplikacji do wysłania użytkownikowi;
- **USOS API connector** – pozwoli łatwo wysyłać zapytania do USOS API [8];
- **database queries factory** – klasa funkcji, które będzie zajmować się komunikacją z bazą danych [2][5];
- **logging system** – funkcję, które będą zbierać logi z każdego modułu aplikacji i zapisywać ich do odpowiednich plików.

Diagram przepływu danych został podany na rysunku 2.

2.6 Propozycja encji, ich atrybutów i powiązań

Opierając się na dane, potrzebne dla poprawnej działalności aplikacji oraz strukturę systemu USOS został przygotowany następujący zbiór encji, opisany poniżej. Na podstawie tego zbioru został również przygotowany diagram związków encji, który znajduje się na rysunku 3.

terms – semestry na uczelni (nazwa, początek, koniec).

courses – przedmioty, wykładane na uczelni.

usos units Ta encja wynika z architektury bazy danych USOS. Każdy typ zajęć każdego przedmiotu (np. LAB Bazy Danych) ma swój **usos_unit_id**. Każdy przedmiot może mieć jeden lub więcej **usos_unit_id** (np. WYK Bazy Danych, LAB Bazy Danych i PRO Bazy Danych).

unit groups Każdy **usos_unit_id** może posiadać jedną lub więcej grup (np. grupa 4 LAB Bazy Danych i 5 LAB Bazy Danych), stąd w tej tabeli zostały zebrane wszystkie rzeczywiste grupy dla każdego **usos_unit_id**.

group types – typy grup (np. WYK, PRO, LAB).

rooms – pokoje pracowników uczelni i sale zajęciowe.

chats – czaty, do których ma dostęp bot.

users – użytkownicy bota.

permissions – lista uprawnień każdej klasy użytkowników lub poszczególnych użytkowników (każdy użytkownik może mieć swój własny zbiór uprawnień, ale większość użytkowników będą korzystać ze standardowych zbiorów np. student, moderator, administrator itp.).

study programmes – lista programów studiów.

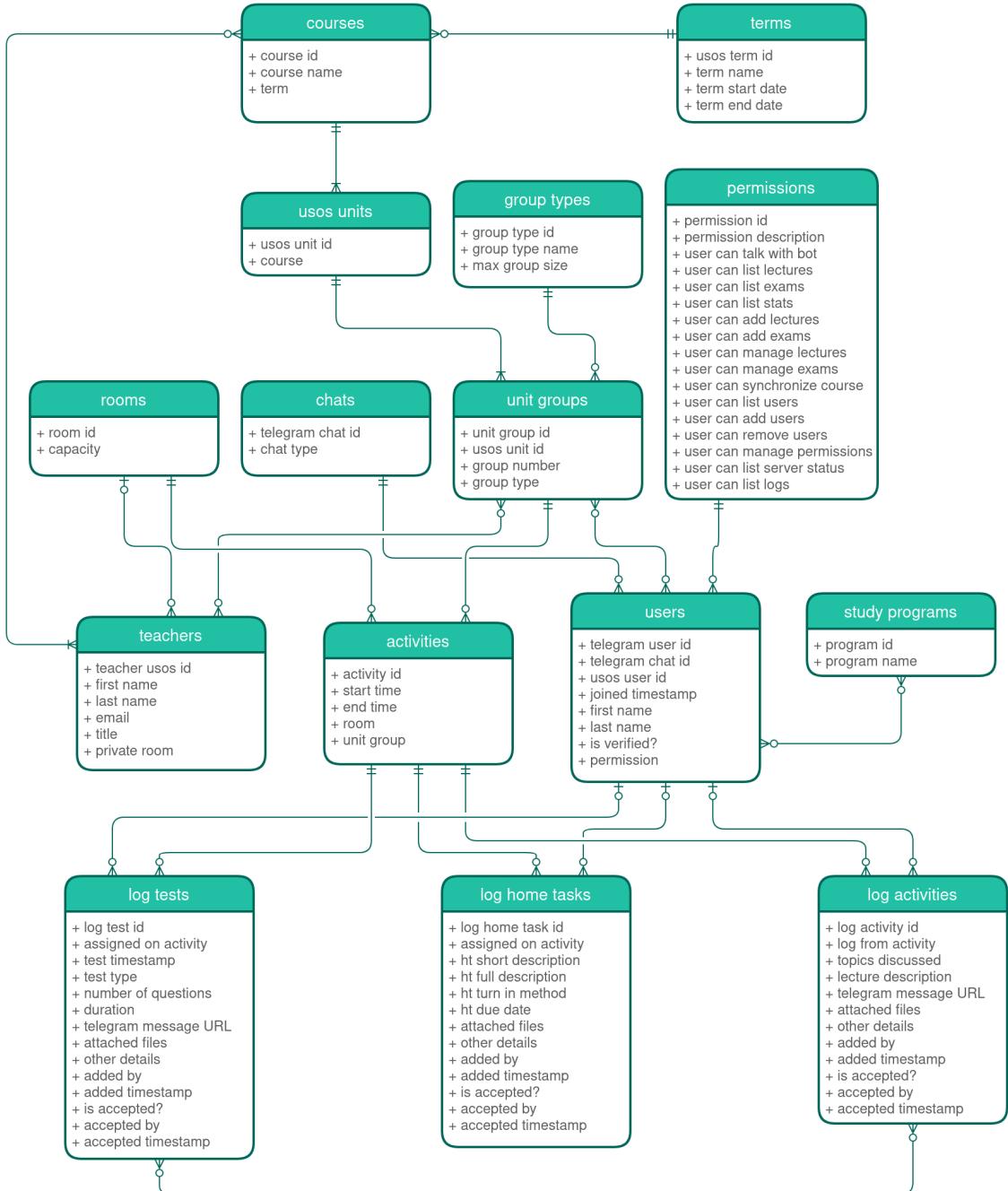
teachers – informacje o prowadzących i koordynatorach przedmiotów.

activities – lista zajęć, czas ich rozpoczęcia i zakończenia.

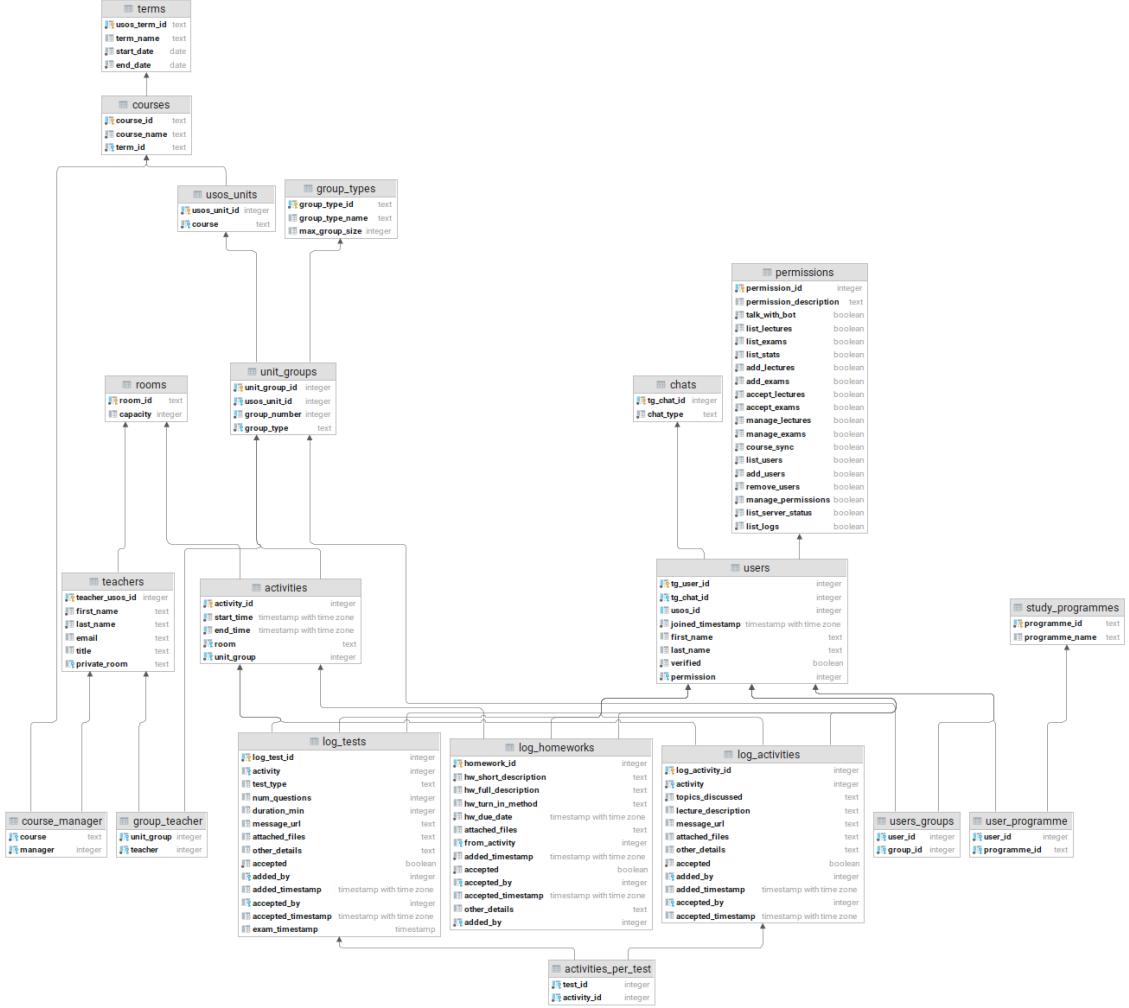
log activities – notatki lekcyjne, opis zajęć i omówionych tematów, pliki oraz dane osób, które pracowali nad przygotowaniem notatki.

log home tasks – informacje o treści, terminie i sposobie oddania prac domowych oraz dane osób, które pracowali nad przygotowaniem notatki.

log tests – informacje o terminie i zagadnieniach dotyczących sprawdzianu oraz dane osób, które pracowali nad przygotowaniem notatki.



Rysunek 3: Diagram związków encji.



Rysunek 4: Diagram relacyjnej bazy danych przed normalizacją.

3 Projekt logiczny

3.1 Przejście na model relacyjny

Na podstawie diagramu związków encji przedstawionego na rysunku 3 został przygotowany model relacyjny. Wszystkie relacje $N - N$ (wiele do wiele) zostały zamienione tabelami asocjacyjnymi, niektóre nazwy były skrócone w celu ulepszenia czytelności. Diagram utworzonej relacyjnej bazy został podany na rysunku 4.

3.2 Analiza zależności funkcyjnych

W tym podrozdziale zostaną przeanalizowane wszystkie tabele pod kątem zależności funkcyjnej. Jak okaże się później, wszystkie tabele zostały zoptymalizowane do czwartej postaci normalnej (1NF, 2NF, 3NF, BCNF, 4NF), oprócz dwóch, które nie spełniają kryterium 3NF. Ponieważ autory projektu wybrali metodę "bottom-top network design", zamiast bardziej poprawnej "top-down network design", jeszcze do rozpoczęcia analizy funkcyjnej została już napisana aplikacja, która mocno polega na bazie danych. Wracając do tematu optymalizacji, żeby poprawić jedną tabelę, trzeba by było przerobić znaczącą ilość procedur tej aplikacji, dlatego została

przyjęta decyzja stworzyć nową pustą bazę danych, na której zostanie pokazana poprawna postać znormalizowana, natomiast wszystkie kwerendy wykonać na starej.

3.2.1 Terms

usos_term_id	term_name	start_date	end_date
1 2021/22-Z	Semestr zimowy 2021/22	2021-10-01	2022-02-28

Rysunek 5: Nagłówek tabeli `terms` i przykładowy wiersz.

Oznaczenia		
uti = usos_term_id tn = term_name sd = start_date ed = end_date		
Zależności funkcyjne	Superklucz	Klucz kandydujący
uti → tn, sd, ed	✓	✓
tn → uti, sd, ed	✓	✓
sd → uti, tn, ed	✓	✓
ed → uti, tn, sd	✓	✓
uti, tn → sd, ed	✓	✗
sd, ed → uti, tn	✓	✗

Wybrany klucz główny: `uti = usos_term_id`

Tabela 1: Analiza zależności funkcyjnych tabeli `terms`.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $usos_term_id \rightarrow \{term_name, start_date, end_date\}$.
3NF	Wszystkie atrybuty są kluczowymi.
4NF	Wszystkie zależności wielowartościowe zaczynają się superkluczem.

Tabela 2: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli `terms`.

3.2.2 Courses

	course_id	course_name	term_id
1	FK/E-DU>ZwIPdzw	Zajęcia wybieralne I - przetwarzanie danych z wykorzystaniem arkusza...	2021/22-Z
2	FK/E-DU>ZWIITG	Zajęcia wybieralne II - Teoria grafów	2021/22-Z
3	FS0-DI>ZWHhg	Zajęcia wybieralne humanistyczne - Historia gospodarcza	2021/22-Z
4	FK/E-DU>ZhIEiPA	Zajęcia humanistyczne I - Etyczne i psychologiczne aspekty zarządzania	2021/22-Z
5	FS0-DI>WMI	Wykład monograficzny I - Równania różnicowe	2021/22-Z
6	FK/E-DU>SemMgr1	Seminarium magisterskie	2021/22-Z
7	FS0-DI>RR	Równania różniczkowe	2021/22-Z
8	FK/E-DU>RR	Równania różniczkowe	2021/22-Z
9	FS0-DI>RP	Rachunek prawdopodobieństwa	2021/22-Z
10	FS0-DI>PSISK	Projektowanie systemów i sieci komputerowych	2021/22-Z

Rysunek 6: Nagłówek tabeli courses i kilka przykładowych wierszy.

Oznaczenia	ci = course_id cn = course_name ti = term_id
Zależności funkcyjne	Superklucz Klucz kandydujący
ci → cn, ti	✓ ✓
ci, cn → ti	✓ ✗
ci, ti → cn	✓ ✗
Klucz główny: ci = course_id	

Tabela 3: Analiza zależności funkcyjnych tabeli courses.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $course_id \rightarrow \{course_name, term_id\}$.
3NF	Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.
4NF	Tabela nie posiada zależności wielowartościowych.

Tabela 4: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli courses.

3.2.3 Group types

	group_type_id	group_type_name	max_group_size
1	PRO	Projekt	16
2	ĆW	Ćwiczenia	40
3	WYK	Wykład	200

Rysunek 7: Nagłówek tabeli `group_types` i kilka przykładowych wierszy.

Oznaczenia	gti = group_type_id gtn = group_type_name mgs = max_group_size
Zależności funkcyjne	Superklucz Klucz kandydujący
$gti \rightarrow gtn, mgs$	✓ ✓
$gtn \rightarrow gti, mgs$	✓ ✓
$gti, gtn \rightarrow mgs$	✓ ✗
Wybrany klucz główny: gti = group_type_id	

Tabela 5: Analiza zależności funkcyjnych tabeli `group_types`.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $group_type_id \rightarrow \{group_type_name, max_group_size\}$.
3NF	Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.
4NF	Tabela nie posiada zależności wielowartościowych.

Tabela 6: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli `group_types`.

3.2.4 USOS units

	usos_unit_id	course
1	113442	FK/E-DU>GR
2	113441	FK/E-DU>GR
3	113444	FK/E-DU>PaMFiu
4	113443	FK/E-DU>PaMFiu
5	113446	FK/E-DU>RR
6	113445	FK/E-DU>RR
7	113447	FK/E-DU>SemMgr1

Rysunek 8: Nagłówek tabeli usos_units i kilka przykładowych wierszy.

Oznaczenia	uni = usos_unit_id crs = course
Zależności funkcyjne	Superklucz Klucz kandydujący
uni → crs	✓ ✓
Klucz główny: uni = usos_unit_id	

Tabela 7: Analiza zależności funkcyjnych tabeli usos_units.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $usos_unit_id \rightarrow \{course\}$.
3NF	Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.
4NF	Tabela nie posiada zależności wielowartościowych.

Tabela 8: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli usos_units.

3.2.5 Unit groups

	unit_group_id	usos_unit_id	group_number	group_type
1	77	113442	1	ĆW
2	78	113441	1	WYK
3	79	113444	1	ĆW
4	80	113443	1	WYK
5	81	113446	1	ĆW
6	82	113445	1	WYK
7	83	113447	1	SEM-MGR
8	84	113447	2	SEM-MGR

Rysunek 9: Nagłówek tabeli unit_groups i kilka przykładowych wierszy.

Oznaczenia	ugi = unit_group_id uni = usos_unit_id gn = group_number gt = group_type
Zależności funkcyjne	Superklucz Klucz kandydujący
ugi → uni, gn, gt	✓ ✓
uni, gn → ugi, gt	✓ ✗
uni → gt	✗ ✗
Klucz główny: ugi = unit_group_id	

Tabela 9: Analiza zależności funkcyjnych tabeli unit_groups.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $unit_group_id \rightarrow \{usos_unit_id, group_number, group_type\}$.
3NF	Atrybut niekluczowy group_type jest zależny od innego atrybutu niekluczowego usos_unit_id .

Tabela 10: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli unit_groups.

Możemy zauważyć, iż tabela unit_groups nie spełnia warunków trzeciej i czwartej postaci normalnej. Rozwiążanie tego problemu jest dość łatwe i oczywiste: wystarczy przenieść kolumnę **group_type** do tabeli **usos_units**. W takim razie obie tabeli **usos_units** i **unit_groups** będą spełniać warunki czwartej postaci normalnej.

3.2.6 Rooms

	room_id	capacity
1	V.5(V-D.49)	20
2	K.36	15
3	F.604	16
4	V.12(V-D.113)	30
5	J.006	14

Rysunek 10: Nagłówek tabeli rooms i kilka przykładowych wierszy.

Oznaczenia	ri = room_id cp = capacity
Zależności funkcyjne	Superklucz Klucz kandydujący
ri → cp	✓ ✓
Klucz główny: ri = room_id	

Tabela 11: Analiza zależności funkcyjnych tabeli rooms.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $room_id \rightarrow \{capacity\}$.
3NF	Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.
4NF	Tabela nie posiada zależności wielowartościowych.

Tabela 12: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli rooms.

3.2.7 Chats

	tg_chat_id	chat_type
1	[REDACTED] 845	private
2	[REDACTED] 238	private
3	[REDACTED] 188	private
4	[REDACTED] 517	private
5	[REDACTED] 507	private

Rysunek 11: Nagłówek tabeli `chats` i kilka przykładowych wierszy. Niektóre dane zostały ukryte ze względów bezpieczeństwa i prywatności.

Oznaczenia	tci = tg_chat_id ct = chat_type
Zależności funkcyjne	Superklucz Klucz kandydujący
tci → ct	✓ ✓
Klucz główny: ct = tg_chat_id	

Tabela 13: Analiza zależności funkcyjnych tabeli `chats`.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $tg_chat_id \rightarrow \{chat_type\}$.
3NF	Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.
4NF	Tabela nie posiada zależności wielowartościowych.

Tabela 14: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli `chats`.

3.2.8 Permissions

	1	2	3	4	5	6
permission_id	1	2	3	4	5	6
permission_description	New user	Main admin	Ordinary student	Moderator	Administrator	Banned user
talk_with_bot	true	true	true	true	true	false
list_lectures	false	true	true	true	true	false
list_exams	false	true	true	true	true	false
list_stats	false	true	true	true	true	false
add_lectures	false	true	true	true	true	false
add_exams	false	true	true	true	true	false
accept_lectures	false	true	false	true	true	false
accept_exams	false	true	false	true	true	false
manage_lectures	false	true	false	false	true	false
manage_exams	false	true	false	false	true	false
course_sync	false	true	false	false	true	false
list_users	false	true	false	false	true	false
add_users	false	true	false	false	true	false
remove_users	false	true	false	false	false	false
manage_permissions	false	true	false	false	false	false
list_server_status	false	true	false	false	false	false
list_logs	false	true	false	false	false	false

Rysunek 12: Nagłówek tabeli permissions i kilka przykładowych wierszy w postaci transponowanej.

Oznaczenia	pi = permission_id pd = permission_description attrs – zbiór wszystkich pozostałych atrybutów tabeli
Zależności funkcyjne	Superklucz Klucz kandydujący
pi → pd, attrs	✓ ✓
pd → pi, attrs	✓ ✓
pi, pd → attrs	✓ ✗
Wybrany klucz główny: pi = permission_id	

Tabela 15: Analiza zależności funkcyjnych tabeli permissions.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $permission_id \rightarrow \{permission_description\} \cup attrs$.
3NF	Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.
4NF	Wszystkie zależności wielowartościowe zaczynają się superkluczem.

Tabela 16: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli permissions.

3.2.9 Teachers

	teacher_usos_id	first_name	last_name	email	title	private_room
1	■■■■■8 P ■■■■■	W ■■■■■	<null>	<null>	<null>	<null>
2	■■■■■7 L ■■■■■	R ■■■■■	■■■■■ ■■■■■	<null>	<null>	<null>
3	■■■■■0 R ■■■■■	N ■■■■■	<null>	<null>	<null>	<null>
4	■■■■■1 T ■■■■■	Z ■■■■■	<null>	<null>	<null>	<null>
5	■■■■■4 J ■■■■■	S ■■■■■	<null>	<null>	<null>	<null>
6	■■■■■7 E ■■■■■	R ■■■■■	■■■■■ ■■■■■	<null>	<null>	<null>

Rysunek 13: Nagłówek tabeli `teachers` i kilka przykładowych wierszy. Niektóre dane zostały ukryte ze względów bezpieczeństwa i prywatności.

Oznaczenia	tui = teacher_usos_id fn = first_name ln = last_name e = email t = title pr = private_room
Zależności funkcyjne	Superklucz Klucz kandydujący
$tui \rightarrow fn, ln, e, t, pr$	✓ ✓
$e \rightarrow tui, fn, ln, t, pr$	✓ ✓
$fn, ln, e \rightarrow tui, t, pr$	✓ ✗
$tui, e \rightarrow fn, ln, t, pr$	✓ ✗
Wybrany klucz główny: <code>tui = teacher_usos_id</code>	

Tabela 17: Analiza zależności funkcyjnych tabeli `teachers`.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $tui \rightarrow \{fn, ln, e, t, pr\}$.
3NF	Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.
4NF	Wszystkie zależności wielowartościowe zaczynają się superkluczem.

Tabela 18: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli `teachers`.

3.2.10 Activities

	activity_id	start_time	end_time	room	unit_group	
1	6	2022-01-05 11:15...	2022-01-05 12:45...	F.502		2
2	7	2022-01-13 11:15...	2022-01-13 12:45...	F.502		2
3	8	2022-01-20 11:15...	2022-01-20 12:45...	F.502		2
4	3	2022-01-20 13:00...	2022-01-20 14:30...	F.502		1
5	10	2022-01-31 11:15...	2022-01-31 12:45...	F.502		2

Rysunek 14: Nagłówek tabeli activities i kilka przykładowych wierszy.

Oznaczenia	ai = activity_id st = start_time et = end_time r = room ug = unit_group
Zależności funkcyjne	Superklucz Klucz kandydujący
ai → st, et, r, ug	✓ ✓
st, ug → ai, et, r	✓ ✗
et, ug → ai, st, r	✓ ✗
st, et, ug → ai, r	✓ ✗
r, st, ug → ai, et	✓ ✗
Klucz główny: ai = activity_id	

Tabela 19: Analiza zależności funkcyjnych tabeli activities.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $ai \rightarrow \{st, et, r, ug\}$.
3NF	Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.
4NF	Wszystkie zależności wielowartościowe zaczynają się superkluczem.

Tabela 20: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli activities.

3.2.11 Users

	<code>tg_user_id</code>	<code>tg_chat_id</code>	<code>usos_id</code>	<code>joined_timestamp</code>	<code>first_name</code>	<code>last_name</code>	<code>verified</code>	<code>permission</code>
1	██████████ 188	██████████ 188	<null>	2022-01-02 07:19:18...	<null>	<null>	false	1
2	██████████ 238	██████████ 238	██████████ 98	2022-01-03 11:31:18...	A██████████	G██████████	true	5
3	██████████ 585	██████████ 585	██████████ 51	2022-01-02 15:18:30...	K██████████	P██████████	true	3
4	██████████ 517	██████████ 517	██████████ 13	2022-01-03 11:31:15...	P██████████	K██████████	true	5
5	██████████ 845	██████████ 845	██████████ 07	2022-01-04 16:10:16...	P██████████	K██████████	true	3

Rysunek 15: Nagłówek tabeli `users` i kilka przykładowych wierszy. Niektóre dane zostały ukryte ze względów bezpieczeństwa i prywatności.

Oznaczenia	<code>tui</code> = <code>tg_user_id</code> <code>tci</code> = <code>tg_chat_id</code> <code>ui</code> = <code>usos_id</code> <code>jt</code> = <code>joined_timestamp</code> <code>fn</code> = <code>first_name</code> <code>ln</code> = <code>last_name</code> <code>v</code> = <code>verified</code> <code>p</code> = <code>permission</code>
Zależności funkcyjne	Superklucz Klucz kandydujący
$tui \rightarrow tci, ui, jt, fn, ln, v, p$	✓ ✓
$tci \rightarrow tui, ui, jt, fn, ln, v, p$	✓ ✓
$tui, tci \rightarrow ui, jt, fn, ln, v, p$	✓ ✗
$ui \rightarrow v$	✗ ✗
Klucz główny: <code>ai</code> = <code>activity_id</code>	

Tabela 21: Analiza zależności funkcyjnych tabeli `users`.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $tui \rightarrow \{tci, ui, jt, fn, ln, v, p\}$.
3NF	Atrybut niekluczowy <code>verified</code> jest zależny od innego atrybutu niekluczowego <code>usos_id</code> .

Tabela 22: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli `users`.

Możemy zauważyc, iż tabela `users` nie spełnia warunków trzeciej i czwartej postaci normalnej. Zoptymalizować tę tabelę możemy usuwając kolumnę `verified`, ponieważ można ją zawsze wyliczyć z wartości kolumny `usos_id`: jeżeli `usos_id = NULL`, to `verified = false`, w przeciwnym wypadku `verified = true`.

3.2.12 Study programmes

programme_id	programme_name
1 FS-DI	Inżynieria i analiza danych, st. I-go stopnia (inż.)
2 FK/E-DU	Matematyka-zastosowania matematyki w ekonomii, st. II-go stopnia

Rysunek 16: Nagłówek tabeli `study_programmes` i kilka przykładowych wierszy.

Oznaczenia	pi = programme_id pn = programme_name
Zależności funkcyjne	Superklucz Klucz kandydujący
pi → pn pn → pi	✓ ✓ ✓ ✓
Klucz główny: pi = programme_id	

Tabela 23: Analiza zależności funkcyjnych tabeli `study_programmes`.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $pi \rightarrow \{pn\}$.
3NF	Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.
4NF	Tabela nie posiada zależności wielowartościowych.

Tabela 24: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli `study_programmes`.

3.2.13 Log activities

	log_activity_id	activity	topics_discussed	lecture_description	m...	a...	o...	a...	a...	adde...	ac...	acce...
1	4	23 Równania Ricattiego..	Powtórzenie materiału	Różniczkowe - Ćwiczeni...	<null>	<null>	<null>	• true	507	2022-01-08 ..	507	2022-02-08 ..
2	9	71 Całki, pochodne	Różniczkowe - Ćwiczeni...	<null>	<null>	<null>	<null>	false	507	2022-02-08 ..	<null>	<null>
3	11	99 Zmienne losowe dwuwy..	Powtórzenie materiału	Różniczkowe - Ćwiczeni...	<null>	<null>	<null>	false	95..	2022-02-08 ..	<null>	<null>
4	7	49 Przewidywanie, układy	Różniczkowe - Ćwiczeni...	<null>	<null>	<null>	<null>	• true	95..	2021-12-01 ..	507	2022-01-02 ..
5	10	98 Prace nad projektem ..	- LabView - Projektowe n..	<null>	<null>	<null>	<null>	• true	33..	2022-01-17 ..	188	2022-01-25 ..

Rysunek 17: Nagłówek tabeli `log_activities` i kilka przykładowych wierszy.
Niektóre dane zostały ukryte ze względów bezpieczeństwa i prywatności.

Oznaczenia	li = log_activity_id a = activity t = topics_discussed l = lecture_description m = message_url f = attached_files d = other_details adt = added_timestamp adb = added_by ac = accepted act = accepted_timestamp acb = accepted_by accol = {ac, act, acb} adcol = {adt, adb}
Zależności funkcyjne	Superklucz Klucz kandydujący
$li \rightarrow a, t, l, m, f, d, adcol, accol$ $m \rightarrow li, a, t, l, f, d, adcol, accol$	✓ ✓ ✓ ✓
Klucz główny: li = log_activity_id	

Tabela 25: Analiza zależności funkcyjnych tabeli `log_activities`.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $li \rightarrow \{a, t, l, m, f, d, adcol, accol\}$.
3NF	Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.
4NF	Wszystkie zależności wielowartościowe zaczynają się superkluczem.

Tabela 26: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli `log_activities`.

3.2.14 Log tests

	log_test_id	activity	te...	n...	d...	m...	a...	other_details	a...	a...	adde...	a...	acc...	exa...	
1	2	50 Egzamin	5	120	████████	<null>	ubrać się galowo!	• true	████	88	2021-12-07...	████	188	2022-01-01...	2022-02-19...
2	4	100 Kolokwium	10	90	████████	<null>	nie spóźnić się	• true	████	17	2022-02-02...	████	188	2022-02-20...	2022-02-17...
3	5	125 Kartkówka	50	30	████████	<null>	stówka na teamsach	false	████	3...	2022-02-01...	<null>	<null>	2022-02-02...	
4	1	25 Kolokwium	10	90	████████	<null>	pamiętać o kalku...	• true	████	07	2022-01-01...	████	507	2022-01-08...	2022-02-12...
5	3	75 Kolokwium	5	90	████████	<null>	5 min wcześniej	• true	████	5...	2022-01-11...	████	507	2022-01-15...	2022-02-08...

Rysunek 18: Nagłówek tabeli `log_tests` i kilka przykładowych wierszy.
Niektóre dane zostały ukryte ze względów bezpieczeństwa i prywatności.

Oznaczenia	<code>lti = log_test_id</code> <code>m = message_url</code> <code>attrs – zbiór wszystkich pozostałych atrybutów tabeli</code>
Zależności funkcyjne	Superklucz Klucz kandydujący
<code>lti → m, attrs</code>	✓ ✓
<code>m → lti, attrs</code>	✓ ✓
Klucz główny: <code>lti = log_test_id</code>	

Tabela 27: Analiza zależności funkcyjnych tabeli `log_tests`.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $lti \rightarrow \{m, attrs\}$.
3NF	Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.
4NF	Wszystkie zależności wielowartościowe zaczynają się superkluczem.

Tabela 28: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli `log_tests`.

3.2.15 Log homeworks

	hw_id	hw_short_descri...	hw_full_description	h...	hw_d...	...	adde...	a...	a...	acce...	other_details	ad...
1	4	Równania Riccatiego	Nauczyć się rozwiązywać równania ...	Teams	2021-11-16..	<null>	30	2022-01-08..	• true	1521989...	2022-01-11.. Przynieś a dosta...	576145188
2	5	Śłówka - UNIT 5		Teams	2022-01-12..	<null>	40	2022-02-06..	false	<null> <null>	<null>	576145188
3	3	Opis projekt	Przygotować sprawozdanie, które d...	Email	2022-01-18..	<null>	20	2022-01-03..	false	<null> <null>	Czas do 16!	533847587
4	6	Wszystkie prace domowe	Wykonać wszystkie prace domowe z ...	IRL	2022-01-16..	<null>	50	2022-01-30..	• true	1366223...	2022-02-02.. Musi być wydrukow...	533847587
5	2	Równanian I-go rzędu	Nauczyć się rozwiązywać równania ...	Teams	2021-12-21..	<null>	10	2022-01-10..	• true	5338475...	2022-01-13.. Lepiej przynieść!	533847587

Rysunek 19: Nagłówek tabeli log_homeworks i kilka przykładowych wierszy.

Niektóre dane zostały ukryte ze względów bezpieczeństwa i prywatności.

Oznaczenia	lhi = log_homeworks_id attrs – zbiór wszystkich pozostałych atrybutów tabeli
Zależności funkcyjne	Superklucz Klucz kandydujący
lhi → attrs	✓ ✓
Klucz główny: lhi = log_homeworks_id	

Tabela 29: Analiza zależności funkcyjnych tabeli log_homeworks.

Postać	Sprawdzenie
1NF	Wszystkie wartości są atomowe.
2NF	Wszystkie wartości są zależne od klucza głównego: $lhi \rightarrow attrs$.
3NF	Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.
4NF	Wszystkie zależności wielowartościowe zaczynają się superkluczem.

Tabela 30: Sprawdzenie spełnienia warunków czwartej postaci normalnej tabeli log_homeworks.

3.2.16 Course manager (tabela asocjacyjna)

Tabela `course_manager` łączy przedmioty i wykładowców, które są koordynatorami odpowiednich przedmiotów. Jest to relacja wiele do wiele, ponieważ każdy wykładowca może być kierownikiem wielu przedmiotów i każdy przedmiot może mieć wiele koordynatorów. Jedynym superkluczem tej tabeli jest zbiór obu kolumn tabeli, a mianowicie `{course, manager}`. Ten superklucz występuje również kluczem kandydującym oraz kluczem głównym tej tabeli.

	course	manager
1	FS0-DI>DB	3
2	FS0-DI>Jang2	8
3	FS0-DI>Jang2	6
4	FS0-DI>Jang2	6
5	FS0-DI>Jang2	8

Rysunek 20: Nagłówek tabeli `course_manager` i kilka przykładowych wierszy.
Niektóre dane zostały ukryte ze względów bezpieczeństwa i prywatności.

3.2.17 Group teacher (tabela asocjacyjna)

Tabela `group_teacher` łączy prowadzących i grupy, w których prowadzą zajęcia. Jest to relacja wiele do wiele, ponieważ każdy prowadzący może prowadzić zajęcia w wielu grupach i każda grupa może mieć wiele prowadzących. Jedynym superkluczem tej tabeli jest zbiór obu kolumn tabeli, a mianowicie `{unit_group, teacher}`. Ten superklucz występuje również kluczem kandydującym oraz kluczem głównym tej tabeli.

	unit_group	teacher
1	243	1
2	242	1
3	241	1
4	240	1
5	239	1
6	238	1
7	238	4
8	238	9
9	237	8
10	237	0

Rysunek 21: Nagłówek tabeli `group_teacher` i kilka przykładowych wierszy.
Niektóre dane zostały ukryte ze względów bezpieczeństwa i prywatności.

3.2.18 Activities per test (tabela asocjacyjna)

Tabela `activities_per_test` łączy notatki z zajęć i zapowiedziane sprawdziany. Pozwoli to łatwo otrzymać informacje o treściach, które będą obowiązywać na sprawdzianie. Jest to relacja wiele do wiele, ponieważ każdy sprawdzian może zawierać wiele tematów oraz każdy temat może być sprawdzany na różnych sprawdzianach. Jedynym superkluczem tej tabeli jest zbiór obu kolumn tabeli, a mianowicie `{test_id, activity_id}`. Ten superklucz występuje również kluczem kandydującym oraz kluczem głównym tej tabeli.

	test_id	activity_id
1	1	4
2	1	5
3	2	6
4	2	7
5	3	8

Rysunek 22: Nagłówek tabeli `activities_per_test` i przykładowe wierszy.

3.2.19 Users groups (tabela asocjacyjna)

Tabela `users_groups` łączy grupy i przypisanych do nich studentów. Jest to relacja wiele do wiele, ponieważ każda grupa może zawierać wiele studentów oraz każdy student może być zawartym w wielu grupach. Jedynym superkluczem tej tabeli jest zbiór obu kolumn tabeli, a mianowicie `{user_id, group_id}`. Ten superklucz występuje również kluczem kandydującym oraz kluczem głównym tej tabeli.

	user_id	group_id
1	[REDACTED]507	4
2	[REDACTED]507	10
3	[REDACTED]507	13
4	[REDACTED]507	14
5	[REDACTED]507	22
6	[REDACTED]507	28
7	[REDACTED]507	31
8	[REDACTED]507	33
9	[REDACTED]507	38
10	[REDACTED]507	41

Rysunek 23: Nagłówek tabeli `users_groups` i kilka przykładowych wierszy.

Niektóre dane zostały ukryte ze względów bezpieczeństwa i prywatności.

3.2.20 User programme (tabela asocjacyjna)

Tabela `user_programme` łączy użytkowników oraz kierunki ich studiów. Jest to relacja wiele do wiele, ponieważ każdy student może studiować wiele kierunków oraz każdy kierunek może zawierać wiele użytkowników. Jedynym superkluczem tej tabeli jest zbiór obu kolumn tabeli, a mianowicie `{user_id, programme_id}`. Ten superklucz występuje również kluczem kandydującym oraz kluczem głównym tej tabeli.

Po dokonaniu analizy wnioskujemy, iż kolumnę `user_id` w tej tabeli warto zmienić kolumną `course_id` i połączyć z tabelą `courses`, ponieważ prawdziwy związek występuje pomiędzy encjami `course` i `programme`, a nie `user` i `programme`.

	user_id	programme_id
1	507	FS-DI
2	585	FS-DI
3	718	FS-DI
4	517	FS-DI
5	238	FS-DI
6	845	FS-DI
7	332	FS-DI
8	332	FK/E-DU
9	664	FS-DI

Rysunek 24: Nagłówek tabeli `user_programme` i kilka przykładowych wierszy.

Niektóre dane zostały ukryte ze względów bezpieczeństwa i prywatności.

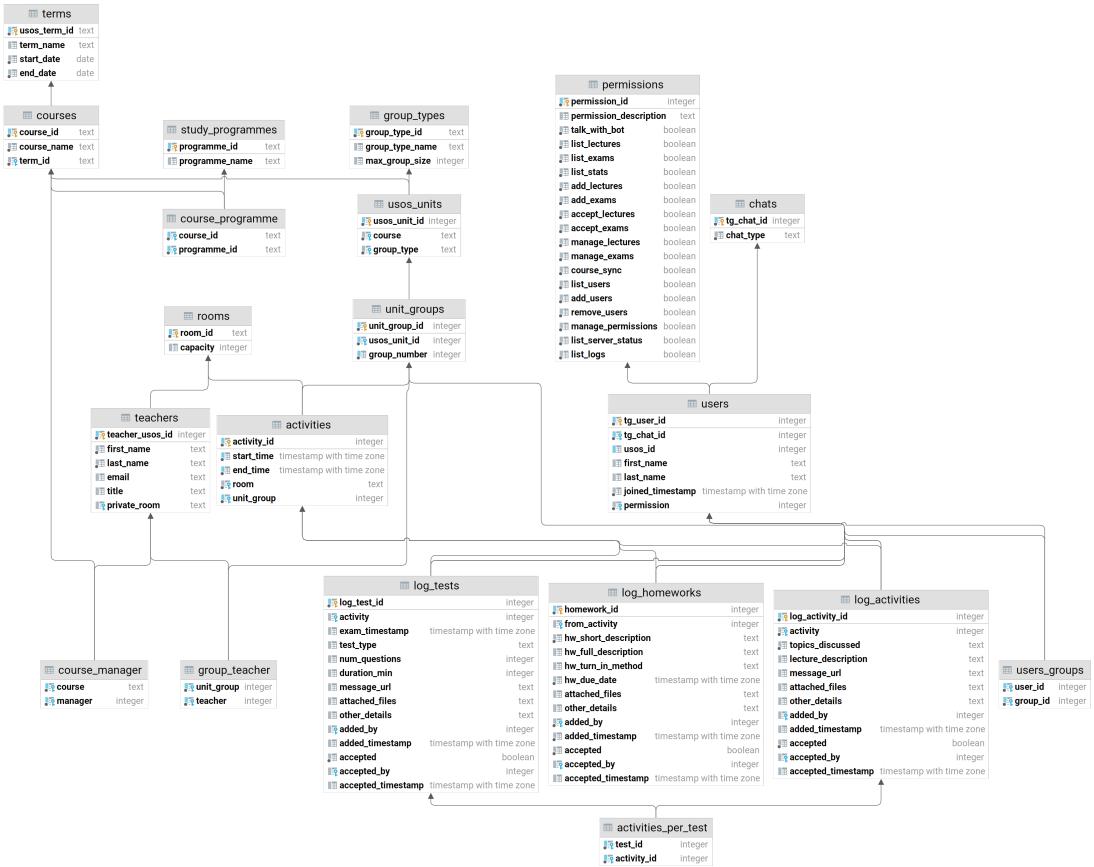
3.3 Model relacyjny po normalizacji

Po uwzględnieniu wszystkich problemów, o których szła mowa w podrozdziale 3.2, został przygotowany nowy model relacyjny, którego diagram znajduje się na rysunku 25. Przypominamy, że diagram modelu relacyjnego przed normalizacją można znaleźć na rysunku 4.

3.4 Operacje na danych w notacji algebry relacyjnej

1. Sale, które pomieszczą wszystkich zalogowanych użytkowników.

$$\begin{aligned} & \Pi_{AVG}(\text{user_count}) \\ & \gamma_{AVG}(\text{user_count}) \\ & \rho_{groups} \\ & \Pi_{COUNT}(\text{tg_user_id}) \rightarrow \text{user_count} \\ & \gamma_{unit_group_id, COUNT(tg_user_id)} \\ & (\rho_u \text{ users} \bowtie_u \text{ tg_user_id} = usg \cdot \text{user_id} \\ & \rho_{usg} \text{ users_groups} \bowtie_{ung}^o \text{ unit_group_id} = usg \cdot \text{group_id} \\ & \rho_{ung} \text{ unit_groups}) \end{aligned}$$



Rysunek 25: Diagram relacyjnej bazy danych po dokonaniu normalizacji każdej tabeli do czwartej postaci normalnej.

2. Średnia ilość studentów w grupie.

$$\begin{aligned}
 & \Pi_{AVG} (\text{user_count}) \\
 & \gamma_{AVG} (\text{user_count}) \\
 & \rho_{\text{groups}} \\
 & \Pi_{COUNT} (\text{tg_user_id}) \rightarrow \text{user_count} \\
 & \gamma_{\text{unit_group_id}}, \text{COUNT} (\text{tg_user_id}) \\
 & (\rho_u \text{ users } \bowtie_u \text{ tg_user_id} = \text{usg} \text{ . user_id} \\
 & \rho_{\text{usg}} \text{ users_groups } \bowtie_{\text{ung}}^o \text{ unit_group_id} = \text{usg} \text{ . group_id} \\
 & \rho_{\text{ung}} \text{ unit_groups})
 \end{aligned}$$

3. Administratory, które dołączyły się do aplikacji przed wybraną datą (np. 02.01.2022).

$$\begin{aligned}
 & \Pi_{COUNT} (\text{users}) \\
 & \gamma_{COUNT} (\text{users}) \\
 & \sigma_{\text{permission_id} = 2 \text{ AND } \text{joined_timestamp} \geq "2021-01-02"} (\text{users } \bowtie_{\text{permission_id}} \text{ permission} \\
 & \text{permissions})
 \end{aligned}$$

4. Lista zajęć od teraz do końca semestru.

$$\begin{aligned} & \Pi_{\text{COUNT}}(\text{ac}) \\ & \Gamma_{\text{COUNT}}(\text{ac}) \\ & \sigma_{\text{ac} . \text{start_time} = t . \text{end_date}} \\ & (\rho_{\text{ac}} \text{ activities} \bowtie_{\text{ac}} \text{ unit_group} = \text{ung} . \text{unit_group_id} \\ & \rho_{\text{ug}} \text{ unit_groups} \bowtie_{\text{ung}} \text{ usos_unit_id} = \text{uu} . \text{usos_unit_id} \\ & \rho_{\text{uu}} \text{ usos_units} \bowtie_{\text{uu}} \text{ course} = \text{c} . \text{course_id} \\ & \rho_{\text{c}} \text{ courses} \bowtie_{\text{c}} \text{ term_id} = t . \text{usos_term_id} \\ & \rho_t \text{ terms}) \end{aligned}$$

5. Pokoje, w których przeprowadzono największą liczbę zajęć, posortowane malejąco.

$$\begin{aligned} & \Pi_{\text{COUNT}(\text{activity_id})} \downarrow \gamma_{\text{room_id}}, \text{COUNT}(\text{activity_id}) \\ & (\rho_a \text{ activities} \bowtie_a \text{ room} = r . \text{room_id} \\ & \rho_r \text{ rooms}) \end{aligned}$$

6. Procent zweryfikowanych użytkowników.

$$\begin{aligned} & \Pi_{\text{verified_amount} / \text{total_amount} * 100} \\ & (\rho_{\text{verified_amount}} \\ & \Pi_{\text{COUNT}(\text{users}) \rightarrow \text{verified_amount}} \\ & \gamma_{\text{COUNT}(\text{users})} \\ & \sigma_{\text{verified} = \text{true}} \} \text{ users} \times \\ & \rho_{\text{total_amount}} \\ & \Pi_{\text{COUNT}(\text{users}) \rightarrow \text{total_amount}} \\ & \gamma_{\text{COUNT}(\text{users}) \text{ users}}) \end{aligned}$$

7. Zadania domowe na przyszły tydzień

$$\begin{aligned} & \Pi_{\text{hw_short_description}, \text{hw_full_description}, \text{hw_turn_in_method}, \text{hw_due_date}} \\ & \sigma_{\text{hw_due_date} = \text{now} + 7} \end{aligned}$$

8. W ilu grupach prowadzą zajęcia, posortowane malejąco względem ilości grup.

$$\begin{aligned} & \Pi_{\text{MAX}(\text{ilosc_zajec})} \\ & \gamma_{\text{MAX}(\text{ilosc_zajec})} \\ & (\rho_{\text{najwiecej_zajec}} \\ & \Pi_{\text{COUNT}(\text{group_type}) \rightarrow \text{ilosc_zajec}, t} \\ & \gamma_{\text{teacher_usos_id}, \text{group_type}, \text{teacher_usos_id}, \text{COUNT}(\text{group_type})} \\ & (\rho_t \text{ teachers} \bowtie_t \text{ teacher_usos_id} = \text{gt} . \text{teacher} \\ & \rho_{\text{gt}} \text{ group_teacher} \bowtie_{\text{gt}} \text{ unit_group} = \text{ug} . \text{unit_group_id} \\ & \rho_{\text{ug}} \text{ unit_groups} \bowtie_{\text{uu}} \text{ usos_unit_id} = \text{ug} . \text{usos_unit_id} \\ & \rho_{\text{uu}} \text{ usos_units} \bowtie_{\text{uu}} \text{ course} = \text{c} . \text{course_id} \\ & \rho_c \text{ courses}) \times \\ & \rho_t \text{ teachers}) \end{aligned}$$

9. Prowadzące, które prowadzą zajęcia dla obu studentów X i Y

$$\begin{aligned} & \delta \Pi_{t .} \text{first_name}, t . \text{last_name} \\ & \sigma_{u .} \text{tg_user_id} = X \\ & (\rho_t \text{ teachers} \bowtie_{t .} \text{teacher_usos_id} = \text{grt} . \text{teacher} \\ & \rho_{\text{grt}} \text{ group_teacher} \bowtie_{\text{grt} .} \text{unit_group} = \text{ung} . \text{unit_group_id} \\ & \rho_{\text{ung}} \text{ unit_groups} \bowtie_{\text{ung} .} \text{unit_group_id} = \text{usg} . \text{group_id} \\ & \rho_{\text{usg}} \text{ users_groups} \bowtie_{\text{usg} .} \text{user_id} = u . \text{tg_user_id} \\ & \rho_u \text{ users}) \cap \delta \\ & \Pi_{t .} \text{first_name}, t . \text{last_name} \\ & \Pi_{t .} \text{first_name}, t . \text{last_name} \\ & \sigma_{u .} \text{tg_user_id} = Y \\ & (\rho_t \text{ teachers} \bowtie_{t .} \text{teacher_usos_id} = \text{grt} . \text{teacher} \\ & \rho_{\text{grt}} \text{ group_teacher} \bowtie_{\text{grt} .} \text{unit_group} = \text{ung} . \text{unit_group_id} \\ & \rho_{\text{ung}} \text{ unit_groups} \bowtie_{\text{ung} .} \text{unit_group_id} = \text{usg} . \text{group_id} \\ & \rho_{\text{usg}} \text{ users_groups} \bowtie_{\text{usg} .} \text{user_id} = u . \text{tg_user_id} \\ & \rho_u \text{ users}) \end{aligned}$$

4 Projekt implementacyjny

4.1 Tworzenie tabel

Całkowity kod w języku PL/pgSQL (dla systemu bazodanowego PostgreSQL), który tworzy wszystkie tabele został podany w załączniku `pgsql_SysOrgStud.sql`, natomiast na obrazku 26 został podany przykładowy kod, generujący tabelę `activities` (zarówno nagłówek, jak i kilka przykładowych wierszy z tej tabeli zostały podane na rysunku 14) [2].

```
CREATE TABLE activities
(
    activity_id integer GENERATED ALWAYS AS IDENTITY
        CONSTRAINT activities_pk
            PRIMARY KEY,
    start_time timestamp NOT NULL,
    end_time   timestamp NOT NULL,
    room       text      NOT NULL
        CONSTRAINT activities_room_fk
            REFERENCES rooms
            ON DELETE RESTRICT,
    unit_group integer    NOT NULL
        CONSTRAINT activities_unit_group_fk
            REFERENCES unit_groups
            ON DELETE RESTRICT
);

CREATE UNIQUE INDEX activities_activity_id_uindex
    ON activities (activity_id);

CREATE UNIQUE INDEX activities_group_collision_uindex
    ON activities (unit_group, start_time, end_time);
```

Rysunek 26: Tworzenie tabeli `activities` w języku PL/pgSQL.

4.2 Uzupełnianie tabel danymi

Ponieważ dane, wykorzystywane w trakcie realizacji projektu, zawierają informacje wrażliwe (np. telegram user id, imię i nazwisko użytkownika) nie są generowane losowo, a pobierane z prawdziwych źródeł, to całkowity kod uzupełnienia tabeli nie może zostać udostępniony. Natomiast zostaną pokazane przykładowe funkcji, które uzupełniają bazę danych w sposób automatyczny.

Na rysunku 27 został podany przykładowy kod uzupełnienia tabeli `activities` w sposób ręczny. Dla rzeczywistego uzupełnienia tabeli została stworzona aplikacja bazodanowa w języku Python z użyciem biblioteki `psycopg2` [3][4]. Taka działalność pozwoliła stworzyć 1506 unikalnych prawdziwych wierszy w tabeli `activities`.

Funkcja uzupełniająca tabelę `activities` została podana na rysunku 28. Można zauważyć, iż funkcja pokazana na tym rysunku używa upsert zamiast zwykłego inserta. Taka decyzja została zrobiona dla ulepszenia niezawodności aplikacji (np. jeżeli dwa użytkownicy będą chcieli dodać zajęcie, które rozpoczyna się i kończy się w tym samym czasie dla tej samej grupy, to aktualizuje się tylko pokój, w którym to zajęcie będzie prowadzone).

```
INSERT INTO public.activities (start_time, end_time, room, unit_group)
VALUES ('2022-01-05 14:00:00+01'::timestamptz,
        '2022-01-05 15:30:00+01'::timestamptz,
        'F.502', 1),
        ('2022-01-13 14:00:00+01'::timestamptz,
        '2022-01-13 15:30:00+01'::timestamptz,
        'F.502', 1);
```

Rysunek 27: Uzupełnienie tabeli `activities` w sposób ręczny.

```
def upsert_activities(self, start_time, end_time, room, unit_group):
    """Updates or inserts an activity."""
    cur = self.conn.cursor()
    query = ("INSERT INTO public.activities "
             "(start_time, end_time, room, unit_group) VALUES "
             "(:start_time, :end_time, :room, :unit_group)"
             "ON CONFLICT (start_time, end_time, unit_group) "
             "DO UPDATE SET room = excluded.room;")
    cur.execute(query, {"start_time": start_time,
                       "end_time": end_time,
                       "room": room,
                       "unit_group": unit_group})
    self.conn.commit()
    cur.close()
    logging.debug(f"{unit_group=} {start_time=} {end_time=} {room=}")
```

Rysunek 28: Uzupełnienie tabeli `activities` w sposób automatyczny za pomocą aplikacji w języku Python z użyciem biblioteki psycopg2.

4.3 Tworzenie kwerend w języku PL/pgSQL

W tym rozdziale zostaną stworzone wszystkie kwerendy opisane w podrozdziale 3.4. Jak już wspominano w rozdziale 3.2, wszystkie kwerendy zostały przygotowane dla starej wersji modelu relacyjnego (rys. 4) w związku z brakiem danych w nowym modelu.

- 1. Sale, które pomieszcza wszystkich zalogowanych użytkowników (jest 8 zalogowanych użytkowników).**

```
SELECT r.room_id AS "Room",
       r.capacity AS "Room capacity"
  FROM rooms r
 WHERE r.capacity >=
    (SELECT COUNT(u.tg_user_id)
      FROM users u
           JOIN users_groups usg ON (u.tg_user_id = usg.user_id)
           JOIN unit_groups ung ON (ung.unit_group_id = usg.group_id)
     WHERE ung.unit_group_id = 13
    );

```

"Room"	"Room capacity"
1 V.5(V-D.49)	20
2 K.36	15
3 F.604	16
4 V.12(V-D.113)	30
5 J.006	14
6 P.22(P-2.500)	60
7 J.004	14
8 J.005	14
9 J.003	14
10 F.602	16

- 2. Średnia ilość studentów na każdy typ grupy.**

```
SELECT groups.group_type      AS "Group type",
       AVG(groups.user_count) AS "Srednia ilosc studentow na kazdy typ grupy"
  FROM (SELECT ung.group_type, COUNT(u.tg_user_id) AS user_count
        FROM users u
             JOIN users_groups usg ON (u.tg_user_id = usg.user_id)
             FULL OUTER JOIN unit_groups ung ON (ung.unit_group_id = usg.group_id)
       GROUP BY ung.unit_group_id
      ) AS groups
 GROUP BY groups.group_type;

```

"Group type"	"Srednia ilosc studentow na kazdy typ grupy"
1 ĆW	1.2777777777777778
2 LAB	0.52941176470588235294
3 PRO	0.39130434782608695652
4 WYK	2.25
5 LEK	0.66666666666666666667
6 SEM-MGR	0.5

Liczby wychodzą tak małe tylko dlatego, że mamy 8 zalogowanych studentów z 3 różnych roczników.

- 3. Administratory, które dołączyły się do aplikacji przed wybraną datą (np. 02.01.2022).**

```
SELECT COUNT(u) AS administratorzy_przed_data
  FROM users u
       JOIN permissions p ON (p.permission_id = u.permission)
 WHERE (p.permission_id = 5 OR p.permission_id = 2)
   AND u.joined_timestamp <= '2022-02-01'::date;

```

"administratorzy_przed_data"
3

4. Ile zajęć minęło od początku semestru (we wszystkich grupach).

```

SELECT COUNT(ac) AS "Zajęc minęły od początku semestru"
FROM activities ac
    JOIN unit_groups ung ON ac.unit_group = ung.unit_group_id
    JOIN usos_units uu 1..n<>1: ON ung.usos_unit_id = uu.usos_unit_id
    JOIN courses c ON uu.course = c.course_id
    JOIN terms t ON c.term_id = t.usos_term_id
WHERE (ac.start_time, ac.end_time) OVERLAPS (t.start_date, NOW());

```

	"Zajęc minęły od początku semestru"
1	1561

5. Pokaż, w których przeprowadzono największą liczbę zajęć (posortowane malejąco względem liczby zajęć).

```

SELECT r.room_id      AS "Room",
       COUNT(a.activity_id) AS "Number of activities in the room"
FROM activities a
    JOIN rooms r ON (a.room = r.room_id)
GROUP BY r.room_id
ORDER BY "Number of activities in the room" DESC;

```

	"Room"	"Number of activities in the room"
1	V.5(V-D.49)	157
2	F.203	146
3	F.502	138
4	L-27.100a	137
5	V.13(V-D.114)	118
6	P.9(P-2.300)	86
7	K.36	80
8	F.505	75
9	Zajęcia_zdalne_WMiFS	57
10	F.402	53

6. Procent zweryfikowanych użytkowników.

```

SELECT CONCAT((CAST(verified_amount AS decimal) / CAST(total_amount AS decimal) * 100)::real,
             '%') AS procent_zweryfikowanych
FROM (SELECT COUNT(users) AS verified_amount
      FROM users
      WHERE verified = TRUE) AS verified_amount,
     (SELECT COUNT(users) AS total_amount
      FROM users) AS total_amount;

```

	procent_zweryfikowanych
1	88.888885%

7. Zadania domowe w wybranym tygodniu (np. zaczynając od 12.01.2022).

```

SELECT hw_short_description AS zadanie,
       hw_full_description AS opis_zadania,
       hw_turn_in_method   AS sposob_oddania,
       hw_due_date          AS termin_oddania
FROM log_homeworks
WHERE (hw_due_date, hw_due_date) OVERLAPS
      ('2022-01-12'::date, '2022-01-12'::date + INTERVAL '1 week');

```

	zadanie	opis_zadania	sposob_oddania	termin_oddania
1	Słówka - UNIT 5		Teams	2022-01-12 00:00:00.000000
2	Wszystkie prace domowe	Wykonać wszystkie prace domowe z całego sem...	IRL	2022-01-16 00:00:00.000000
3	Opis projekt	Przygotować sprawozdanie, które dokładnie o...	Email	2022-01-18 00:00:00.000000

8. W ilu grupach prowadzą prowadzą zajęcia (posortowane malejaco względem ilości grup).

```
SELECT t.first_name, t.last_name, COUNT(ug.unit_group_id) AS num_of_groups
FROM public.teachers t
    JOIN public.group_teacher gt ON t.teacher_usos_id = gt.teacher
    JOIN public.unit_groups ug 1..n<->1: ON gt.unit_group = ug.unit_group_id
    JOIN public.usos_units uu 1..n<->1: ON uu.usos_unit_id = ug.usos_unit_id
    JOIN public.courses c 1..n<->1: ON uu.course = c.course_id
GROUP BY (t.teacher_usos_id, t.first_name, t.last_name)
ORDER BY num_of_groups DESC
LIMIT 10;
```

	first_name	last_name	num_of_groups
1	Paweł	D	12
2	Piotr	H	12
3	Mirosław	M	10
4	Krzysztof	P	9
5	Bohdan	D	8
6	Myroslav	K	7
7	Wiesław	S	7
8	Marek	B	6
9	Michał	W	6
10	Bartosz	K	6

9. Prowadzące, które prowadzą zajęcia dla obu studentów X i Y

```
SELECT DISTINCT t.first_name || ' ' || t.last_name AS "Imie i nazwisko wspólnych prowadzących"
FROM public.teachers t
    JOIN public.group_teacher grt ON t.teacher_usos_id = grt.teacher
    JOIN public.unit_groups ung 1..n<->1: ON grt.unit_group = ung.unit_group_id
    JOIN public.users_groups usg 1<->1..n: ON ung.unit_group_id = usg.group_id
    JOIN public.users u ON usg.user_id = u.tg_user_id
WHERE u.tg_user_id = [REDACTED] 332
INTERSECT
SELECT DISTINCT t.first_name || ' ' || t.last_name
FROM public.teachers t
    JOIN public.group_teacher grt ON t.teacher_usos_id = grt.teacher
    JOIN public.unit_groups ung 1..n<->1: ON grt.unit_group = ung.unit_group_id
    JOIN public.users_groups usg 1<->1..n: ON ung.unit_group_id = usg.group_id
    JOIN public.users u ON usg.user_id = u.tg_user_id
WHERE u.tg_user_id = [REDACTED] 507
ORDER BY "Imie i nazwisko wspólnych prowadzących";
```

	"Imie i nazwisko wspólnych prowadzących"	
1	Adrian	M
2	Andrzej	P
3	Bartosz	K
4	Bohdan	D
5	Dorota	B
6	Grzegorz	O
7	Grzegorz	S
8	Janusz	D
9	Krzysztof	S
10	Marek	B

- 10. Przedmioty prowadzącego.** Dla każdego prowadzącego wypisać listę jego przedmiotów oraz policzyć ilość: zajęć, zadań domowych, sprawdzianów i zarejestrowanych studentów. Dla przedmiotów, gdzie prowadzący występuje również koordynatorem przedmiotu, wpisać słowo "YES". Jeżeli nazwa przedmiotu jest dłuższa od 38 znaków, to obciąć ją po 35 znaku, a na końcu dodać 3 kropki. Posortować malejąco względem ilości zajęć oraz rosnąco względem imienia i nazwiska prowadzącego.

```

SELECT t.first_name || ' ' || t.last_name "Teacher",
       CASE
         WHEN LENGTH(crs.course_name) > 38 THEN SUBSTRING(crs.course_name FOR 35) || '...'
         ELSE crs.course_name
       END                               "Course",
       CASE
         WHEN crs.course_id = cmng.course THEN '-----YES-----'
         ELSE ''
       END                               "Leader?",
       COUNT(DISTINCT usg.user_id)        "Students",
       COUNT(DISTINCT act.activity_id)    "Activities",
       COUNT(DISTINCT lh.homework_id)     "Home tasks",
       COUNT(DISTINCT lt.log_test_id)      "Tests"
FROM public.teachers t
       JOIN public.group_teacher grt ON grt.teacher = t.teacher_usos_id
       JOIN public.unit_groups ung 1..n<>1: ON grt.unit_group = ung.unit_group_id
       JOIN public.usos_units uu 1..n<>1: ON ung.usos_unit_id = uu.usos_unit_id
       JOIN public.courses crs 1..n<>1: ON uu.course = crs.course_id
       JOIN public.activities act 1<>1..n: ON ung.unit_group_id = act.unit_group
       LEFT OUTER JOIN public.log_homeworks lh 1<>0..n: ON act.activity_id = lh.from_activity
       LEFT OUTER JOIN public.log_tests lt 1<>0..n: ON act.activity_id = lt.activity
       LEFT OUTER JOIN public.users_groups usg 1<>0..n: ON ung.unit_group_id = usg.group_id
       LEFT OUTER JOIN public.course_manager cmng ON t.teacher_usos_id = cmng.manager
GROUP BY (t.teacher_usos_id, t.first_name, t.last_name, crs.course_name, crs.course_id, cmng.course)
ORDER BY "Activities" DESC, "Teacher";

```

"Teacher"	"Course"	"Leader?"	"S..."	"A..."	"H..."	"T..."	
Bartosz K	Bazy danych			6	84	2	1
Bohdan D	Metody numeryczne			4	71	0	0
Krzysztof P	Optymalizacja nieliniowa			0	67	0	0
Jacek D	Programowanie obiektowe			0	54	0	0
Wiesław S	LabView - akwizycja danych pomiarowych	-----YES-----		6	53	0	1
Myroslav K	Metody numeryczne	-----YES-----		6	50	0	1
Maciej K	Sztuczna inteligencja			0	49	0	0
Michał W	Projektowanie systemów i sieci komp...			6	45	0	1
Mariusz N	Projektowanie systemów bezpieczeństwa			0	44	0	0
Paweł D	Bazy danych	-----YES-----		6	44	1	0

- 11. Przykład zapytania wywoływanego aplikacją bazodanową.** Wypisać listę zajęć pewnego studenta pomiędzy określonymi datami. Dla każdego zajęcia podać również jego ID, czas rozpoczęcia i zakończenia, salę, typ i numer grupy zajęciowej oraz nazwę przedmiotu. Kod zapytania został podany na rysunku 39, a przykład zastosowania można znaleźć w podrozdziale 4.4.

```

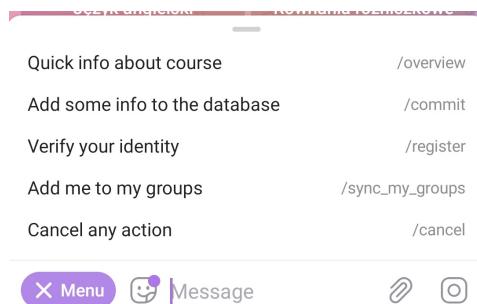
def get_user_activities_details(self, tg_user_id, start_date=None,
                                end_date=None, course_id=None):
    course_id_query, end_date_query, start_date_query = "", "", ""
    query_dict = {"tg_user_id": tg_user_id}
    if start_date:
        start_date_query = " AND act.start_time >= %(start_date)s"
        query_dict["start_date"] = start_date
    else:
        start_date_query = " AND act.start_time >= current_timestamp"
    if end_date:
        end_date_query = "AND act.start_time <= %(end_date)s"
        query_dict["end_date"] = end_date
    if course_id:
        course_id_query = "AND crs.course_id = %(course_id)s"
        query_dict["course_id"] = course_id
    query = (f"SELECT act.activity_id, "
              f"       act.start_time at time zone 'cet', "
              f"       act.end_time at time zone 'cet', "
              f"       act.room, "
              f"       grt.group_type_id, "
              f"       ung.group_number, "
              f"       crs.course_name "
              f"FROM courses crs "
              f"JOIN usos_units ON "
              f"  crs.course_id = usos_units.course "
              f"JOIN unit_groups ung ON "
              f"  ung.usos_unit_id = usos_units.usos_unit_id "
              f"JOIN group_types grt ON "
              f"  ung.group_type = grt.group_type_id "
              f"JOIN users_groups ON "
              f"  ung.unit_group_id = users_groups.group_id "
              f"JOIN activities act ON "
              f"  act.unit_group = ung.unit_group_id "
              f"WHERE users_groups.user_id = %(tg_user_id)s "
              f"  {end_date_query} {course_id_query} {start_date_query} "
              f"ORDER BY act.start_time;")
    df = pd.read_sql(query, self.sqlalchemy_engine, params=query_dict,
                     parse_dates={"start_date": {"format": "%Y-%m-%d"},
                                  "end_date": {"format": "%Y-%m-%d"}})
    df.columns = ["activity_id", "start_time", "end_time", "room",
                  "group_type", "group_number", "course_name"]
    logging.debug(f"{tg_user_id=} {course_id=} {end_date=}")
    return df

```

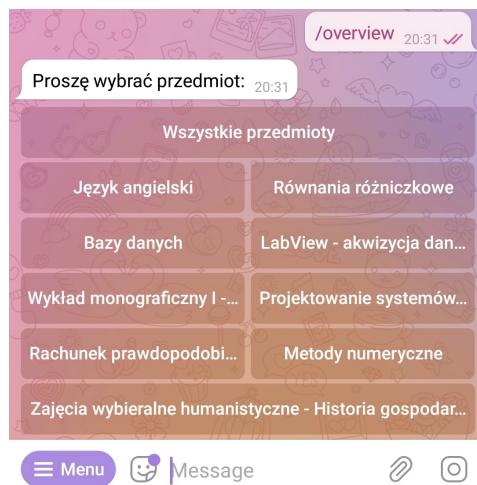
Rysunek 39: Pobieranie listy zajęć użytkownika za pomocą aplikacji w języku Python z użyciem biblioteki psycopg2.

4.4 Przykłady działania aplikacji bazodanowej

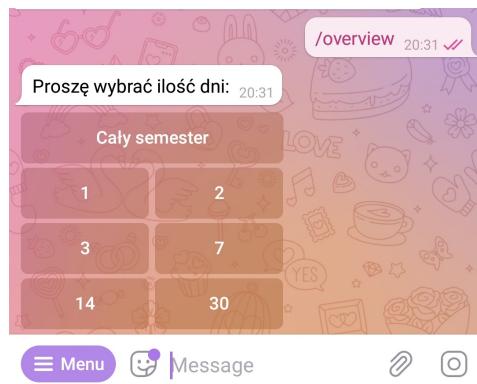
- Po zalogowaniu się do systemu użytkownik ma możliwość wybrać jedno z następujących polecień:



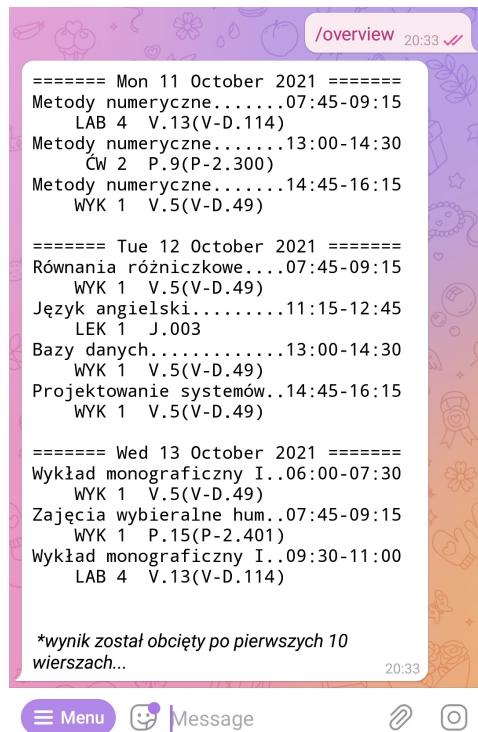
- Polecenie `/overview` pokazuje niektóre informacje z planu zajęć studenta. Użytkownik ma możliwość wybrać przedmiot:



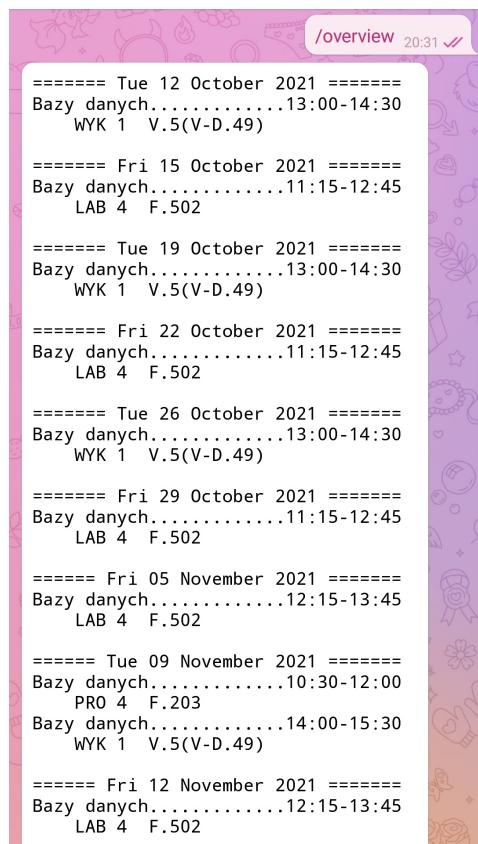
- Możemy wybrać wszystkie przedmioty, żeby zobaczyć swój rozkład zajęć. Wtedy aplikacja zapyta nas wybrać zakres czasowy zajęć:



4. Wynik dla przycisku *Cały semestr* wygląda następująco:



5. Gdybyśmy wybrali w punkcie numer 2 jakiś oddzielny przedmiot (np. Bazy danych), to wynik byłby następujący:



4.5 Doświadczenia, wynikające z realizacji projektu

W trakcie realizacji projektu nauczyliśmy się pracować z API USOSa i Telegramu, korzystać z systemów bazodanowych PostgreSQL i RethinkDB, używać asynchronicznej bibliotek asyncio i aiogram, zbierać logi i obserwować działalność aplikacji.

W czasie prowadzenia projektu wszyscy uczestnicy zadeklarowali pozytywne odczucia na temat możliwości jego realizacji w grupie programistów, pozwalając na zdobycie doświadczenia przy pracy w zespole.

Zrozumieliśmy również, jak ważne jest podejście *top-down network design* w przygotowaniu jakichkolwiek projektów. Bez względu na to, że takie podejście czasami wydaje się zbyt utrudniającym i niepotrzebnym, zwykle tylko zmniejsza łączną ilość pracy, w porównaniu do podejścia *down-top network design*.

W czasie realizacji naszych zadań mieliśmy także możliwość korzystania z platformy Github, gdzie organizowaliśmy wszystkie nasze prace, od trzymania naszego projektu w repozytorium, jak i wnoszenia poprawek do dokumentacji i baz danych przy pomocy *issue'ów*.

Patrząc z perspektywy czasu, doszliśmy do wniosku, że wybranie własnego projektu do realizacji, było bardzo dobrym pomysłem, pod względem możliwości rozwoju aplikacji bazodanowej, która w założeniach ma funkcjonować jako rzeczywisty "produkt-- zwiększając diametralnie ilość starań, aby baza danych, jak i cała aplikacja były starannie i profesjonalnie dobrane, aby mogły się w przyszłości skalować wraz z większą ilością *developmentu*, jak i ilości użytkowników.

Przygotowanie tak rozbudowanego projektu pozwoliło nauczyć się poprawnie szacować swoje możliwości i czas, potrzebny do realizacji określonych zadań. Na diagramie scenariuszy użycia (rys. 1) można zauważyć, jak dużo różnych rzeczy planowaliśmy zrobić. Natomiast przeglądając podrozdział 4.4, rozumiemy, jak szacunkowo mało udało się zrobić w trakcie semestru.

Łączna ilość wierszy we wszystkich tabelach	2299
Ilość wierszy tego sprawozdania w formacie LATEX	1546
Ilość spędzonych godzin pracy studentów	≈ 120
Ilość wierszy ostatecznego kodu źródłowego projektu ¹	3354

Tabela 31: Wybrane metryki opracowanego projektu

¹Nie licząc tego sprawozdania.

5 Bibliografia

- [1] AIOGram.
<https://github.com/aiogram/aiogram/>
- [2] PostgreSQL – The World’s Most Advanced Open Source Relational Database.
<https://www.postgresql.org/>
- [3] psycopg2 – Python-PostgreSQL Database Adapter.
<https://github.com/psycopg/psycopg2/>
- [4] Python.
<https://www.python.org/>
- [5] RethinkDB – The open-source database for the realtime web.
<https://rethinkdb.com/>
- [6] Telegram – a new era of messaging.
<https://telegram.org/>
- [7] Telegram Bot API.
<https://core.telegram.org/bots/>
- [8] PRz USOS API.
<https://usosapps.prz.edu.pl/developers/api/>