Projekt bazy danych aplikacji zarządzającej gospodarstwem domowym

Damian Koper - 241292
Wiktor Pieklik - 241282

Household App

Opis systemu

Gospodarstwo domowe jest miejscem, który zrzesza mieszkających w nim domowników, w którym każdy z nich ma wkład w jego rozwój. W celu śledzenia i zrównoważenia kosztów wynikających z użytkowania przez domowników zasobów dostępnych w gospodarstwie potrzebny jest rozbudowany system śledzący i obliczający wydatki w przełożeniu na konkretną osobę i, jeśli domownicy godzą się na równy bądź też odgórnie zdefiniowany podział kosztów, wyliczanie konkretnych kwot. Działania te, razem ze śledzeniem mediów (woda, prąd, gaz) pozwolą na dokładniejsze przeanalizowanie wydatków, co w przyszłości może przełożyć się na optymalizację kosztów bycia domownikiem.

Gospodarstwo domowe rozpatrywane jako obiekt fizyczny z odpowiednim ułożeniem sensorów, może dostarczyć dane o swoim stanie, które można wykorzystać do sporządzenia różnorakich statystyk. Statystyki te mogą być podstawą do sterowania np. temperaturą w konkretnym pomieszczeniu.

Wymagania funkcjonalne aplikacji

Zarządzanie użytkownikami i gospodarstwem

Utworzenie konta użytkownika i logowanie	
Opis	Niezalogowany użytkownik jest w stanie stworzyć swoje konto i zalogować się.
Wejście	Dane użytkownika: login, hasło, imię, nazwisko, data urodzenia,
Wyjście	Potwierdzenie

Utworzenie gospodarstwa, dodawanie członków	
Opis	Zalogowany użytkownik jest w stanie utworzyć instancję gospodarstwa, do którego może dodać innych użytkowników.
Wejście	Dane gospodarstwa: nazwa, adres, identyfikatory użytkowników
Wyjście	Potwierdzenie
Czynności dodatkowe	Powiadomienie użytkownika o dodaniu go do gospodarstwa.

Zarządzanie użytkownikami	
Opis	Zalogowany użytkownik z uprawnieniami administratora jest w stanie zarządzać kontami użytkowników i ich uprawnieniami zgrupowanymi w role.
Wejście	Konta, uprawnienia, role członków
Wyjście	Potwierdzenie
Czynności dodatkowe	Powiadomienie użytkownika o zmianie jego uprawnień, danych konta

Powiadomienia	
Opis	Każdy użytkownik może generować i otrzymywać powiadomienia.
Wejście	Akcja użytkownika
Wyjście	---

Zakupy

CRUD Zakupy - sklepy, kategorie, zakupy	
Opis	Każdy użytkownik z odpowiednimi uprawnieniami może edytować dane dotyczące zrobionych zakupów.
Wejście	Wprowadzone dane
Wyjście	Potwierdzenie
Czynności dodatkowe	Podpowiedzi w trakcie wprowadzania na podstawie obecnych danych w bazie.

CRUD Lista zakupów - listy i podlisty	
Opis	Każdy użytkownik z odpowiednimi uprawnieniami może tworzyć, edytować i usuwać listy zakupów z podziałem na podlisty. Każda podlista może reprezentować np jeden sklep.
Wejście	Wprowadzone dane
Wyjście	Potwierdzenie
Czynności dodatkowe	Wysłanie powiadomienia o nowej liście zakupów

Lista zakupów - konwersja na zakupy	
Opis	Każdy użytkownik z odpowiednimi uprawnieniami może przekonwertować listę zakupów na zakupione produkty
Wejście	Obecna lista zakupów
Wyjście	Widok zakupionych produktów gotowych do zatwierdzenia
Czynności dodatkowe	Autouzupełnianie na podstawie obecnych danych w bazie.

Wydatki gospodarstwa

CRUD Stałe wydatki gospodarstwa	
Opis	Każdy użytkownik z odpowiednimi uprawnieniami może tworzyć, edytować i usuwać dane związane ze wydatkami gospodarstwa
Wejście	● Np wysokość miesięcznej opłaty za internet, telewizję. ● Początek okresu rozliczeniowego, okresu miesięcznego podsumowania. ● Ostrzeżenia o wydatkach
Wyjście	Potwierdzenie
Czynności dodatkowe	Generowanie powiadomienia o nowej liście zakupów

Zakupy - podsumowanie	
Opis	Każdy użytkownik z odpowiednimi uprawnieniami może wyświetlić informacje o wydatkach na zakupy swoich i gospodarstwa.
Wejście	Wyświetlany okres.
Wyjście	Widoki wykresów i statystyk
Czynności dodatkowe	---

Wydatki gospodarstwa - podsumowanie	
Opis	Każdy użytkownik z odpowiednimi uprawnieniami może wyświetlić informacje o wydatkach gospodarstwa.
Wejście	● Wyświetlany okres. ● Czy uwzględniać wydatki na zakupy czy nie.
Wyjście	Widoki wykresów i statystyk
Czynności dodatkowe	---

Dane fizyczne

Pobór mediów	
Opis	System zbiera dane o poborze mediów, a jeśli pobranie nie jest możliwe, umożliwia ręczne wprowadzenie
Wejście	Dane o poborze mediów
Wyjście	---
Czynności dodatkowe	Generowanie powiadomień

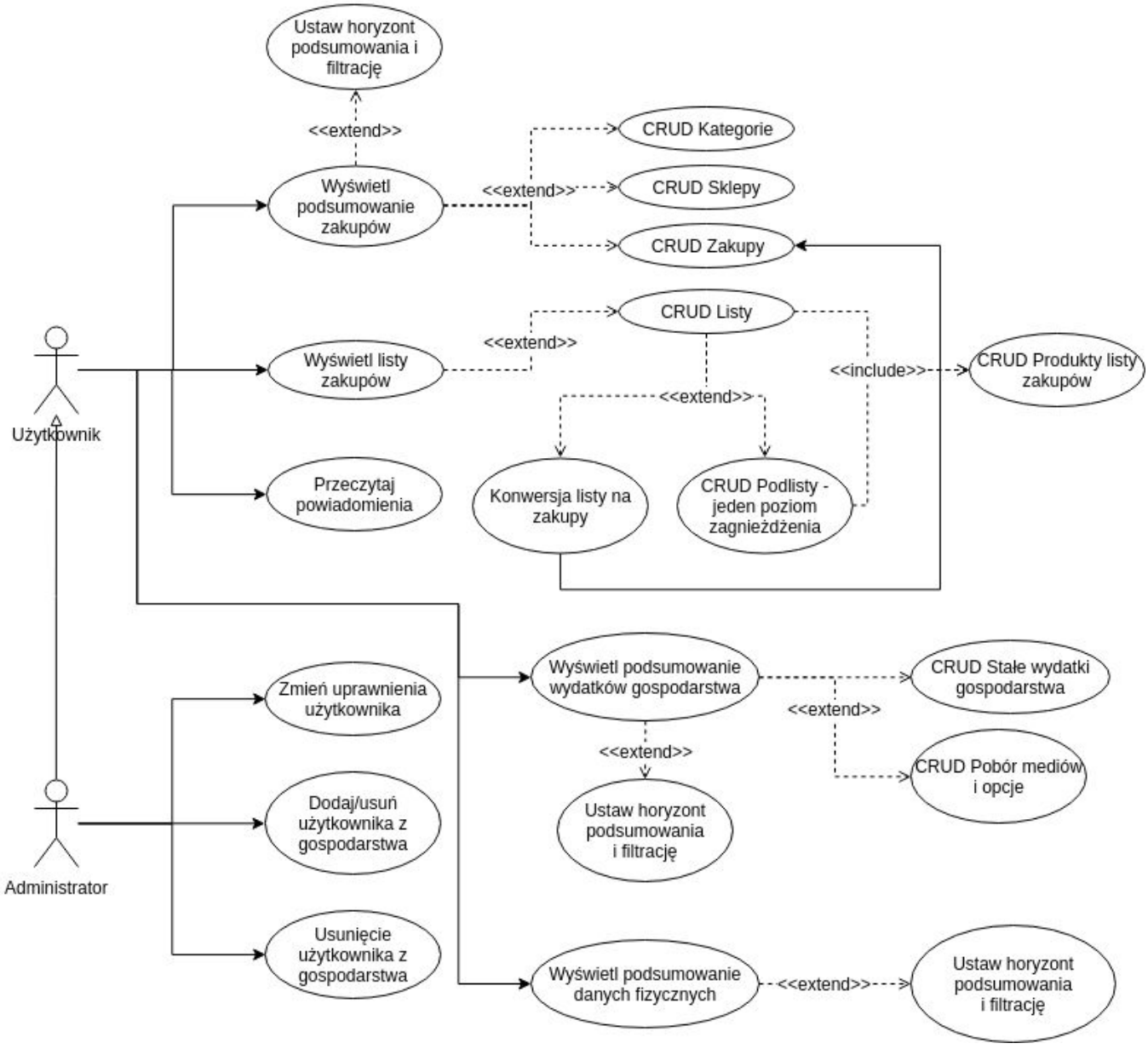
Temperatura, wilgotność, ruch, itp	
Opis	System zbiera informacje o temperaturze, ruchu itp w pomieszczeniach.
Wejście	Zebrane dane
Wyjście	---
Czynności dodatkowe	● Jeśli możliwe, sterowanie parametrami urządzeń domowych na podstawie zebranych danych. Np. sterowanie ogrzewaniem ● Generowanie powiadomień, alertów

Dane fizyczne - podsumowanie	
Opis	Każdy użytkownik z odpowiednimi uprawnieniami może wyświetlić zebrane dane fizyczne w formie wykresów i statystyk.
Wejście	Wyświetlany okres.
Wyjście	Widoki wykresów i statystyk
Czynności dodatkowe	---

Diagram przypadków użycia

Diagram przypadków użycia dla aplikacji systemu zarządzającego gospodarstwem domowym.

HouseholdApp



Projekt bazy danych

Entity Relationship Diagram

- CDM: https://raw.githubusercontent.com/damiankoper/dbProject1/master/docs/assets/ERD%20-%20CDM.png
- PDM: https://raw.githubusercontent.com/damiankoper/dbProject1/master/docs/assets/ERD%20-%20PDM.png

Analiza liczby instancji encji

Encja	Przewidywana liczba instancji
users	0+
users_permissions	(~5) * users
permissions	~15
roles	~5
permission_role	(~5) * roles
households	0+
utility_costs	~5
utility_usage_records	0+
env_records	0+
fixed_monthly_expenses	0+
shopping_items	0+
shopping_shops	0+
shopping_categories	(5~10)+
shopping_lists	(~5) * households
shopping_sublists	(~3) * households * lists
shopping_list_items	(~5) * households * lists * sublists
notifications	0+

Dla aplikacji lepszą z punktu widzenia organizacji danych i skalowalności okazałaby się architektura *multitenancy.*

Podstawowe transakcje

Wszystkie działania mogą generować powiadomienia, jednak nie są one kluczowe, więc nie są elementem transakcji. Wszystkie encje opatrzone zostaną w usuwanie kaskadowe, co czyni zbędnym ręczne usuwanie obiektów zależnych w jednej transakcji. Poniżej opisano integralne operacje wykorzystujące więcej niż jedną tabelę.

Nazwa	Tabele
• CRUD użytkownik	• users • users_permissions
• CRUD lista zakupów	• shopping_lists • shopping_sublists • shopping_list_items
• Konwersja listy zakupów na zakupy	• shopping_lists • shopping_sublists • shopping_list_items • shopping_items

Widoki

Widoki poprzez łączenie tabel i wykorzystywanie funkcji wbudowanych mogą wyświetlać informacje zrozumiałe dla użytkownika takie jak:

1. Zakupione produkty z danego okresu
2. Podsumowanie wydatków na zakupy z danego okresu dla każdego użytkownika
3. Podsumowanie wydatków na media z danego okresu
4. Podsumowanie wydatków gospodarstwa z danego okresu
5. Podsumowanie zebranych danych fizycznych dla danego okresu
6. Ostatnie pomiary temperatury/wilgotności/etc
7. Obecne zużycie mediów

Funkcje

Jako, że widoki nie mogą być parametryzowane, aby osiągnąć zamierzony cel użyte zostały procedury składowe. Zostały one użyte również do przeprowadzenia operacji opisanych w transakcjach.

Wyzwalacze

Na potrzeby projektu na wyzwalaczach opiera się system notyfikacji, który w normalnym przypadku działałby po stronie aplikacji.

Implementacja

Baza została zaimplementowana z wykorzystaniem DBMS Mysql Ver 14.14 Distrib 5.7.26, for Linux (x86_64) zainstalowanym na VPS z systemem Ubuntu 16.04. Przy tworzeniu struktury danych korzystano tylko i wyłącznie z interfejsu CLI. Do zapisu struktury i danych bazy do pliku `.sql` wykorzystano narzędzie `mysqldump` z flagą `--routines`. Bazę wypełniono danymi generowanymi losowo korzystając ze skryptu napisanym w języku JavaScript używającym bibliotek `mysql2` i `faker.js`.

Utworzone widoki:

- `shopping_items_view`,
- `env_records_temperature`
- `env_records_humidity`
- `env_records_movement`

Utworzone procedury składowe (przykładowe użycie):

```
call showShoppingItemsOfFromTo("dom_2", '2019-01-01', '2019-04-03');
call avgTemp_hh_room("dom_3", "room_2");
call avgHum_hh_room("dom_3", "room_2");
call shoppingExpences_hh_from_to('dom_3', '2019-03-15',
'2019-03-21');
call showPermissions_userId(50);
call insertNotification_hh(1,'info','{}'); - dodaje powiadomienie dla
     każdego użytkownika gospodarstwa (1)
```

Niektóre procedury składowe oparte są na widokach.

Utworzone wyzwalacze i zdarzenia wywołujące:

tooExpensive - after insert when price > 1000:

```
insert into shopping_items (shopping_shop_id, shopping_category_id,
user_id, household_id, name, price, discount_on_unit, quantity,
shared, date) values (1,1, 43, 2, "Coś drogiego", 1100, 0, 1 , false,
now());
```

tooHot - after insert when temperatura > 40

```
insert into env_records (household_id, type, value) values (1,
'temperature', '{"room_1": 18.267568418382893, "room_2":
19.08522642714848, "room_3": 41.4250575944651, "room_4":
18.718427486881644}');
```

tooHumid - after insert when humidity > 95

```
insert into env_records (household_id, type, value) values (1,
humidity, '{"room_1": 97.22885434447494, "room_2": 91.53714377798264,
"room_3": 52.6151348179647, "room_4": 95.76761420075096}');
```

newShoppingList - after insert

```
insert into shopping_lists values (2,1,'Lista 1');
```

Przykładowe zapytania:

https://github.com/damiankoper/dbProject1/blob/master/exampleQueries.md

Struktura bazy:

https://github.com/damiankoper/dbProject1/blob/master/dbStructure.md

Wnioski

Podczas prac nad projektem udało się osiągnąć wszystkie zakładane kamienie milowe. W fazie projektu opisano system, oraz przypadki użycia aplikacji w nim działającej. Zaprojektowano model logiczny i fizyczny - wykonano diagramy CDM i PDM oraz przeanalizowano przewidywaną liczbę instancji każdej encji.

W fazie implementacji utworzono zaprojektowane wcześniej struktury, wypełniono je danymi oraz utworzono przykładowe widoki, procedury składowe i wyzwalacze. Sam etap tworzenia widoków, procedur składowych i wyzwalaczy jest czysto demonstracyjny i w pewien sposób bezcelowy, ponieważ na tych funkcjonalnościach nie bazuje już żadna wyższa warstwa, którą mogłaby być faktyczna aplikacja. Przy jej projektowaniu wyklarowałyby się potrzeby związane z tymi funkcjonalnościami.

Zamiast skomplikowanych struktur wyrażeń sterujących trudnych do zaimplementowana w języku sql - foreach, if, while etc., podczas tworzenia aplikacji znacznie lepiej jest wykorzystać mapowanie obiektowo-relacyjne (ORM), które doskonale się sprawdza w przypadku aplikacji, gdzie dostęp i wymiana danych z bazą nie ma znaczącego wpływu na wydajność. Metoda ta generuje mniej optymalne zapytania, lecz bardziej zrozumiałe i łatwiej modyfikowalne przez programistę w porównaniu do zapytań całkowicie skrojonych na miarę.

Widoki, jako wirtualne tabele są przydatne w przypadku kwestii bezpieczeństwa, ograniczając dostęp użytkownika do określonych kolumn. Ułatwiają również wdrożenia szybkich zmian struktur tabeli, gdzie trzeba zachować zgodność z wdrożoną i starą strukturą.

Literatura

[1] Dokumentacja MySql 5.7 - https://dev.mysql.com/doc/refman/5.7/en/