

Projekt Bazy Danych

System zarządzania sprzętem koła naukowego

Emilia Szymańska
Igor Zieliński

Grupa: środa 9¹⁵ – 11⁰⁰
Prowadzący: dr inż. Roman Ptak
Marzec 2020

Spis treści

1	Wstęp	3
1.1	Cel projektu	3
1.2	Zakres projektu	3
2	Analiza wymagań	3
2.1	Opis działania i schemat logiczny systemu	3
2.2	Wymagania funkcjonalne	3
2.2.1	Niezałogowany użytkownik	3
2.2.2	Członek koła	3
2.2.3	Administrator	4
2.3	Wymagania niefunkcjonalne	4
2.3.1	Wykorzystywane technologie i narzędzia	4
2.3.2	Wymagania dotyczące rozmiaru bazy danych	4
2.3.3	Wymagania dotyczące bezpieczeństwa systemu	4
3	Projekt systemu	5
3.1	Projekt bazy danych	5
3.1.1	Analiza rzeczywistości i uproszczony model konceptualny	5
3.1.2	Model logiczny i normalizacja	5
3.1.3	Model fizyczny i ograniczenia integralności danych	6
3.1.4	Inne elementy schematu – mechanizmy przetwarzania danych	7
3.1.5	Projekt mechanizmów bezpieczeństwa na poziomie bazy danych	8
3.2	Projekt aplikacji użytkownika	9
3.2.1	Architektura aplikacji i diagramy projektowe	9
3.2.2	Interfejs graficzny i struktura menu	10
3.2.3	Projekt wybranych funkcji systemu	10
3.2.4	Metoda podłączania do bazy danych – integracja z bazą danych	10
3.2.5	Projekt zabezpieczeń na poziomie aplikacji	10
4	Implementacja systemu bazy danych	11
4.1	Tworzenie tabel i definiowanie ograniczeń	11
4.2	Implementacja mechanizmów przetwarzania danych	12
4.3	Implementacja uprawnień i innych zabezpieczeń	13
4.4	Testowanie bazy danych na przykładowych danych	15

5	Implementacja i testy aplikacji	21
5.1	Instalacja i konfigurowanie systemu	21
5.2	Instrukcja użytkowania aplikacji	23
5.3	Testowanie opracowanych funkcji systemu	32
5.4	Omówienie wybranych rozwiązań programistycznych	37
5.4.1	Implementacja interfejsu dostępu do bazy danych	37
5.4.2	Implementacja wybranych funkcjonalności systemu	40
5.4.3	Implementacja mechanizmów bezpieczeństwa	42
6	Podsumowanie i wnioski	44

1 Wstęp

1.1 Cel projektu

Celem projektu jest stworzenie bazy danych wraz z aplikacją dostępową umożliwiającą zarządzanie dystrybucją komponentów elektronicznych oraz przyrządów pomiarowych w kole naukowym zajmującym się projektowaniem i wykonywaniem układów elektronicznych oraz płytek PCB.

1.2 Zakres projektu

1. Określenie celu oraz wymagań wobec projektu.
2. Zaprojektowanie bazy danych: stworzenie modelu conceptualnego, logicznego oraz fizycznego.
3. Implementacja bazy danych oraz aplikacji użytkownika.
4. Przeprowadzenie testów aplikacji dostępowej opartej na bazie danych.
5. Wyciągnięcie wniosków, podsumowanie projektu – prezentacja multimedialna.

2 Analiza wymagań

2.1 Opis działania i schemat logiczny systemu

System ma umożliwiać zalogowanie się użytkownika (z uprawnieniami administratora lub członka koła), a następnie przegląd dostępnego sprzętu w kole naukowym. Ponadto każdy użytkownik ma zgodę lub zakaz na korzystanie z danego sprzętu. Sprzęt dzieli się na dwa rodzaje: „zużywalny” i „niezużywalny”. Z poziomu członka koła można pobrać sprzęt „zużywalny” lub wypożyczyć „niezużywalny” na określony czas. Z poziomu administratora można dodatkowo zwiększać ilość danego sprzętu, a także zmieniać uprawnienia pozostałym użytkownikom. Jeśli ilość sprzętu „zużywalnego” spadnie do określonej wartości krytycznej bądź użytkownik przetrzyma sprzęt „niezużywalny”, administrator zostaje powiadomiony o zdarzeniu.

2.2 Wymagania funkcjonalne

2.2.1 Niezalogowany użytkownik

1. Możliwość zalogowania się do systemu.

2.2.2 Członek koła

1. Wyszukiwanie komponentów elektronicznych i przyrządów po nazwie.
2. Wypożyczanie sprzętu „niezużywalnego” na określony czas.
3. Pobór sprzętu „zużywalnego”.
4. Tworzenie zamówienia hurtowo rezerwującego/pobierającego wymienione sprzęty lub zwracającego informację o brakach.
5. Obsługa zamówień hurtowych.
6. Wysłanie informacji do administratora o potrzebie dokupienia danego sprzętu.
7. Zmiana parametrów konta.

2.2.3 Administrator

1. Otrzymywanie powiadomień o potrzebie dokupienia danego sprzętu lub przetrzymaniu go przez członka koła.
2. Poszerzanie lub modyfikacja asortymentu koła.
3. Dodawanie, usuwanie kont członków koła.

2.3 Wymagania niefunkcjonalne

2.3.1 Wykorzystywane technologie i narzędzia

1. DBMS: MySQL.
2. Język programowania: SQL, C++, Python.
3. Program do modelowania baz danych: MySQL Workbench.
4. System operacyjny: Windows.
5. Aplikacja: desktopowa.

2.3.2 Wymagania dotyczące rozmiaru bazy danych

- od 2 do 10 użytkowników z uprawnieniami administratora,
- od 10 do 60 członków koła,
- od 50 do 100 rodzajów sprzętu „zużywalnego” i „niezużywalnego”,
- od 10 do 100 wypożyczeń sprzętu „niezużywalnego” co semestr (dane usuwane co semestr akademicki),
- od 100 do 1000 pobrań sprzętu „zużywalnego” na semestr (dane usuwane co semestr akademicki).

2.3.3 Wymagania dotyczące bezpieczeństwa systemu

1. Konieczność logowania się użytkownika w celu dostania się do bazy.
2. System logowania za pomocą systemu typu login i hasło.
3. Podział uprawnień na administratorów i członków koła.

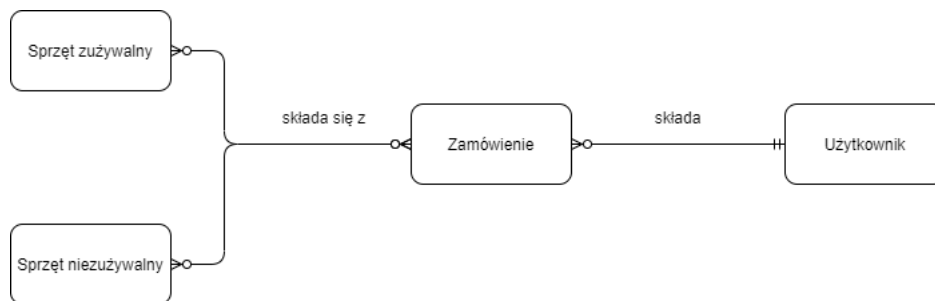
3 Projekt systemu

3.1 Projekt bazy danych

3.1.1 Analiza rzeczywistości i uproszczony model konceptualny

Projekt zakłada aplikację dostępową połączoną z bazą danych umożliwiającą składanie zamówień na wybrany sprzęt z koła naukowego. Dodatkowymi funkcjami - takimi jak nadawanie uprawnień przez administratora, usuwanie przez niego sprzętu, modyfikacja - w tym punkcie się nie zajmujemy. Koncentrujemy się za to na wyodrębnieniu samych encji, niezbędnych do późniejszych implementacji. Zarówno przeciętny członek koła, jak i administrator (użytkownicy) mogą (ale nie muszą) składać zamówienia. Zamówienie składa się ze sprzętu - może zawierać tylko sprzęt zużywalny, tylko nieużywalny lub oba typy jednocześnie. Na etapie modelu konceptualnego nie rozdzielamy zamówień na oddzielne typy, stanie się to w kolejnym etapie.

Poniżej został przedstawiony uproszczony model konceptualny wraz ze wstępnymi związkami.



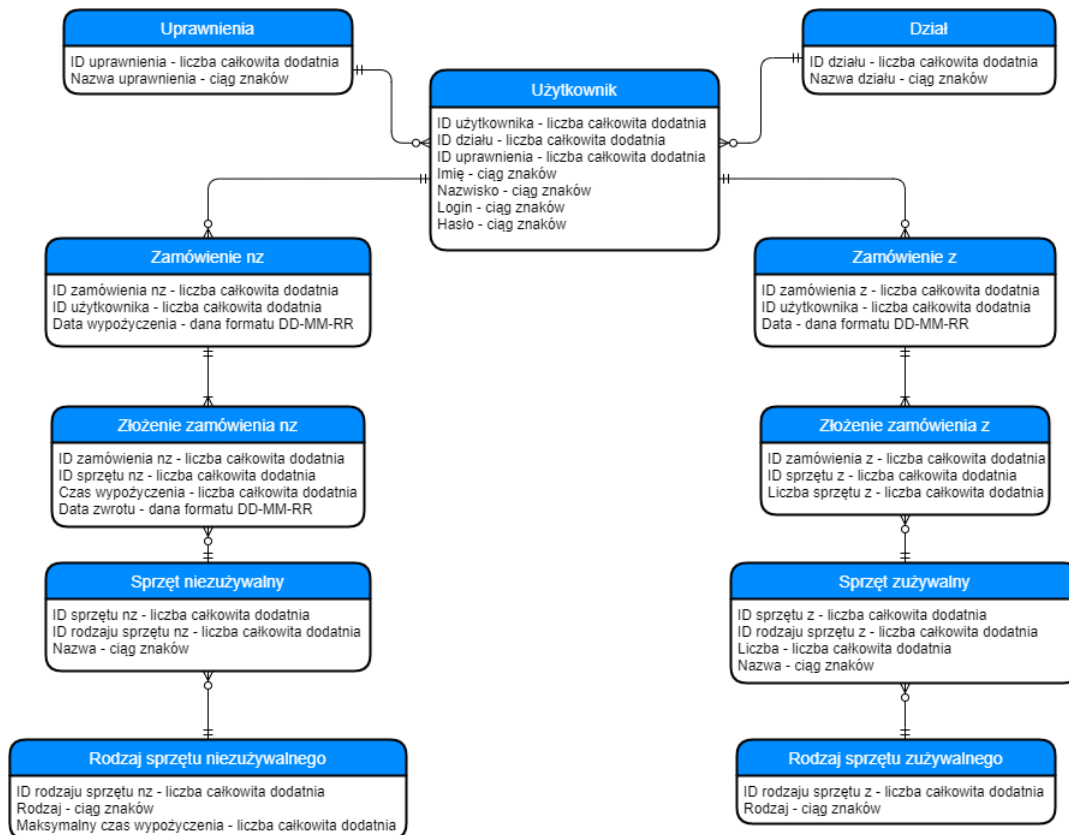
Rysunek 1: Model konceptualny

3.1.2 Model logiczny i normalizacja

W modelu logicznym rozdzieliliśmy już typy zamówień na te dotyczące sprzętu zużywalnego jak i nieużywalnego. Atrybut „ID uprawnienia” w encji „Użytkownik” odpowiada za rozpoznawanie dostępnych przez użytkownika funkcji, w zależności od statusu administratora lub zwykłego członka koła. Do tego - a także do rozpoznawania działu użytkownika - użyliśmy tabel słownikowych. W tym kroku sprecyzowaliśmy także, jakie są i w jaki sposób będziemy zapamiętywać atrybuty - podaliśmy, jakie typy wartości będą potrzebne, jednak bez określania dokładnych nazw typów zależnych od języka, w którym bazę danych zaimplementujemy. Dokonałmy także normalizacji, czyli usunięcia nadmiarowości informacji z naszego modelu. Normalizacja jest widoczna np. przy zamówieniu - nie podajemy już tam imienia i nazwiska osoby wypożyczającej, ale jej ID, dzięki czemu możemy znaleźć po ID osobę wypożyczającą i dostać się do interesujących nas danych, bez nadmiaru atrybutów w samym zamówieniu. Normalizację zastosowaliśmy także poprzez wprowadzenie tabel słownikowych dla rodzajów sprzętów, działów i uprawnień, a także zastępując związki wiele do wielu relacjami z tabelami pośredniczącymi.

Sam zwrot sprzętu nieużywalnego będzie realizowany za pomocą funkcji, której argumentem będzie ID sprzętu (dla każdego unikatowe, dostępne w zrealizowanym zamówieniu lub poprzez wysłanie odpowiedniego zapytania), dlatego zwrotu nie rozpatrujemy jako oddzielnej encji. W zamówieniu wyróżniliśmy atrybut „Data zwrotu” i jego wartość - zmieniana za pomocą funkcji zwrotu z NULL na datę aktualną - będzie decydowała o tym, czy można dalej sprzęt wypożyczać czy nie. Zamówienie sprzętu zużywalnego automatycznie po jego złożeniu jest uznawane za zamówienie zrealizowane.

Poniżej znajduje się model logiczny naszego projektu bazy danych.



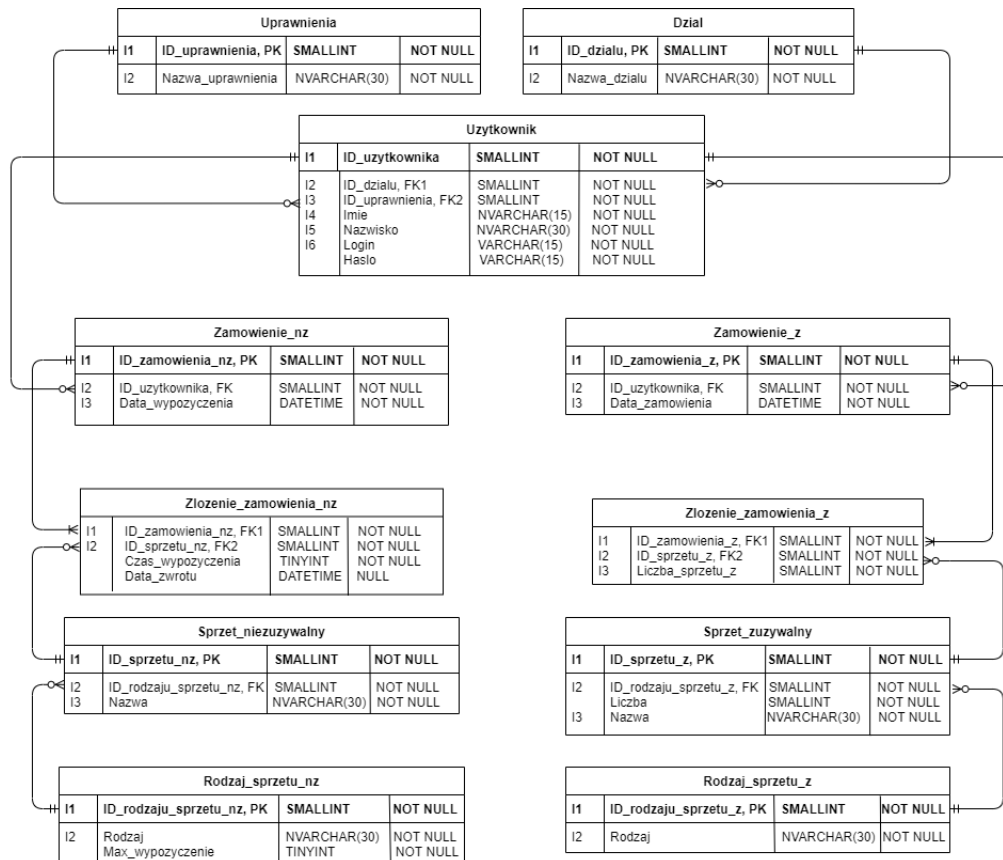
Rysunek 2: Model logiczny

3.1.3 Model fizyczny i ograniczenia integralności danych

W modelu fizycznym określiliśmy już dokładnie, uwzględniając składnię języka SQL, jakie typy danych będą nam potrzebne oraz wskazaliśmy klucze główne (Primary Key) oraz obce (Foreign Key) w encjach. Bazę danych będziemy implementować za pomocą MySQL Workbench z użyciem silnika InnoDB. Dodatkowo musieliśmy wprowadzić ograniczenia:

- wszystkie dane o typie SMALLINT i TINYINT w naszym przypadku muszą mieć wartości nieujemne,
- ilość sprzętu w zamówieniu nie może przekraczać ilości sprzętu na stanie,
- długość wypożyczenia sprzętu nieużywalnego nie może przekraczać maksymalnej dozwolonej długości,
- nie można wypożyczyć sprzętu, który nie jest w danej chwili dostępny,
- nie można zwrócić sprzętu, który nie jest w danej chwili wypożyczony,
- w przypadku odwołania się do pustych rekordów wyświetlany jest komunikat o błędzie,
- w przypadku usunięcia użytkownika, powiązane z nim zamówienia nie są kasowane, w szczególności zamówienia sprzętu nieużywalnego zostają zaznaczone jako zrealizowane,
- w przypadku usunięcia sprzętu nieużywalnego z bazy danych podczas jego wypożyczenia jest on automatycznie zwracany i od razu usuwany z bazy sprzętu nieużywalnego, dzięki czemu nie można go dalej wypożyczać.

Poniżej znajduje się schemat przedstawiający model fizyczny naszego projektu wraz z indeksami i kluczami.



Rysunek 3: Model fizyczny

3.1.4 Inne elementy schematu – mechanizmy przetwarzania danych

□ Sekwencje

Sekwencje realizujemy poprzez zadeklarowanie wszystkich ID (kluczy) jako typu AUTONUMBER.

□ Przykładowe widoki

→ Nazwa widoku: `dostepny_sprzet_nz`

Użyte tabele: `Sprzet_niezuzywalny`, `Zamowienie_nz`, `Zlozenie_zamowienia_nz`

Tabela wynikowa: trzy kolumny - nazwa i rodzaj, maksymalny czas wypożyczenia

Tabela zawiera informacje o nazwach, rodzajach i maksymalnym czasie wypożyczenia dostępnych sprzętów niezużywalnych.

→ Nazwa widoku: `uzytkownicy_wypozyczajacy`

Użyte tabele: `Zamowienie_nz`, `Uzytkownik`, `Sprzet_niezuzywalny`, `Zlozenie_zamowienia_nz`

Tabela wynikowa: pięć kolumn - imię, nazwisko, dział, nazwa sprzętu, przewidywany czas zwrotu.

Tabela zawiera dane osób oraz sprzętu, który został wypożyczony, ale nie został zwrócony wraz z ich datą przewidywanego zwrotu.

→ Nazwa widoku: `uzytkownicy_zadluzeni`

Użyte tabele: `Zamowienie_nz`, `Uzytkownik`, `Sprzet_niezuzywalny`, `Zlozenie_zamowienia_nz`

Tabela wynikowa: pięć kolumn - imię, nazwisko, dział, nazwa sprzętu, przewidywany czas zwrotu.

Tabela zawiera dane osób oraz sprzętu, który został wypożyczony, ale nie został zwrócony i ponadto, okres jego wypożyczenia przekroczył zadeklarowaną przy zamówieniu liczbę dni (data, przed nadejściem której należało zwrócić sprzęt, jest również podana).

- Nazwa widoku: `uzytkownicy_i_dzial`
 Użyte tabele: `Uzytkownik`, `Dzial`
 Tabela wynikowa: trzy kolumny - imię, nazwisko, dział
 Widok informuje o użytkownikach i działach, do których należą.

□ Przykładowe procedury składowane

- `zwrot_sprzetu_nz`
 Parametry: ID sprzętu, który chcemy zwrócić.
 Najpierw wywołujemy sprawdzenie, czy dany sprzęt jest wypożyczony (wyszukujemy w tabeli `Zamowienie_nz` zamówienia z ID z parametru i sprawdzamy, czy atrybut `Data_zwrotu` = NULL). Jeśli ten sprzęt istnieje w którymś zamówieniu i warunek jest spełniony, ustawiamy atrybut daty zwrotu na aktualną datę. Wówczas przy następnym wyszukiwaniu sprzętu będzie on widoczny jako dostępny.
- `dodanie_elementu_zamowienia_z`
 Parametry: numer zamówienia, nazwa sprzętu zużywanego, liczba
 Wywołujemy sprawdzenie, czy sprzęt istnieje w bazie i czy liczba zamówionego sprzętu (attribut `Liczba_sprzetu_z` z tabeli `Zamowienie_z`) jest mniejsza bądź równa liczbie dostępnego sprzętu o tej nazwie (attribut `Liczba` z tabeli `Sprzet_zuzywalny`). Jeśli warunek ten jest spełniony, wówczas powstaje rekord w tabelach `Zamowienie_z` i `Zlozenie_zamowienia_z` z odpowiednimi wartościami pod atrybutami, od liczby dostępnego sprzętu odejmowana jest liczba sprzętu zamówionego, a zamówienie zostaje zrealizowane z pobraniem aktualnej daty. W przeciwnym przypadku pojawi się komunikat o błędzie.
- `dodanie_elementu_zamowienia_z`
 Parametry: numer zamówienia, nazwa sprzętu, żądany czas wypożyczenia
 Wywołujemy sprawdzenie, czy sprzęt istnieje w bazie (w tabeli `Sprzet_zuzywalny`) i czy nikt obecnie nie wypożycza sprzętu (wyszukujemy w tabeli `Zamowienie_nz` zamówienia z ID z parametru i sprawdzamy, czy atrybut `Data_zwrotu` != NULL). Następnie sprawdzany jest warunek, czy żądany czas wypożyczenia (parametr) jest mniejszy bądź równy maksymalnej długości wypożyczenia (attribut `Max_wypozyczenie` z tabeli `Rodzaj_sprzetu_nz` odpowiadający ID sprzętu). Jeśli powyższe warunki są spełnione, to powstaje rekord w tabelach `Zamowienie_nz` oraz `Zlozenie_zamowienia_z` z odpowiednimi wartościami pod atrybutami, pobrana zostaje aktualna data do wartości `Data_wypozyczenia`, a `Data_zwrotu` = NULL. W przeciwnym przypadku pojawi się odpowiedni komunikat o błędzie.

□ Wyzwalacze

Wyzwalacze nie zostaną przez nas wykorzystane, gdyż ich funkcjonalność będzie przeniesiona do procedur.

3.1.5 Projekt mechanizmów bezpieczeństwa na poziomie bazy danych

Autoryzacja polega na rozgraniczeniu uprawnień użytkowników przez atrybut „Uprawnienia”. Poniżej znajduje się diagram z tabelami występującymi w projekcie. W wierszu z nazwą tabeli mamy podział kolumn na A - uprawnienia administratora - i C - uprawnienia członka koła. Do każdego atrybutu są zdefiniowane prawa dostępu: A (All = Read, Write) oraz R (Read). Myślnik oznacza, że do danego atrybutu nie ma dostępu - ani administrator, ani członek koła nie potrzebuje bezpośrednio dostawać się do ID będących tutaj kluczami. Pogrubione zostały nazwy atrybutów, które są unikatowe - oprócz kluczy są to również login oraz nazwa sprzętu nieużywanego (zakładamy, że każdy sprzęt nieużywalny jest niepowtarzalny, gdyż rzadko kiedy koło naukowe potrzebuje dwóch identycznych sprzętów, każdy z założenia ma mieć inne zastosowanie niż poprzednik).

Uprawnienia	A	C
ID_uprawnienia	-	-
Nazwa_uprawnienia	A	R

Dział	A	C
ID_dzialu	-	-
Nazwa_dzialu	A	R

Uzytkownik	A	C
ID_uzytkownika	-	-
ID_dzialu	-	-
ID_uprawnienia	-	-
Imie	A	R
Nazwisko	A	R
Login	A	R
Haslo	A	A

Zamowienie_nz	A	C
ID_zamowienia_nz	-	-
ID_uzytkownika	-	-
Data_wypozyczenia	R	R

Zamowienie_z	A	C
ID_zamowienia_z	-	-
ID_uzytkownika	-	-
Data_zamowienia	R	R

Sprzet_niezuzywalny	A	C
ID_sprzetu_nz	-	-
ID_rodzaju_sprzetu_nz	-	-
Nazwa	A	R

Sprzet_zuzywalny	A	C
ID_sprzetu_n	-	-
ID_rodzaju_sprzetu_n	-	-
Liczba	A	R
Nazwa	A	R

Rodzaj_sprzetu_nz	A	C
ID_rodzaju_sprzetu_nz	-	-
Rodzaj	A	R
Max_wypozyczenie	A	R

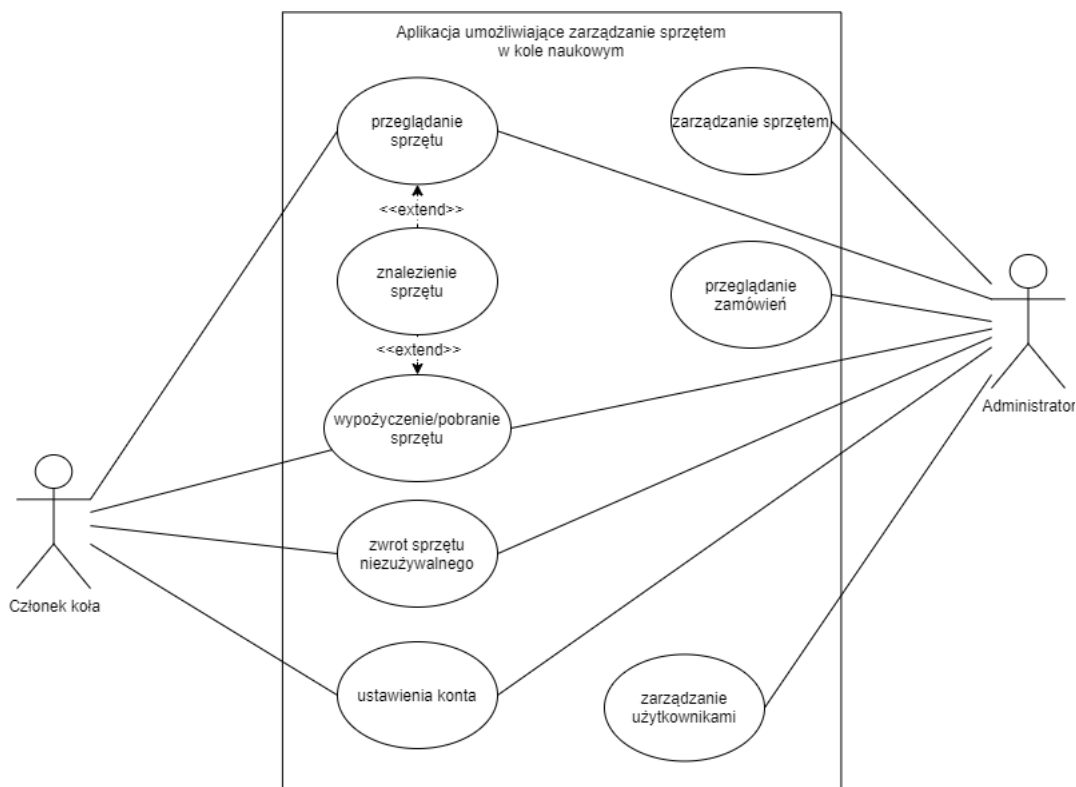
Rodzaj_sprzetu_z	A	C
ID_rodzaju_sprzetu_z	-	-
Rodzaj	A	R

Rysunek 4: Tabele z uprawnieniami

3.2 Projekt aplikacji użytkownika

3.2.1 Architektura aplikacji i diagramy projektowe

Ten punkt projektowania realizujemy za pomocą diagramu przypadków użycia:

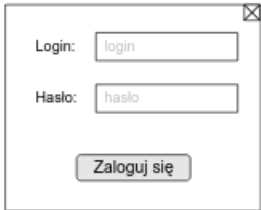


Rysunek 5: Diagram Przypadków Użycia

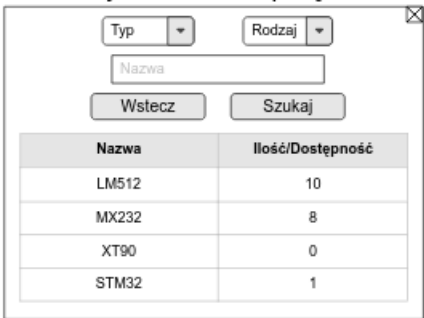
Nie uwzględniamy tutaj modyfikacji parametrów konta czy też - w przypadku administratora - zarządzanie użytkownikami, gdyż te opcje bezpośrednio nie realizują samej funkcjonalności wypożyczania/poboru sprzętu, za to chcemy podkreślić, że administrator może modyfikować sprzęt, co wpływa na stan zasobów koła naukowego, a w konsekwencji na wybór sprzętu przez członka koła.

3.2.2 Interfejs graficzny i struktura menu

Ekran logowania

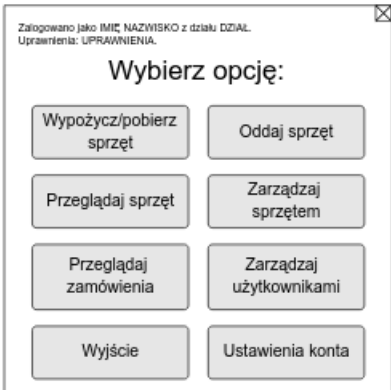


Wyszukiwanie sprzętu



Nazwa	Ilość/Dostępność
LM512	10
MX232	8
XT90	0
STM32	1

Zalogowany jako administrator



Zalogowany jako zwykły użytkownik



Rysunek 6: Projekt interfejsu graficznego

3.2.3 Projekt wybranych funkcji systemu

Cała funkcjonalność aplikacji jest realizowana za pomocą procedur składowych bazy danych.

3.2.4 Metoda podłączania do bazy danych – integracja z bazą danych

W przypadku naszego projektu do integracji z bazą danych będziemy korzystać ze standardu ODBC.

3.2.5 Projekt zabezpieczeń na poziomie aplikacji

Zabezpieczeniem zastosowanym w naszym projekcie jest konieczność zalogowania się użytkownika, by mieć możliwość składania zamówień, wyświetlania sprzętu itp. Hasła użytkowników będą szyfrowane z użyciem funkcji haszującej i ciągu zaburzającego (soli). Ponadto podział na administratora i zwykłego członka koła zapewnia nam brak modyfikacji w sprzęcie i edycji dotyczących użytkowników przez niepożądane osoby.

4 Implementacja systemu bazy danych

4.1 Tworzenie tabel i definiowanie ograniczeń

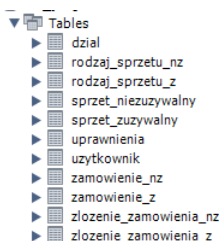
Za pomocą narzędzia MySQL Workbench z silnikiem InnoDB tworzymy naszą bazę danych. Tabele, relacje oraz definiowanie ograniczeń realizujemy przy pomocy skryptu w polu „Query”. Tutaj jest przykładowy kod użyty do generowania trzech tabel - „dział”, „uzytkownik” i „uprawnienia”.

```
1 • CREATE TABLE `dzial` (  
2   `id_dzialu` smallint NOT NULL AUTO_INCREMENT,  
3   `nazwa_dzialu` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,  
4   PRIMARY KEY (`id_dzialu`),  
5   UNIQUE KEY `nazwa_dzialu_UNIQUE` (`nazwa_dzialu`)  
6 ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;  
7  
8 • CREATE TABLE `uprawnienia` (  
9   `id_uprawnienia` smallint NOT NULL AUTO_INCREMENT,  
10  `nazwa_uprawnienia` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,  
11  PRIMARY KEY (`id_uprawnienia`),  
12  UNIQUE KEY `nazwa_uprawnienia_UNIQUE` (`nazwa_uprawnienia`)  
13 ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;  
14
```

Rysunek 7: Tworzenie tabel „dział” oraz „uprawnienia”

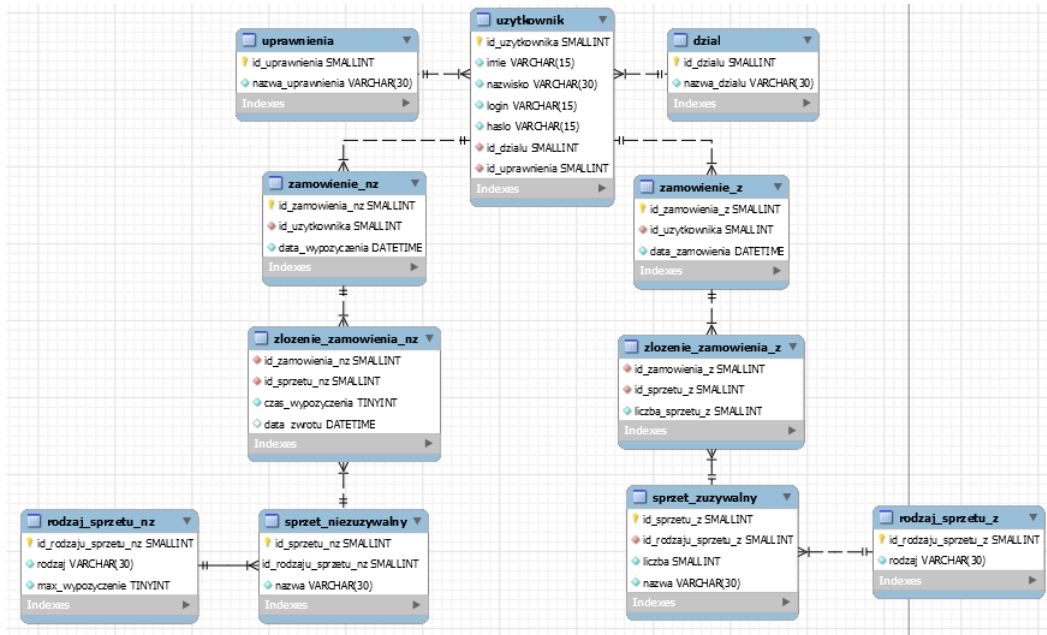
```
15 • CREATE TABLE `uzytkownik` (  
16  `id_uzytkownika` smallint NOT NULL AUTO_INCREMENT,  
17  `imie` varchar(15) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,  
18  `nazwisko` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,  
19  `login` varchar(15) NOT NULL,  
20  `haslo` varchar(15) NOT NULL,  
21  `id_dzialu` smallint NOT NULL,  
22  `id_uprawnienia` smallint NOT NULL,  
23  PRIMARY KEY (`id_uzytkownika`),  
24  UNIQUE KEY `login_UNIQUE` (`login`),  
25  KEY `imie_INDEX` (`imie`),  
26  KEY `nazwisko_INDEX` (`nazwisko`),  
27  KEY `id_dzialu_INDEX` (`id_dzialu`),  
28  KEY `id_uprawnienia_INDEX` (`id_uprawnienia`),  
29  CONSTRAINT `fk_dzial` FOREIGN KEY (`id_dzialu`) REFERENCES `dzial` (`id_dzialu`) ON DELETE CASCADE ON UPDATE CASCADE,  
30  CONSTRAINT `fk_uprawnienia` FOREIGN KEY (`id_uprawnienia`) REFERENCES `uprawnienia` (`id_uprawnienia`)  
31 ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;  
32
```

Rysunek 8: Tworzenie tabeli „uzytkownik”



Rysunek 9: Tabele w bazie danych

Po wygenerowaniu wszystkich tabel i relacji użyliśmy opcji „Reverse engineer”, która na podstawie zawartości naszej bazy danych wygenerowała model fizyczny. Zrobiliśmy to w celu weryfikacji poprawności i możliwości porównania z projektowym modelem fizycznym. Poniżej znajduje się wygenerowany model, który zgadza się z zakładanym modelem.



Rysunek 10: „Reverse engineer”

4.2 Implementacja mechanizmów przetwarzania danych

Głównymi mechanizmami przetwarzania danych, jakimi się zajęliśmy, są procedury składowane oraz widoki (funkcjonalność wyzwalaczy jest zawarta w procedurach, a sekwencje realizujemy za pomocą autonumeracji). Poniżej zostały pokazane przykładowe kody generujące widoki oraz procedury, wyniki ich działania zostaną ukazane w podpunkcie dotyczącym testowania bazy danych.

```

1 • CREATE VIEW `dostepny_sprzet_z` AS
2     SELECT
3         `rodzaj_sprzetu_z`.`rodzaj` AS `rodzaj`,
4         `sprzet_zuzywalny`.`nazwa` AS `nazwa`,
5         `sprzet_zuzywalny`.`liczba` AS `liczba`
6     FROM
7         (`sprzet_zuzywalny`
8         JOIN `rodzaj_sprzetu_z` ON ((`sprzet_zuzywalny`.`id_rodzaju_sprzetu_z` = `rodzaj_sprzetu_z`.`id_rodzaju_sprzetu_z`)))
9     WHERE
10        (`sprzet_zuzywalny`.`liczba` > 0);

```

Rysunek 11: Tworzenie widoku „dostepny_sprzet_z”

```

12 • CREATE VIEW `zamowienia_nz_zwrocone` AS
13     SELECT
14         `zamowienia_nz_wszystkie`.`nazwisko` AS `nazwisko`,
15         `zamowienia_nz_wszystkie`.`imię` AS `imię`,
16         `zamowienia_nz_wszystkie`.`numer zamówienia` AS `numer zamówienia`,
17         `zamowienia_nz_wszystkie`.`rodzaj sprzętu` AS `rodzaj sprzętu`,
18         `zamowienia_nz_wszystkie`.`nazwa sprzętu` AS `nazwa sprzętu`,
19         `zamowienia_nz_wszystkie`.`data wypożyczenia` AS `data wypożyczenia`,
20         `zamowienia_nz_wszystkie`.`termin zwrotu` AS `termin zwrotu`,
21         `zamowienia_nz_wszystkie`.`data zwrotu` AS `data zwrotu`
22     FROM
23         `zamowienia_nz_wszystkie`
24     WHERE
25         (`zamowienia_nz_wszystkie`.`data zwrotu` IS NOT NULL);

```

Rysunek 12: Tworzenie widoku „zamowienia_nz_wszystkie”

```

1  DELIMITER $$
2 • CREATE PROCEDURE `zmien_login`(login_uzytkownika VARCHAR(15), nowy_login VARCHAR(15))
3  BEGIN
4      UPDATE uzytkownik
5      SET login = nowy_login
6      WHERE login = login_uzytkownika;
7  END$$

```

Rysunek 13: Tworzenie procedury „zmien_login”

```

1  DELIMITER $$
2
3 • CREATE PROCEDURE `zwrot_sprzetu_nz`( nazwa_sprzetu varchar(30))
4  BEGIN
5      declare id_sprzetu_zwracanego tinyint;
6
7      set id_sprzetu_zwracanego = (select id_sprzetu_nz from sprzet_niezuzywalny where nazwa = nazwa_sprzetu);
8
9      update zlozenie_zamowienia_nz
10     set data_zwrotu = curdate()
11     where (id_sprzetu_nz = id_sprzetu_zwracanego and data_zwrotu is null);
12 END$$
13
14 DELIMITER ;

```

Rysunek 14: Tworzenie widoku „zwrot_sprzetu_nz”

4.3 Implementacja uprawnień i innych zabezpieczeń

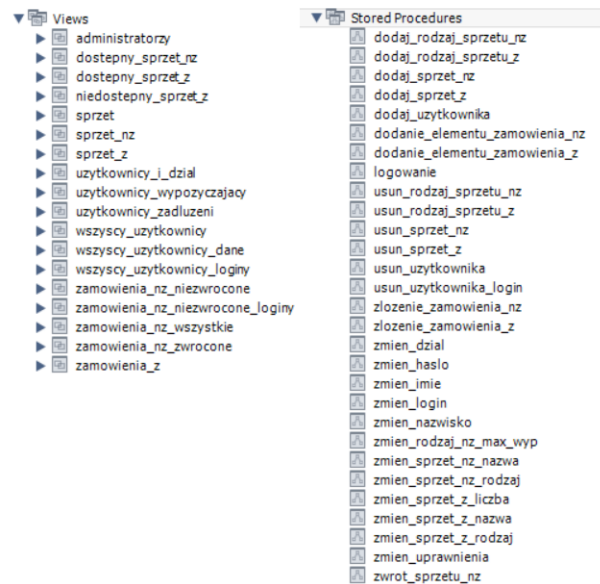
W naszym projekcie istnieją dwa rodzaje uprawnień: uprawnienia administratora i uprawnienia członka koła. W zależności od uprawnienia można mieć dostęp do wybranych procedur. Procedury dotyczące zmian zawartości pól, które w punkcie 3.1.5 różnią się w przypadku administratora i członka koła, są wywoływane z dodatkowym parametrem określającym login osoby wywołującej procedurę. Jest sprawdzany warunek, czy osoba wywołująca ma odpowiednie uprawnienia i jeśli tak, to wykonywane są następne kroki. Poniżej widoczny jest kod takiej procedury, która wymaga podania loginu użytkownika wywołującego. Przykład działania tego zabezpieczenia jest widoczny w następnym punkcie.

```

1 DELIMITER $$
2 • CREATE PROCEDURE `usun_uzytkownika`(login_wywołującego varchar(15), imie_uzytk VARCHAR(15), nazwisko_uzytk VARCHAR(30))
3 BEGIN
4     DECLARE zmienna_uzytk SMALLINT;
5     DECLARE zmienna_uprawnienie_wywołującego VARCHAR(15);
6
7     SET zmienna_uprawnienie_wywołującego = (SELECT nazwa_uprawnienia
8     FROM (`bd_projekt`.`uprawnienia`
9     INNER JOIN `bd_projekt`.`uzytkownik` ON ((`bd_projekt`.`uzytkownik`.`id_uprawnienia` = `bd_projekt`.`uprawnienia`.`id_uprawnienia`)))
10    WHERE (`bd_projekt`.`uzytkownik`.`login` = login_wywołującego));
11
12    SET zmienna_uzytk = (SELECT id_uzytkownika
13    FROM uzytkownik
14    WHERE (`bd_projekt`.`uzytkownik`.`imie` = imie_uzytk) AND (`bd_projekt`.`uzytkownik`.`nazwisko` = nazwisko_uzytk));
15
16    IF zmienna_uprawnienie_wywołującego = "administrator" THEN
17        DELETE FROM uzytkownik WHERE id_uzytkownika = zmienna_uzytk;
18    END IF;
19 END$$
20
21 DELIMITER ;

```

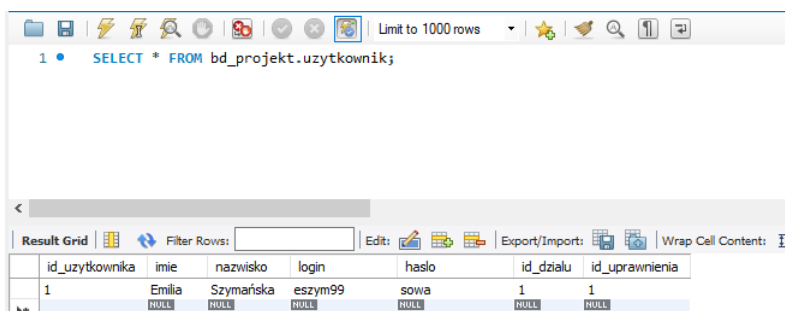
Rysunek 15: Przykładowe zabezpieczenie w bazie danych



Rysunek 16: Procedury i widoki

4.4 Testowanie bazy danych na przykładowych danych

Do przykładowego testowania posłużymy nam procedura dodawania użytkownika, która wymaga podania loginu osoby wywołującej. Tak wygląda początkowo tabela użytkowników:

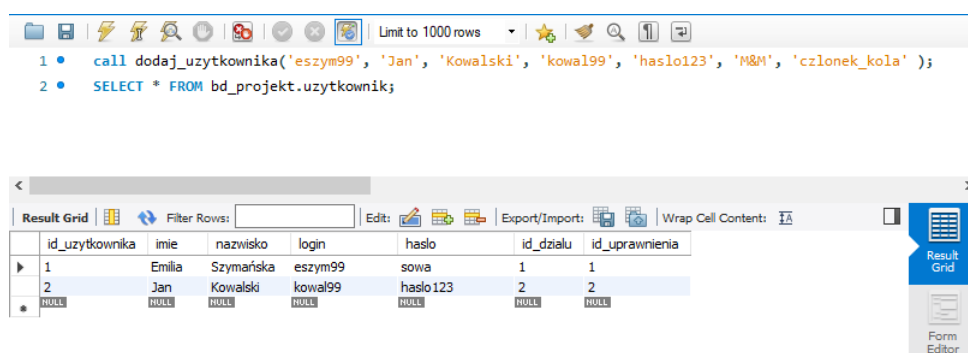


The screenshot shows a database client interface with a query editor at the top containing the SQL statement: `SELECT * FROM bd_projekt.uzytkownik;`. Below the editor is a toolbar with icons for various functions. The main area displays a 'Result Grid' with a table of data. The table has columns: `id_uzytkownika`, `imie`, `nazwisko`, `login`, `haslo`, `id_dzialu`, and `id_uprawnienia`. There is one row of data.

id_uzytkownika	imie	nazwisko	login	haslo	id_dzialu	id_uprawnienia
1	Emilia	Szymańska	eszym99	sowa	1	1

Rysunek 17: Lista użytkowników przed dodaniem

Jest jeden użytkownik o uprawnieniach administratora (ID uprawnienia jako 1). Próbowemy następnie dodać drugiego użytkownika, podając jako login osoby wywołującej login administratora.

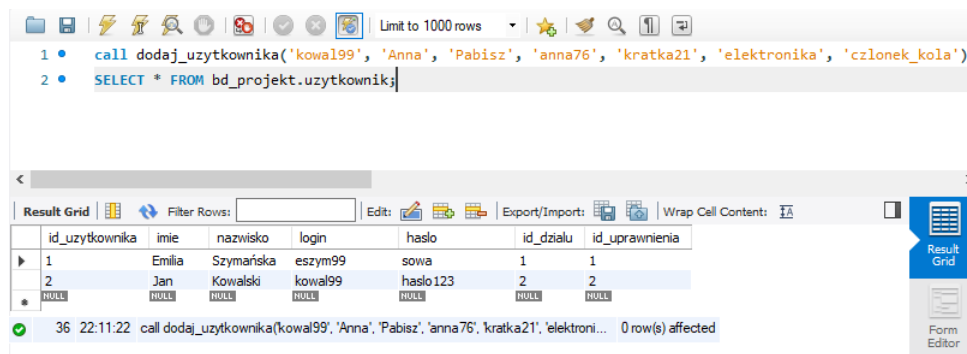


The screenshot shows the same database client interface. The query editor now contains two SQL statements: `call dodaj_uzytkownika('eszym99', 'Jan', 'Kowalski', 'kowa199', 'haslo123', 'M&M', 'czlonek_kola');` followed by `SELECT * FROM bd_projekt.uzytkownik;`. The 'Result Grid' now displays two rows of data.

id_uzytkownika	imie	nazwisko	login	haslo	id_dzialu	id_uprawnienia
1	Emilia	Szymańska	eszym99	sowa	1	1
2	Jan	Kowalski	kowa199	haslo123	2	2

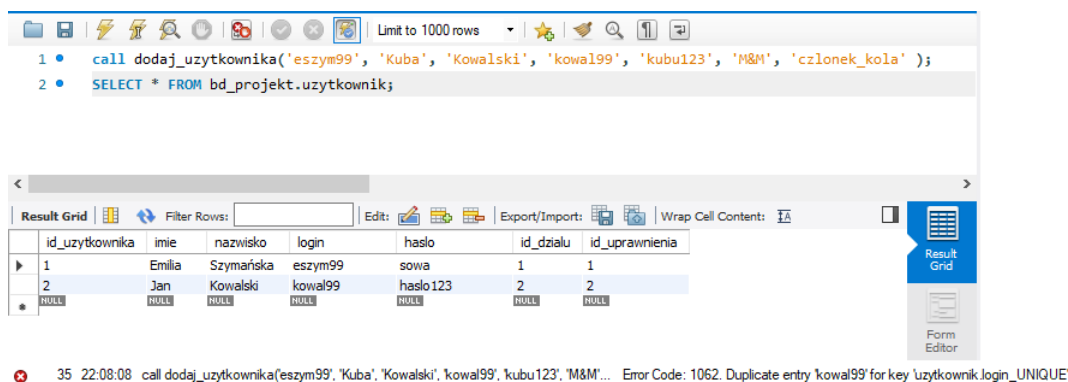
Rysunek 18: Lista użytkowników po dodaniu użytkownika

Udaje nam się dodać użytkownika. Następnie próbujemy wywołać tę procedurę, podając członka koła (ID uprawnienia jako 2) jako osobę wywołującą. Oto wynik próby wykonania tej operacji:



Rysunek 19: Próba dodania użytkownika przez zwykłego członka koła

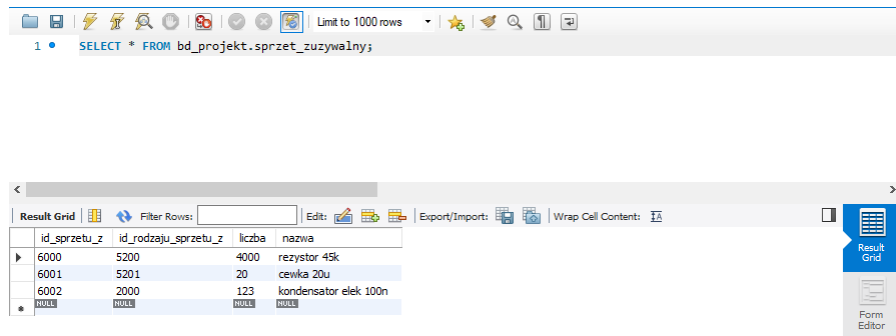
Jak widać pojawia się informacja, że procedura została wywołana, ale nic się nie zmieniło („0 row(s) affected”). Następnie próbujemy dodać użytkownika, który ma login taki sam jak użytkownik już istniejący.



Rysunek 20: Próba dodania użytkownika z nieunikatowym loginem

Baza danych nie pozwala nam na dodanie użytkownika, gdyż wpisany login nie jest unikatowy. W żadnym z przypadków nie musimy się martwić numeracją ID, gdyż - jak można zauważyć - ID numerują się automatycznie.

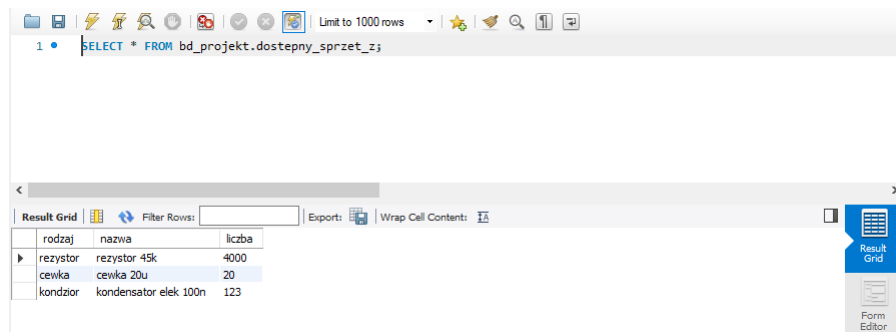
Przechodzimy do testu przykładowego widoku dostępnych sprzętów zużywalnych. Na początku pokazujemy jak wygląda tabela „sprzet_zuzywalny” oraz widok „dostepny_sprzet_z”.



1 • `SELECT * FROM bd_projekt.sprzet_zuzywalny;`

id_sprzetu_z	id_rodzaju_sprzetu_z	liczba	nazwa
6000	5200	4000	rezystor 45k
6001	5201	20	cewka 20u
6002	2000	123	kondensator elek 100n
NULL	NULL	NULL	NULL

Rysunek 21: Widok „sprzet_zuzywalny”

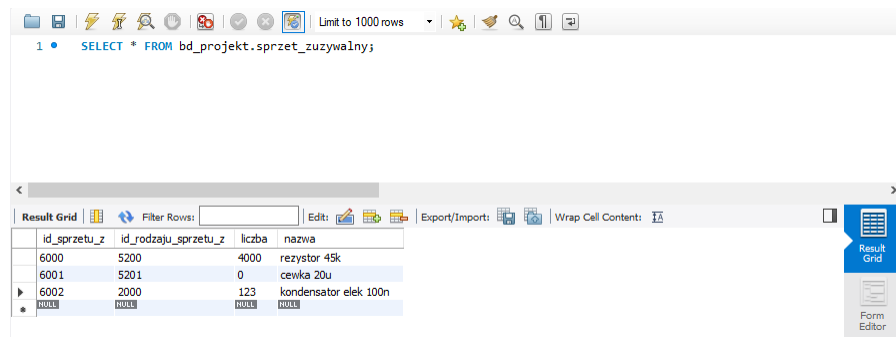


1 • `SELECT * FROM bd_projekt.dostepny_sprzet_z;`

rodzaj	nazwa	liczba
rezystor	rezystor 45k	4000
cewka	cewka 20u	20
kondzior	kondensator elek 100n	123

Rysunek 22: Widok „dostepny_sprzet_z”

Obecnie nie widać wielkiej różnicy, jednak jeśli ustawimy liczbę sprzętu na 0 widok od tabeli będzie się różnił w następujący sposób:



1 • `SELECT * FROM bd_projekt.sprzet_zuzywalny;`

id_sprzetu_z	id_rodzaju_sprzetu_z	liczba	nazwa
6000	5200	4000	rezystor 45k
6001	5201	0	cewka 20u
6002	2000	123	kondensator elek 100n
NULL	NULL	NULL	NULL

Rysunek 23: Widok „sprzet_zuzywalny” po zmianie ilości sprzętu

1 • `SELECT * FROM bd_projekt.dostepny_sprzet_z;`

rodzaj	nazwa	liczba
rezystor	rezystor 45k	4000
kondziór	kondensator elek 100n	123

Rysunek 24: Widok „dostepny_sprzet_z” po zmianie ilości sprzętu

Widok ten bowiem nie pokazuje niedostępnego sprzętu.
Po dodaniu losowych danych przetestowaliśmy działanie widoków. Oto wyniki ich testów:

1 • `SELECT * FROM bd_projekt.wszyscy_uzytkownicy_loginy;`

nazwisko	imie	login
Kowalski	Jan	kowal89
Malik	Mateusz	malik66
Mariski	Marek	mmanski
Osak	Maciek	mosak
Szymańska	Aga	agaszyn
Szymańska	Emilia	eszym99
Zieliński	Igor	iziel99

Rysunek 25: Widok „wszyscy_uzytkownicy_loginy”

1 • `SELECT * FROM bd_projekt.administratorzy;`

nazwisko	imie
Mariski	Marek
Osak	Maciek
Szymańska	Emilia
Zieliński	Igor

Rysunek 26: Widok „administratorzy”

1 • `SELECT * FROM bd_projekt.dostepny_sprzet_nz;`

rodzaj	nazwa	maksymalny_czas
lutownica	LT_100	6
lutownica	LT_200	6
lutownica	lutownica2000	6
omomierz	multimetr Z344	32
oscylloskopy	oscylloskop2000	30
lutownica	super lutownica	6

Rysunek 27: Widok „dostepny_sprzet_nz”

1 • `SELECT * FROM bd_projekt.zamowienia_nz_niezwroczone;`

nazwisko	imie	numer_zamowienia	rodzaj	nazwa	data_wypozyczenia	termin_zwrotu
Kowalski	Jan	1118	lutownica	LT_600	2020-05-29	2020-06-03

Rysunek 28: Widok „zamowienia_nz_niezwroczone”

1 • SELECT * FROM bd_projekt.uzytownicy_i_dzial;

nazwa_dzialu	nazwisko	imie
elektronika	Mański	Marek
elektronika	Osak	Maciek
elektronika	Zielński	Igor
konstrukcja	Malik	Mateusz
MSM	Kowalski	Jan
MSM	Szymańska	Aga
software	Szymańska	Emilia

Rysunek 29: Widok „uzytownicy_i_dzial”

Poniżej mamy zaprezentowane przykładowe wywołanie procedur oraz zmiany przez nie powodowane.
Procedura dodania sprzętu zużywanego:

Call stored procedure bd_projekt.dodaj_sprzet_z

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

login_wywołującego: eszym99 [IN] varchar(15)

nowa_nazwa: cewka Q=5 [IN] VARCHAR(30)

ile: 1000 [IN] SMALLINT

rodzaj_sprzetu: cewka [IN] VARCHAR(30)

Execute Cancel

Rysunek 30: Wywołanie procedury „dodaj_sprzet_z”

id_sprzetu_z	id_rodzaju_sprzetu_z	liczba	nazwa
6000	5200	4000	rezystor 1M
6001	5201	0	cewka 20u
6002	2000	0	kondensator elek 100n
6009	5200	500	rezystor 100k
6011	5206	13	Tranzystory szybkie
6012	5206	56	Tranzystor szybki
6013	5206	89	Szybki tranzystorek
6021	5201	1000	cewka Q=5

Rysunek 31: Porównanie tabeli „sprzet_zuzywalny” przed i po wywołaniu procedury „dodaj_sprzet_z”

Procedura złożenia zamówienia i dodania do niego elementu:

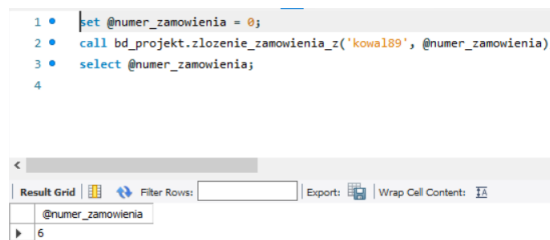
Call stored procedure bd_projekt.zlozenie_zamowienia_z

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

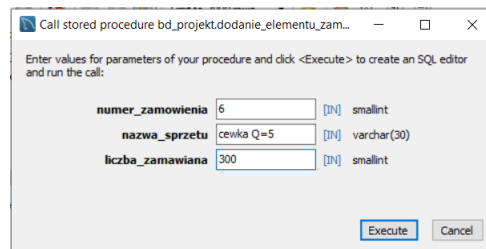
login_zamawiającego: kowal89 [IN] varchar(15)

Execute Cancel

Rysunek 32: Wywołanie procedury „zlozenie_zamowienia_z”



Rysunek 33: Wynik wywołania procedury „zlozenie_zamowienia_z”

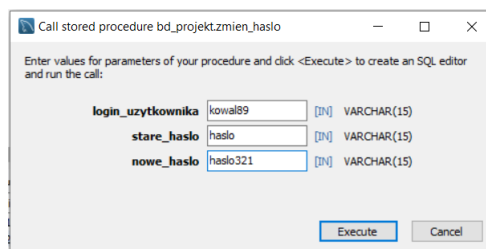


Rysunek 34: Wywołanie procedury „dodanie_elementu_zamowienia_z”

	nazwisko	imie	numer_zamowienia	rodzaj	nazwa	liczba	data_zamowienia
▶	Szymańska	Emilia	4	rezystor	rezystor 1M	400	2020-05-03
	Szymańska	Emilia	4	cewka	cewka 20u	10	2020-05-03
	Szymańska	Emilia	4	cewka	cewka 20u	20	2020-05-03
	Szymańska	Emilia	5	rezystor	rezystor 1M	10000	2020-05-05
	Szymańska	Emilia	5	rezystor	rezystor 1M	1000	2020-05-05
	nazwisko	imie	numer_zamowienia	rodzaj	nazwa	liczba	data_zamowienia
▶	Szymańska	Emilia	4	rezystor	rezystor 1M	400	2020-05-03
	Szymańska	Emilia	4	cewka	cewka 20u	10	2020-05-03
	Szymańska	Emilia	4	cewka	cewka 20u	20	2020-05-03
	Szymańska	Emilia	5	rezystor	rezystor 1M	10000	2020-05-05
	Szymańska	Emilia	5	rezystor	rezystor 1M	1000	2020-05-05
	Kowalski	Jan	6	cewka	cewka Q=5	300	2020-05-31

Rysunek 35: Porównanie widoków „zamowienia_z” przed i po złożeniu zamówienia i dodaniu do niego elementu

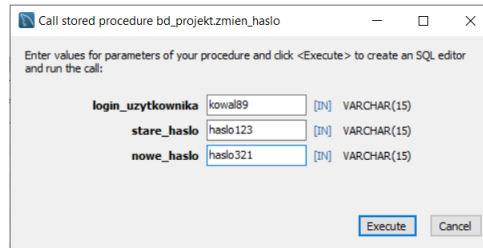
Procedura zmiany hasła:



Rysunek 36: Wywołanie procedury „zmien_haslo” dla podanego błędnie starego hasła

13 16:50:50 call bd_projekt.zmien_haslo('kowa189', 'haslo', 'haslo321') 0 row(s) affected

Rysunek 37: Wynik wywołania procedury „zmien_haslo” dla podanego błędnie starego hasła



Rysunek 38: Wywołanie procedury „zmien_haslo” dla podanego poprawnie starego hasła

	nazwisko	imie	numer_zamowienia	rodzaj	nazwa	liczba	data_zamowienia
▶	Szymańska	Emilia	4	rezystor	rezystor 1M	400	2020-05-03
	Szymańska	Emilia	4	cewka	cewka 20u	10	2020-05-03
	Szymańska	Emilia	4	cewka	cewka 20u	20	2020-05-03
	Szymańska	Emilia	5	rezystor	rezystor 1M	10000	2020-05-05
	Szymańska	Emilia	5	rezystor	rezystor 1M	1000	2020-05-05
	nazwisko	imie	numer_zamowienia	rodzaj	nazwa	liczba	data_zamowienia
▶	Szymańska	Emilia	4	rezystor	rezystor 1M	400	2020-05-03
	Szymańska	Emilia	4	cewka	cewka 20u	10	2020-05-03
	Szymańska	Emilia	4	cewka	cewka 20u	20	2020-05-03
	Szymańska	Emilia	5	rezystor	rezystor 1M	10000	2020-05-05
	Szymańska	Emilia	5	rezystor	rezystor 1M	1000	2020-05-05
	Kowalski	Jan	6	cewka	cewka Q=5	300	2020-05-31

Rysunek 39: Porównanie tabel „uzytkownik” przed i po wywołaniu procedury zmiany hasła

Na podobnej zasadzie testowaliśmy pozostałe procedury i widoki i z dużym prawdopodobieństwem możemy stwierdzić, że zaimplementowana przez nas baza danych działa poprawnie.

5 Implementacja i testy aplikacji

5.1 Instalacja i konfigurowanie systemu

Na podstawie napisanego w języku Python kodu wygenerowaliśmy plik wykonywalny (.exe). Do utworzenia takiego pliku wykorzystaliśmy narzędzie PyInstaller. W tym celu należy w wierszu poleceń wpisać komendę „pip install pyinstaller”, by zainstalować narzędzie. W naszym przypadku framework Kivy wymagał Pythona w wersji 3.7, zatem przed wywołaniem tego polecenia musieliśmy dodać „py -3.7 -m”. Po zainstalowaniu należy najpierw stworzyć środowisko, w którym będziemy budować plik .exe.

```
D:\Studia\4_semestr\BD\DataBases-Application\BD\app> py -3.7 -m PyInstaller --name EquipManager ../BD.py
```

Rysunek 40: Tworzenie środowiska dla pliku .exe

Następnie otwieramy plik specyfikacji w tym folderze za pomocą wybranego edytora tekstu.

```
D:\Studia\4_semestr\BD\DataBases-Application\BD\app> notepad .\EquipManager.spec
```

Rysunek 41: Otworzenie pliku specyfikacji

W tym pliku nanosimy zmiany zgodnie z dokumentacją: <https://kivy.org/doc/stable/guide/packaging-windows.html>.

```
# -*- mode: python ; coding: utf-8 -*-
from kivy_deps import sdl2, glew

block_cipher = None

a = Analysis(['..\BD.py'],
             pathex=['D:\\Studia\\4_semestr\\BD\\DataBases-Application\\BD\\app'],
             binaries=[],
             datas=[],
             hiddenimports=[],
             hookspath=[],
             runtime_hooks=[],
             excludes=[],
             win_no_prefer_redirects=False,
             win_private_assemblies=False,
             cipher=block_cipher,
             noarchive=False)
pyz = PYZ(a.pure, a.zipped_data,
          cipher=block_cipher)
exe = EXE(pyz,
          a.scripts,
          [],
          exclude_binaries=True,
          name='EquipManager',
          debug=False,
          bootloader_ignore_signals=False,
          strip=False,
          upx=True,
          console=True )
coll = COLLECT(exe, Tree('..'),
               a.binaries,
               a.zipfiles,
               a.datas,
               *[Tree(p) for p in (sdl2.dep_bins + glew.dep_bins)],
               strip=False,
               upx=True,
               upx_exclude=[],
               name='EquipManager')
```

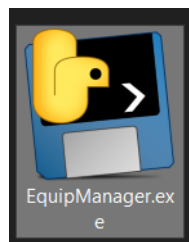
Rysunek 42: Modyfikacja pliku specyfikacji

Na koniec wywołujemy komendę tworzącą plik .exe.

```
D:\Studia\4_semestr\BD\DataBases-Application\BD\app> py -3.7 -m PyInstaller .\EquipManager.spec
```

Rysunek 43: Buildowanie aplikacji

Po takim zabiegu otrzymujemy docelowy plik.



Rysunek 44: Ikona pliku .exe

By móc skorzystać z systemu, należy taki plik dwukrotnie kliknąć w eksploratorze plików lub z wiersza poleceń wykonać komendę poniższą komendę:

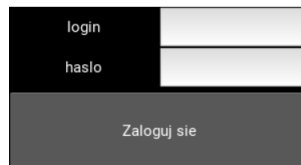
```
D:\Studia\4_semestr\BD\DataBases-Application\BD\app\dist\EquipManager> .\EquipManager.exe
```

Rysunek 45: Odpalenie pliku .exe

Na koniec usuwamy z wynikowego folderu z plikiem .exe skrypty Pythonowe i plik konfiguracyjny .kv. Taki folder można przesłać użytkownikowi, który następnie utworzy sobie skrót .exe w dowolnie wybranym przez siebie miejscu w komputerze.

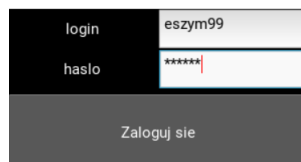
5.2 Instrukcja użytkowania aplikacji

Po uruchomieniu aplikacji pojawia się okno logowania:



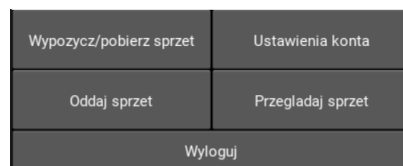
Rysunek 46: Ekran logowania

Musimy wpisać swoje dane logowania. Litery hasła są wyświetlane jako znak „*”.



Rysunek 47: Ekran logowania z danymi

Rozpatrzmy najpierw opcje zwykłego członka koła. Po zalogowaniu okno wygląda w następujący sposób:



Rysunek 48: Ekran po zalogowaniu się członka koła

Opcja wylogowania powoduje przełączenie się do ekranu logowania (rys.40). Opcja ustawień konta pozwala użytkownikowi na wgląd w swoje dane (poza hasłem, które w każdym przypadku jest przedstawione jako ciąg ośmiu znaków „*”) oraz zmianę hasła.

Login:	kowal89	Zmien login
Haslo:	*****	Zmien haslo
Imie:	Jan	Zmien imie
Nazwisko:	Kowalski	Zmien nazwisko
Dzial:	M&M	Zmien dzial
Uprawnienia:	czlonek_kola	Zmien uprawnienia
Powrot		

Rysunek 49: Ustawienia konta czlonka kola

By zmienić hasło, należy wybrać odpowiednią opcję, wpisać stare hasło i dwukrotnie nowe. Tu też nie widać liter hasła.

Stare haslo:	
Nowe haslo:	
Powtorz:	
Powrot	Zatwierdz

Rysunek 50: Zmiana hasla

Przeglądanie sprzętu umożliwia nam zobaczenie dostępnego sprzętu oraz nałożenie filtrów na wyszukiwanie.

Wybierz typ	Wybierz rodzaj
	Szukaj
cewka Q=5 (na stanie: 600)	
rezystor 100k (na stanie: 500)	
rezystor 1M (na stanie: 4000)	
Szybki tranzystorek (na stanie: 89)	
Tranzystor szybki (na stanie: 56)	
Tranzystory szybkie (na stanie: 13)	
LT_100 (maks. wyp. : 6)	
LT_200 (maks. wyp. : 6)	
Powrot	

Rysunek 51: Przeglądanie sprzętu

Poniżej jest przedstawione przykładowe nałożenie filtrów.

Sprzęty zużywalne	rezystor
10	Szukaj
rezystor 100k (na stanie: 500)	
Powrot	

Rysunek 52: Przeglądanie sprzętu z nałożonymi filtrami

Po wejściu w opcję z pobraniem/wypożyczeniem sprzętu pojawia się okno wyboru typu zamówienia.

Złoz zamówienie zużywalne
Złoz zamówienie nieużywalne
Powrot

Rysunek 53: Wybór typu zamówienia

Po wybraniu opcji z zamówieniem sprzętu zużywalnego pojawia się następujące okno:

Wybierz sprzęt z poniższej listy	Wybierz rodzaj	Wybrane sprzęty	
	Szukaj		
cewka Q=5 (na stanie: 600)			
rezystor 100k (na stanie: 500)			
rezystor 1M (na stanie: 4000)			
Szybki tranzystorek (na stanie: 89)			
Tranzystor szybki (na stanie: 56)			
Tranzystory szybkie (na stanie: 13)			
Liczba:		Usun z wybranych	Złoz zamówienie
Dodaj element do zamówienia		Powrot	

Rysunek 54: Pobieranie sprzętu zużywalnego

Wybierz sprzęt z poniższej listy	Wybierz rodzaj	Wybrane sprzęty	
	Szukaj	cewka Q=5 ilość: 200	
cewka Q=5 (na stanie: 600)		rezystor 1M ilość: 300	
rezystor 100k (na stanie: 500)			
rezystor 1M (na stanie: 4000)			
Szybki tranzystorek (na stanie: 89)			
Tranzystor szybki (na stanie: 56)			
Tranzystory szybkie (na stanie: 13)			
Liczba:		Usun z wybranych	Zloz zamowienie
Dodaj element do zamowienia		Powrot	

Analogicznie jest w sprzęcie nieużywalnym - jedyna różnica jest taka, że w dolnym oknie podaje się ilość dni, na którą chcemy wypożyczyć sprzęt.

W opcji oddawania sprzętu jest lista tych elementów, które obecnie są przez nas wypożyczone.

Rysunek 56: Oddawanie wypożyczonego sprzętu

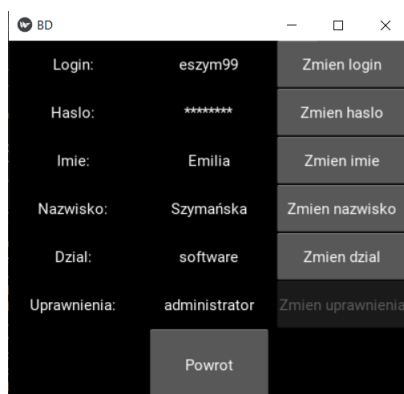
Po wybraniu sprzętu, który chcemy oddać, należy kliknąć przycisk „Oddaj sprzęt”. Pojawi się okno

potwierdzające dokonanie zwrotu:

Oddajesz sprzęt:	LT_600
Zatwierdź	Anuluj

Rysunek 57: Potwierdzenie chęci oddania wypożyczonego sprzętu

Do finalizacji akcji należy wybrać opcję „Zatwierdź”.
W przypadku opcji dostępnych u administratora - oprócz omówionych powyżej - są jeszcze „Zarządzanie użytkownikami” i „Zarządzanie sprzętem”. W przypadku ustawień konta administrator ma więcej opcji możliwych zmian.



Login:	eszym99	Zmien login
Hasło:	*****	Zmien hasło
Imię:	Emilia	Zmien imię
Nazwisko:	Szymańska	Zmien nazwisko
Dział:	software	Zmien dział
Uprawnienia:	administrator	Zmien uprawnienia
		Powrot

Rysunek 58: Ustawienia konta administratora

Administrator nie ma możliwości sam sobie zmienić uprawnień - bezsensownym by było, by sam siebie miał zdegradować. Okna zmian wyglądają analogicznie jak w przypadku zmiany hasła (oprócz tego, że nie trzeba podawać tym razem starych danych) - jedynym wyjątkiem jest zmiana działu, który jest wybierany za pomocą rozwijanej listy.

Okno zarządzania sprzętem wygląda po wybraniu w następujący sposób:

Wybierz typ	Wybierz rodzaj	Dodaj sprzęt zużywalny
	Szukaj	Dodaj sprzęt nieużywalny
LT_100 LT_200 LT_600 lutownica2000 multimetr2344 oscylloskop2000 rezystor 100k rezystor 1M		Dodaj rodzaj sprzętu zużywalnego
		Dodaj rodzaj sprzętu nieużywalnego
		Zmien maksymalne wypożyczenie
		Usun rodzaj sprzętu zużywalnego
		Usun rodzaj sprzętu nieużywalnego
		Zaawansowane
		Usun sprzęt
		Modyfikuj sprzęt
		Powrot

Rysunek 59: Zarządzanie sprzętem

Po lewej stronie mamy wyszukiwanie sprzętu analogiczne jak w przeglądaniu sprzętu, z tą jednak różnicą, że tu są także niedostępne urządzenia i elementy. Dodawanie sprzętów lub ich rodzajów wygląda analogicznie, więc zaprezentujemy jedynie jedną z tych opcji.

Nazwa	
Ilość	
Rodzaj	Wybierz rodzaj
Powrot	Zatwierdź

Rysunek 60: Dodawanie sprzętu zużywalnego

Przy dodawaniu rodzaju sprzętu zużywalnego podaje się tylko jego nazwę, nieużywalnego - nazwę oraz maksymalne możliwe wypożyczenie (w dniach). Można zmienić maksymalne wypożyczenie, okno wygląda w następujący sposób:

Rodzaj	omomierz
Obecny maksymalny czas wypożyczenia	32
Nowy maksymalny czas wypożyczenia	
Powrot	Zatwierdź

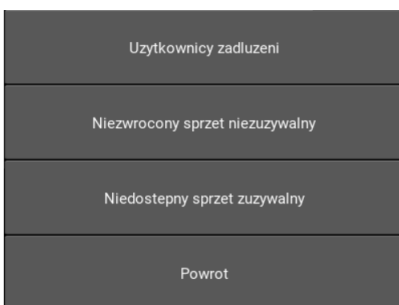
Rysunek 61: Zmiana maksymalnego możliwego wypożyczenia danego rodzaju

Rodzaj sprzętu wybierany jest z rozwijanej listy, wartość obecnego maksymalnego czasu wypożyczenia zmienia się w zależności od wybranego rodzaju. Wybranie opcji usunięcia rodzaju przeniesie nas do listy, na której należy zaznaczyć rodzaj i przejść do usunięcia. Pojawi się okno wymagające potwierdzenia decyzji.



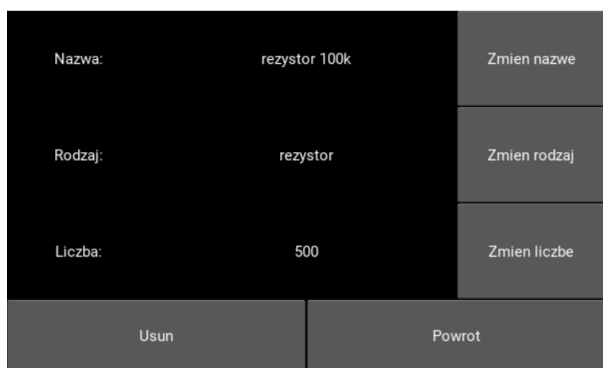
Rysunek 62: Usuwanie rodzaju sprzętu nieużywanego

Zaawansowane opcje pozwalają nam na zobaczenie listy zadłużonych użytkowników, niezwróconych sprzętów nieużywanych lub niedostępnych sprzętów zużywalnych. Z dwóch ostatnich widoków można, po zaznaczeniu sprzętu, przejść bezpośrednio do ich edycji (jak w opcji „Modyfikuj sprzęt”).



Rysunek 63: Zaawansowane opcje sprzętu

Po wybraniu sprzętu można przejść do okna usuwania (pojawi się konieczność potwierdzenia decyzji) lub do modyfikacji właściwości sprzętu. Poniżej są przedstawione okna modyfikacji dwóch typów sprzętów. Okna zmiany nazwy, rodzaju, ilości lub statusu wyglądają analogicznie jak okna zmiany parametrów konta użytkownika.



Rysunek 64: Modyfikacja sprzętu zużywanego

Nazwa:	oscylloskop2000	Zmien nazwe
Rodzaj:	oscylloskopy	Zmien rodzaj
Status:	Wypożyczony przez Szymańska Emilia do 2020-06-03	Zmien status
Usun		Powrot

Rysunek 65: Modyfikacja sprzętu nieużywanego

W zarządzaniu użytkownikami pojawia się okno z listą użytkowników.

Mański Marek (elektronika)	
Osak Maciek (elektronika)	
Szymańska Emilia (elektronika)	
Szymańska Aga (M&M)	
Zieliński Igor (elektronika)	
Usun uzytkownika	Modyfikuj dane
Dodaj uzytkownika	Powrot

Rysunek 66: Zarządzanie użytkownikami

Opcja usuwania działa tak samo jak przy usuwaniu sprzętu - należy zaznaczyć użytkownika i potwierdzić usunięcie. Przy wybraniu dodania pojawi się poniższe okno:

Imie	
Nazwisko	
Login	
Haslo	
Powtorz haslo	
Dzial	wybierz dzial
Uprawnienia	wybierz uprawnienia
Zatwierdz	Powrot

Rysunek 67: Dodawanie użytkownika

W przypadku przejścia do modyfikacji danych użytkownika, mamy podobne okno jak przy zmienianiu parametrów konta - tu jednak mamy dostęp do zmiany wszystkich parametrów.

Login:	mmanski	Zmien login
Haslo:	*****	Zmien haslo
Imie:	Marek	Zmien imie
Nazwisko:	Mański	Zmien nazwisko
Dzial:	elektronika	Zmien dzial
Uprawnienia:	administrator	Zmien uprawnienia
Powrot		

Rysunek 68: Zmiana parametrów konta użytkownika

Ostatnią opcją jest przeglądanie zamówień. Po kliknięciu na przycisk pojawia się okno:

Przeglądaj zamówienia zużywalne
Przeglądaj zamówienia niezubywalne
Powrot

Rysunek 69: Wybór typu przeglądanych zamówień

W obu przypadkach są listy zamówień, których szczegóły można podpatrzyć w innym oknie.

Szymańska Emilia - zam. nr 4 z dnia 2020-05-03	
Szymańska Emilia - zam. nr 5 z dnia 2020-05-05	
Kowalski Jan - zam. nr 6 z dnia 2020-05-31	
Szymańska Emilia - zam. nr 7 z dnia 2020-05-31	
Powrot	Szczegoly

Rysunek 70: Lista zamówień

W przypadku szczegółów wygląda to następująco dla odpowiednich typów:

Zamawiający/a: Szymańska Emilia	LT_100 (lutownica) - zwrócono 2020-05-03
Data złożenia: 2020-05-04	oscyloskop2000 (oscylloskopy) - zwrócono 2020-05-04
Numer zamówienia: 1114	
Powrot	

Rysunek 71: Szczegóły zamówienia sprzętów nieużywalnych

Zamawiający/a: Szymańska Emilia	rezystor 1M (rezystor) - 400 szt.
Data złożenia: 2020-05-03	cewka 20u (cewka) - 10 szt.
Numer zamówienia: 4	cewka 20u (cewka) - 20 szt.
Powrot	

Rysunek 72: Szczegóły zamówienia sprzętów zużywalnych

Po prawych stronach są listy elementów zamówień, po lewych dane dot. użytkownika wypożyczającego, data złożenia zamówienia i jego numer.

Pojawia się parę systemowych niedoskonałości - w niektórych przypadkach (gdy to samo okno jest użyte w więcej niż jednym miejscu) przycisk „Powrót” powoduje cofnięcie nas do innego okna niż przez nas zakładanego (np. z A i C możemy przejść do B, ale z B zawsze nas cofnie do A). Lista wybieralna w tym framework’u ma też mankament, który sprawia, że domyślnie (przy pierwszym wejściu do okna lub cofnięciu się) lista przyjmuje wskazanie na inny element, niekoniecznie ten zaznaczony. By pozbyć się wątpliwości, należy w takim przypadku ponownie kliknąć na wybrany przez nas element. W tym celu, by przypadkiem nie usunąć nie tego elementu z listy, dodane jest okno potwierdzenia decyzji ze wskazanym elementem.

5.3 Testowanie opracowanych funkcji systemu

Pominiemy testowanie przechodzenia między oknami w tym punkcie, gdyż dowód na to widać w instrukcji powyżej - bez możliwości przechodzenia nie moglibyśmy pokazać działania aplikacji. Przy pustych oknach przy logowaniu lub niepoprawnych danych i próbie przejścia dalej okno pozostaje to samo - system nie dopuszcza do przejścia dalej. Podobnie jest przy próbie dodawania sprzętów, rodzajów, użytkowników, modyfikacji danych użytkowników, sprzętów itp. Dzieje się tak, gdyż przed wywołaniem odpowiednich funkcji łączących się z bazą danych sprawdzana jest zawartość pól i weryfikowana ich poprawność.

Poniżej znajdują się przykładowe testy funkcjonalności aplikacji.

Modyfikacja parametrów konta użytkownika

Login:	iziel99	Zmien login
Haslo:	*****	Zmien haslo
Imie:	Igor	Zmien imie
Nazwisko:	Zieliński	Zmien nazwisko
Dzial:	elektronika	Zmien dzial
Uprawnienia:	administrator	Zmien uprawnienia
Powrot		

Rysunek 73: Parametry konta użytkownika przed zmianami

Login:	igorziel99	Zmien login
Haslo:	*****	Zmien haslo
Imie:	Igor	Zmien imie
Nazwisko:	Zieliński	Zmien nazwisko
Dzial:	konstrukcja	Zmien dzial
Uprawnienia:	czlonek_kola	Zmien uprawnienia
Powrot		

Rysunek 74: Parametry konta użytkownika po zmianach

Usuwanie użytkownika

Ilnicka Aga (konstrukcja)	
Kowalski Jan (M&M)	
Malik Mateusz (konstrukcja)	
Mański Marek (elektronika)	
Osak Maciek (elektronika)	
Usun uzytkownika	Modyfikuj dane
Dodaj uzytkownika	Powrot

Rysunek 75: Lista użytkowników przed usunięciem

Usuwasz uzytkownika:	Ilnicka Aga (konstrukcja)
Zatwierdz	Anuluj

Rysunek 76: Potwierdzenie usunięcia użytkownika

Kowalski Jan (M&M)	
Malik Mateusz (konstrukcja)	
Mański Marek (elektronika)	
Osak Maciek (elektronika)	
Szymańska Emilia (elektronika)	
Usun uzytkownika	Modyfikuj dane
Dodaj uzytkownika	Powrot

Rysunek 77: Lista użytkowników po usunięciu

Składanie zamówienia sprzętu nieużywalnego i jego zwrot

Wybierz sprzęt z poniższej listy	Wybierz rodzaj	Wybrane sprzęty	
	Szukaj	LT_100 ilosc dni: 3	
LT_100 (maks. wyp. : 6)		multimetr2344 ilosc dni: 20	
LT_200 (maks. wyp. : 6)			
lutownica2000 (maks. wyp. : 6)			
multimetr2344 (maks. wyp. : 32)			
super lutownica (maks. wyp. : 6)			
Na ile dni:		Usun z wybranych	Zloz zamowienie
Dodaj element do zamowienia		Powrot	

Rysunek 78: Złożenie zamówienia sprzętów nieużywalnych

[illegible]

Oddajesz sprzęt:		multimetr2344	
Zatwierdz		Anuluj	

LT_600 (lutownica) do 2020-06-03

LT_100 (lutownica) do 2020-06-03

Oddaj sprzęt

Powrót

Zamawiający/a: Kowalski Jan	LT_100 (lutownica) do 2020-06-03
Data złożenia: 2020-06-03	multimetr2344 (omomierz) - zwrócono 2020-05-31
Numer zamówienia: 1120	
Powrot	

Rysunek 82: Szczegóły zamówienia po oddaniu jednego sprzętu

Składanie zamówienia sprzętu zużywalnego

Wybierz sprzęt z poniższej listy	cewka	Wybrane sprzęty	
	Szukaj	rezystor 100k ilość: 100	
cewka Q=5 (na stanie: 600)		cewka Q=5 ilość: 200	
Liczba:		Usun z wybranych	Złoz zamówienie
Dodaj element do zamówienia		Powrot	

Rysunek 83: Złożenie zamówienia sprzętów zużywalnych

Zamawiający/a: Kowalski Jan	rezystor 100k (rezystor) - 100 szt.
Data złożenia: 2020-05-31	cewka Q=5 (cewka) - 200 szt.
Numer zamówienia: 8	
Powrot	

Rysunek 84: Szczegóły zamówienia zużywalnego użytkownika

5.4 Omówienie wybranych rozwiązań programistycznych

5.4.1 Implementacja interfejsu dostępu do bazy danych

Uzyskanie takiego interfejsu jest wynikiem użycia framework'a Kivy w języku Python. Cała szata graficzna jest ujęta w plikach z rozszerzeniem .kv - tam, używając elementów takich jak Label, TextInput, Spinner, Button i paru innych, udało się uzyskać taki efekt. Do rozlokowania elementów na ekranie posłużyły GridLayout, FloatLayout i BoxLayout (w zależności od potrzeb). Dodatkowo tworzyliśmy specjalne obiekty (listy z typu Recycle View), by móc wyświetlać np. listę użytkowników i sprzętów. Poniżej znajdują się przykłady elementów interfejsu.



Rysunek 85: Elementy typu Label oraz TextInput



Rysunek 86: Element typu Spinner



Rysunek 87: Element typu RecycleViewList



Rysunek 88: Elementy typu Button

Zaimplementowanie wyglądu prostego okna w tym frameworku może wyglądać w następujący sposób:

```

<ChangeLoginScreen>
  name: "zmiana loginu"

  newlog: newlog
  newlog2: newlog2
  bckbtn: bckbtn
  accbtn: accbtn

  id: parent

  GridLayout:
    rows: 3
    cols: 2
    Label:
      text: "Nowy login: "
    TextInput:
      id: newlog
      text: ""
      multiline: False
    Label:
      text: "Powtorz: "
    TextInput:
      id: newlog2
      text: ""
      multiline: False
    Button:
      id: bckbtn
      text: "Powrot"
      on_press: parent.GetBack()
    Button:
      id: accbtn
      text: "Zatwierdz"
      on_press: parent.SubmitNewLogin()
</ChangeLoginScreen>

```

Rysunek 89: Ekran zmiany loginu

Do każdego elementu, do którego chcieliśmy się odwoływać w innych plikach, należało dodać id (w powyższym przykładzie newlog, newlog2, bckbtn, accbtn). Wszystkie ekrany muszą mieć swoje nazwy (pole „name”), by móc w łatwy sposób się między nimi przełączać. Każdemu takiemu ekranowi w .kv odpowiada klasa o identycznej nazwie w plikach .py oraz ma metody wymienione np. w zdarzeniu „on_press”. Parent jest tu specjalnym id do całego ekranu. W pliku .kv należy dodatkowo umieścić pomniejsze ekrany w menadżerze ekranów (poniżej znajduje się wycinek kodu menadżera).

```

<MyScreenManager>
  LoginScreen:
  UserScreen:
  AdminScreen:
  AccountSettingsScreen:
  ChangeLoginScreen:
  ChangePwdScreen:
  ChangeNameScreen:
  ChangeSnameScreen:
  ChangeDepScreen:
  ChangeRigScreen:
  ChooseUserScreen:
  AddUserScreen:
  DeleteUserScreen:
  ManageEquipScreen:
  AddUsableEqpScreen:
  AddUnUsableEqpScreen:
  AddUsableEqpKindScreen:
  AddUnUsableEqpKindScreen:
  AdvancedEqpScreen:
  DeleteUsKindScreen:
  ConfirmDeleteUsKindScreen:
  DeleteUnUsKindScreen:
  ConfirmDeleteUnUsKindScreen:
  DebtUsersScreen:
  UnavailUsEqpScreen:
  UnavailUnUsEqpScreen:
  MaxBorrowScreen:
  ConfirmDeleteEqpScreen:
  ModUsEqpScreen:
  ModUnUsEqpScreen:
  ModUsEqpKindScreen:
  ModUnUsEqpKindScreen:
  ModNameUsableEqpScreen:
  ModNameUnUsableEqpScreen:

```

Rysunek 90: Menadżer ekranów

Wielkość okien i ich przełączanie jest implementowane w plikach .py w odpowiednich klasach. Przykładowo:

```
def GetBack(self):
    app = App.get_running_app()
    Window.size = (400, 300)
    app.root.current = "zaawansowane zarzadzanie sprzetem"
```

Rysunek 91: Ustawianie aktualnego okna i jego rozmiarów

Kod informuje nas, że wymiary okna będą równe 400x300px i zmieniamy obecny ekran na ekran o nazwie „zaawansowane zarządzanie sprzętem”. Na podstawie tych kilku zasad, zapoznaniu się z elementami framework’a wraz z ich dodatkowymi parametrami można stworzyć (w razie wątpliwości czytając dokumentację) praktycznie każdy ekran.

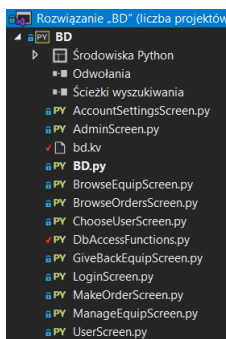
Poniżej znajduje się przykładowe porównanie fragmentu kodu z pliku .py oraz z pliku .kv tego samego okna - ekranu logowania. Widzimy, że w obu przypadkach mamy te same nazwy id obiektów (tu typu TextInput). Do przycisku (Button - on_press) podpięte jest wywołanie metody LoginTry() obiektu parent (to dotyczy id samego ekranu). W implementacji tej metody w klasie LoginScreen można zauważyć przełączanie między ekranami za pomocą komendy app.root.get_screen() i podaniu wartości w Window.size(). Tak postępujemy w przypadku każdej zmiany okna.

<pre>class LoginScreen(Screen): log = ObjectProperty(TextInput) pwd = ObjectProperty(TextInput) def LoginTry(self): app = App.get_running_app() rights = Login(self.log.text, self.pwd.text) if rights != None: app.root.login = self.log.text app.root.rig = rights screen = app.root.get_screen("ustawienia konta") screen.UpdateData(app.root.login) if rights == 'czlonek kola': Window.size = (400, 160) app.root.current = 'opcje czlonka kola' elif rights == 'administrator': Window.size = (400, 360) app.root.current = 'opcje administratora' def ClearInput(self): self.log.text = "" self.pwd.text = ""</pre>	<pre><LoginScreen> name: "ekran logowania" log: log pwd: pwd id: parent BoxLayout: orientation: "vertical" spacing: 10 GridLayout: cols: 2 rows: 2 row_force_default: False row_default_height: 40 Label: text: "login" TextInput: id: log multiline: False text: "" Label: text: "haslo" TextInput: id: pwd multiline: False password: True text: "" Button: text: "Zaloguj sie" on_press: parent.LoginTry() </LoginScreen></pre>
--	--

Rysunek 92: Porównanie kodu dot. ekranu logowania w pliku .py i .kv

5.4.2 Implementacja wybranych funkcjonalności systemu

Cały projekt został podzielony na trzynaście plików, w tym jeden .kv i jeden uruchamiający aplikację. Pozostałe pliki odpowiadają większym zbiorom klas dotyczących danej funkcjonalności.



Rysunek 93: Struktura projektu

Do komunikacji z bazą danych służą funkcje zaimplementowane w pliku DbAccessFunctions.py. Przykładowa komunikacja z bazą danych wygląda w następujący sposób:

```
def UserBasicData(login):
    query = """SELECT * FROM wszyscy_uzytkownicy_dane WHERE login = %s """
    cnx = mysql.connector.connect(user='sudo', password='xbxbpun', database='bd_projekt')
    cur = cnx.cursor(buffered=True)

    cur.execute(query, (login,))
    usr = cur.fetchone()
    user = f'{{usr[0]}} {{usr[1]}} {{usr[3]}}'

    cur.close()
    cnx.close()
    return user
```

Rysunek 94: Komunikacja z bazą danych

Oto kolejne kroki, jakie wykonujemy w celu komunikacji z bazą:

- definiujemy, jakie zapytanie chcemy wystosować do bazy (korzystając z gotowych widoków lub z tabel) - podanie jako wartości w """...""" do query,
- definiujemy parametry połączenia (użytkownika, hasło, wskazujemy bazę danych) - polecenie mysql.connector.connect(),
- tworzymy tzw. kursor związany z połączeniem z bazą - metoda cursor(),
- wykonujemy zapytanie (razem z parametrami lub bez nich) poleceniem execute,
- przechytujemy dane poleceniem fetchone lub fetchall,
- odpowiednio dostosowujemy dane do swoich potrzeb i zamykamy połączenie z bazą.

W przypadku procedury wygląda to analogicznie, używamy tylko innego polecenia (callproc).

```
def ChangeName(who, whom, new):
    cnx = mysql.connector.connect(user='sudo', password='xbxbpun', database='bd_projekt')
    cur = cnx.cursor(buffered=True)

    args = [who, whom, new]
    cur.callproc('zmien_imie', args)

    cnx.commit()
    cur.close()
    cnx.close()
```

Rysunek 95: Wywołanie procedury z poziomu aplikacji

Wszystkie funkcjonalności realizowane są analogicznie - tworzony jest widok w pliku .kv, pod dane zdarzenia (np. kliknięcie przycisku) podpinane są odpowiednie metody, powstaje klasa związana z widokiem (połączona za pomocą id) i w metodach są przełączane ekrany, wywoływane funkcje połączeniowe z bazą (wcześniej zaimplementowane) i realizowane dodatkowe opcje.

```
class ModUsEqKindScreen(Screen):
    kind = ListProperty([])
    eqpname = StringProperty("")
    bckbtn = ObjectProperty(Button)
    confbtn = ObjectProperty(Button)

    def __init__(self, **kwargs):
        super(ModUsEqKindScreen, self).__init__(**kwargs)
        self.UpdateData(self.eqpname)

    def SubmitChange(self):
        app = App.get_running_app()
        if self.kindsel.text != "Wybierz rodzaj":
            ModUsEqKind(app.root.login, self.eqpname, self.kindsel.text)
            screen = app.root.get_screen("modyfikuj sprzęt zużywalny")
            screen.UpdateData(self.eqpname)
            Window.size = (600, 360)
            app.root.current = "modyfikuj sprzęt zużywalny"
            self.ClearInput()

    def UpdateData(self, equip):
        self.kind = UsableEquipmentKind()
        self.eqpname = equip

    def GetBack(self):
        app = App.get_running_app()
        screen = app.root.get_screen("modyfikuj sprzęt zużywalny")
        Window.size = (600, 360)
        app.root.current = "modyfikuj sprzęt zużywalny"
        self.ClearInput()

    def ClearInput(self):
        self.ids.kindsel.text = "Wybierz rodzaj"
        self.eqpname = ""
```

Rysunek 96: Przykładowa klasa (modyfikacja rodzaju sprzętu zużywalnego) związana z ekranem

```
class UserScreen(Screen):
    lendtakebtn = ObjectProperty(Button)
    settingbtn = ObjectProperty(Button)
    givebackbtn = ObjectProperty(Button)
    searchbtn = ObjectProperty(Button)

    def GetToSettings(self):...

    def LogOut(self):...

    def GetToBrowseEq(self):
        app = App.get_running_app()
        screen = app.root.get_screen("przeglądaj sprzęt")
        screen.UpdateData()
        Window.size = (500, 600)
        app.root.current = "przeglądaj sprzęt"

    def GetToReturnEq(self):
        app = App.get_running_app()
        screen = app.root.get_screen("oddaj sprzęt")
        screen.UpdateData()
        Window.size = (500, 600)
        app.root.current = "oddaj sprzęt"

    def GetToMakeOrder(self):
        app = App.get_running_app()
        Window.size = (400, 150)
        app.root.current = "wybor typu zamówienia"
```

Rysunek 97: Przykładowa klasa (ekran użytkownika) związana z ekranem

```

class AdminScreen(Screen):
    lendtakebtn = ObjectProperty(Button)
    settingbtn = ObjectProperty(Button)
    givebackbtn = ObjectProperty(Button)
    searchbtn = ObjectProperty(Button)
    usermgbtn = ObjectProperty(Button)
    gearmgbtn = ObjectProperty(Button)

    def GetToSettings(self):...
    def GetToMgUsers(self):...
    def GetToMgEqp(self):...
    def LogOut(self):...
    def GetToBrowseEqp(self):...
    def GetToBrowseOrders(self):...
    def GetToReturnEqp(self):...
    def GetToMakeOrder(self):...

```

Rysunek 98: Przykładowa klasa (ekran administratora) związana z ekranem

5.4.3 Implementacja mechanizmów bezpieczeństwa

Zabezpieczeniem już jest to, że niezalogowany użytkownik nie ma dostępu do funkcjonalności systemu (widzi jedynie ekran logowania), a po zalogowaniu wyświetlają się różne okna w zależności od tego, czy użytkownik ma uprawnienia administratora czy też nie.

Wypożycz/pobierz sprzęt	Ustawienia konta
Oddaj sprzęt	Przeglądaj sprzęt
Wyloguj	

Rysunek 99: Widok członka koła

Wypożycz/pobierz sprzęt	Ustawienia konta
Oddaj sprzęt	Przeglądaj sprzęt
Zarządzaj sprzętem	Zarządzaj użytkownikami
Przeglądaj zamówienia	Wyloguj

Rysunek 100: Widok administratora

Członek koła zatem nie ma nawet fizycznej możliwości wybrania niedozwolonej dla niego opcji np. zarządzania sprzętem, użytkownikami lub przeglądania zamówień.

Podczas logowania nie są wyświetlane litery hasła, są one zamieniane na znak „*”. W kodzie jest to zrealizowane poprzez dodanie w pliku .kv w elemencie TextInput pola „password: True”.

Rysunek 101: Ukrywanie znaków z hasła

```
TextInput:
  id: pwd
  multiline: False
  password: True
  text: ""
```

Rysunek 102: Ukrywanie znaków z hasła - kod

Ponadto, by zabezpieczyć się przed wykradaniem haseł z bazy danych, dodaliśmy funkcję haszującą z dodatkiem soli (ciągu zaburzającego). Przy dodawaniu użytkownika w aplikacji, hasło jest przetwarzane przez funkcję i wysyłane do bazy już zahasowane, przy logowaniu nakładamy funkcję na wpisany ciąg znaków i porównujemy z zawartością bazy.

```
hashed = blake2b(key=b'secret', digest_size=10)
password = password + "bfbfjqbr"
hashed.update(password.encode())
```

Rysunek 103: Haszowanie hasła

id_uzytkownika	imie	nazwisko	login	haslo	id_dzialu	id_uprawnienia
1	Emilia	Szymańska	eszym99	11a67e1b2e9af7bab7e3	1	1
2	Jan	Kowalski	kowal89	7e5089a94793ab1dfd9e	2	2

Rysunek 104: Haszowanie hasła - wynik w bazie danych

By dodatkowo się upewniać, że dana osoba ma uprawnienia do wykonania danej operacji, przy wywołaniu części kluczowych procedur podajemy jako argument uprawnienia użytkownika, które to potem jest argumentem procedury wywoływanej przez bazę danych.

```
def ChangeDep(who, whom, new):
    cnx = mysql.connector.connect(user='sudo', password='xbxbpun', database='bd_projekt')
    cur = cnx.cursor(buffered=True)

    args = [who, whom, new]
    cur.callproc('zmien_dzial', args)

    cnx.commit()
    cur.close()
    cnx.close()
```

Rysunek 105: Funkcja zmiany działu połączona z procedurą

```
def SubmitNewDep(self):
    app = App.get_running_app()
    if self.ids.depsel.text != "Wybierz dzial":
        ChangeDep(app.root.login, self.login, self.depsel.text)
        screen = app.root.get_screen("ustawienia konta")
        screen.UpdateData(self.login)
        Window.size = (400, 360)
        app.root.current = "ustawienia konta"
        self.ClearInput()
```

Rysunek 106: Zmiana działu w kodzie - wywołanie funkcji

6 Podsumowanie i wnioski

- Ogólny cel został w naszym poczuciu osiągnięty - powstała zaplanowana, znormalizowana baza danych wraz z procedurami i widokami oraz umożliwiającą korzystanie z zasobów bazy aplikacją dostępową.
- Projekt bazy danych i aplikacji dostępowej za pomocą modelu conceptualnego, fizycznego i logicznego oraz makiet aplikacji pozwala na wczesne zobrazowanie wyglądu produktu oraz szybkie znalezienie i usunięcie ewentualnych błędów logicznych, bądź uwzględnienie nowej funkcjonalności na wczesnym etapie prac. Gdyby natomiast przystąpić od razu do implementacji, mogłoby się okazać, że zdiagnozowany błąd spowoduje, że cały projekt należy zacząć od nowa.
- W naszym projekcie, w implementacji bazy danych, zamiast definiować sztucznie użytkowników, powinniśmy raczej skorzystać z wbudowanych w myśl kont użytkowników i polecenia GRANT do nadawania im odpowiednich uprawnień. Uprościłoby to kod wielu procedur i zostawiłoby mniej miejsc na błędy.
- Kivy nie jest idealnym frameworkiem i w kolejnych projektach wybralibyśmy jakiś inny, gdyż posiada pewne uciążliwe błędy oraz dokumentacja frameworka Kivy nie należy do bardzo szczegółowych - trzeba szukać rozwiązań błędów bezpośrednio w implementacji.

Wersja kodu aplikacji jest dostępna na repozytorium pod linkiem: <https://github.com/Igor0663/DataBases-Application>.

Spis rysunków

1	Model conceptualny	5
2	Model logiczny	6
3	Model fizyczny	7
4	Tabele z uprawnieniami	9
5	Diagram Przypadków Użycia	9
6	Projekt interfejsu graficznego	10
7	Tworzenie tabel „dział” oraz „uprawnienia”	11
8	Tworzenie tabeli „uzytkownik”	11
9	Tabele w bazie danych	11
10	„Reverse engineer”	12
11	Tworzenie widoku „dostepny_sprzet_z”	12
12	Tworzenie widoku „zamowienia_nz_wszystkie”	13
13	Tworzenie procedury „zmien_login”	13
14	Tworzenie widoku „zwrot_sprzetu_nz”	13
15	Przykładowe zabezpieczenie w bazie danych	14
16	Procedury i widoki	14
17	Lista użytkowników przed dodaniem	15
18	Lista użytkowników po dodaniu użytkownika	15
19	Próba dodania użytkownika przez zwykłego członka koła	16
20	Próba dodania użytkownika z nieunikatowym loginem	16
21	Widok „sprzet_zuzywalny”	17
22	Widok „dostepny_sprzet_z”	17
23	Widok „sprzet_zuzywalny” po zmianie ilości sprzętu	17
24	Widok „dostepny_sprzet_z” po zmianie ilości sprzętu	18
25	Widok „wszyscy_uzytkownicy_loginy”	18
26	Widok „administratorzy”	18
27	Widok „dostepny_sprzet_nz”	18
28	Widok „zamowienia_nz_niezwroczone”	18
29	Widok „uzytkownicy_i_dzial”	19
30	Wywołanie procedury „dodaj_sprzet_z”	19
31	Porównanie tabeli „sprzet_zuzywalny” przed i po wywołaniu procedury „dodaj_sprzet_z” . . .	19

32	Wywołanie procedury „zlozenie_zamowienia_z”	19
33	Wynik wywołania procedury „zlozenie_zamowienia_z”	20
34	Wywołanie procedury „dodanie_elementu_zamowienia_z”	20
35	Porównanie widoków „zamowienia_z” przed i po złożeniu zamówienia i dodaniu do niego elementu	20
36	Wywołanie procedury „zmien_haslo” dla podanego błędnie starego hasła	20
37	Wynik wywołania procedury „zmien_haslo” dla podanego błędnie starego hasła	20
38	Wywołanie procedury „zmien_haslo” dla podanego poprawnie starego hasła	21
39	Porównanie tabel „uzytkownik” przed i po wywołaniu procedury zmiany hasła	21
40	Tworzenie środowiska dla pliku .exe	21
41	Otworzenie pliku specyfikacji	21
42	Modyfikacja pliku specyfikacji	22
43	Buildowanie aplikacji	22
44	Ikona pliku .exe	22
45	Odpalenie pliku .exe	23
46	Ekran logowania	23
47	Ekran logowania z danymi	23
48	Ekran po zalogowaniu się członka koła	23
49	Ustawienia konta członka koła	24
50	Zmiana hasła	24
51	Przeglądanie sprzętu	24
52	Przeglądanie sprzętu z nałożonymi filtrami	25
53	Wybór typu zamówienia	25
54	Pobieranie sprzętu zużywalnego	25
55	Pobieranie sprzętu zużywalnego - przykładowe dane	26
56	Oddawanie wypożyczonego sprzętu	26
57	Potwierdzenie chęci oddania wypożyczonego sprzętu	27
58	Ustawienia konta administratora	27
59	Zarządzanie sprzętem	28
60	Dodawanie sprzętu zużywalnego	28
61	Zmiana maksymalnego możliwego wypożyczenia danego rodzaju	28
62	Usuwanie rodzaju sprzętu niezaużywalnego	29
63	Zaawansowane opcje sprzętu	29
64	Modyfikacja sprzętu zużywalnego	29
65	Modyfikacja sprzętu niezaużywalnego	30
66	Zarządzanie użytkownikami	30
67	Dodawanie użytkownika	30
68	Zmiana parametrów konta użytkownika	31
69	Wybór typu przeglądanych zamówień	31
70	Lista zamówień	31
71	Szczegóły zamówienia sprzętów niezaużywalnych	32
72	Szczegóły zamówienia sprzętów zużywalnych	32
73	Parametry konta użytkownika przed zmianami	33
74	Parametry konta użytkownika po zmianach	33
75	Lista użytkowników przed usunięciem	33
76	Potwierdzenie usunięcia użytkownika	33
77	Lista użytkowników po usunięciu	34
78	Złożenie zamówienia sprzętów niezaużywalnych	34
79	Lista wypożyczonych sprzętów użytkownika po złożeniu zamówienia	35
80	Potwierdzanie oddania sprzętu	35
81	Lista wypożyczonych sprzętów użytkownika po oddaniu sprzętu	35
82	Szczegóły zamówienia po oddaniu jednego sprzętu	36
83	Złożenie zamówienia sprzętów zużywalnych	36
84	Szczegóły zamówienia zużywalnego użytkownika	36

85	Elementy typu Label oraz TextInput	37
86	Element typu Spinner	37
87	Element typu RecyclerViewList	37
88	Elementy typu Button	37
89	Ekran zmiany loginu	38
90	Menadżer ekranów	38
91	Ustawianie aktualnego okna i jego rozmiarów	39
92	Porównanie kodu dot. ekranu logowania w pliku .py i .kv	39
93	Struktura projektu	40
94	Komunikacja z bazą danych	40
95	Wywołanie procedury z poziomu aplikacji	40
96	Przykładowa klasa (modyfikacja rodzaju sprzętu zużywanego) związana z ekranem	41
97	Przykładowa klasa (ekran użytkownika) związana z ekranem	41
98	Przykładowa klasa (ekran administratora) związana z ekranem	42
99	Widok członka koła	42
100	Widok administratora	42
101	Ukrywanie znaków z hasła	43
102	Ukrywanie znaków z hasła - kod	43
103	Haszowanie hasła	43
104	Haszowanie hasła - wynik w bazie danych	43
105	Funkcja zmiany działu połączona z procedurą	43
106	Zmiana działu w kodzie - wywołanie funkcji	43