I. Projekt koncepcji, założenia

1. Zdefiniowanie tematu projektu: krótki opis zadania, zdefiniowanie jego celów i zadań jakie ma realizować

Celem tego projektu miało być zbudowanie działającej bazy danych postgresql, oraz interfejsu do niej zbudowanej w modelu MVC. Moim zadaniem było zbudowanie bazy która działałaby jako encyklopedia gier komputerowych tj, przechowywała informację o danej grze, jej ocenę, kategorię, informację o wydawcy, oraz pozwalała na ocenienie gier. Końcowa aplikacja dostępna na stronie http://pascal.fis.agh.edu.pl/~3hujdus/projekt_bazy/index.php na serwerze wydziałowym Pascal jest przeznaczona dla końcowego użytkownika który ma możliwość wyświetlania, wyszukiwania i dodawania informacji w encyklopedii. Nie ma jednak możliwości usuwania, gdyż nie widzę sensu żeby postronna osoba miała taką możliwość. Jednakże istnieje panel administratora pozwalający na usuwanie z bazy danych pod adresem http://pascal.fis.agh.edu.pl/~3hujdus/projekt_bazy/index.php?sub=baza&action=admin . Baza danych uruchomiona w serwisie cloudowym ElephantSQL (https://www.elephantsql.com/) jest niestety w datacenter w Amazon Web Services US-East-1 (Northern Virginia) dlatego też czasem pojawiają się opóźnienia w odpowiedzi. Jednakże wewnątrz bazy zbudowana jest funkcjonalność pozwalająca na możliwie szybkie działanie. Może nie ma w niej wielu tabel, ale starałem się wykorzystać maksymalnie różne możliwości bazy danych by pokazać umiejętność korzystania z nich.

2. Analiza wymagań użytkownika: określenie funkcjonalności jakie ma spełniać projektowana baza danych.

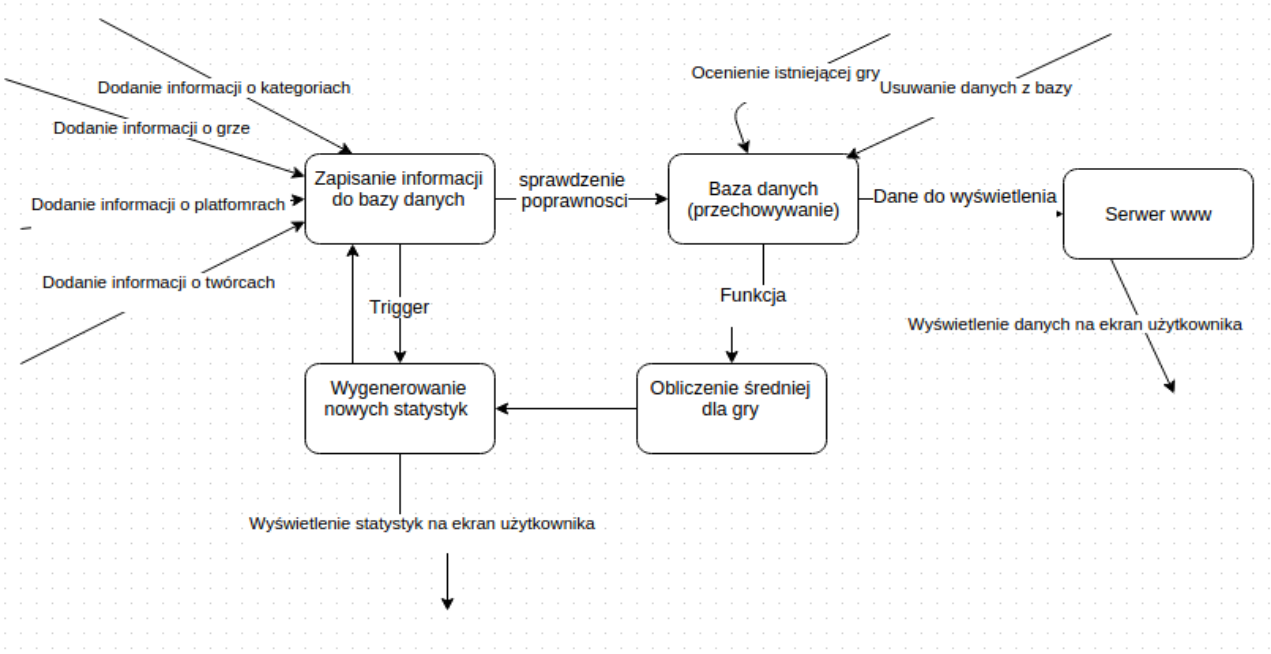
Projektowana baza danych ma za zadanie umożliwić przeglądanie, dodawanie i ocenianie gier, oraz w razie potrzeby usuwanie z niej. Baza również sama gromadzi statystyki o ilości gier, kategorii i platform. Zawarte są w niej triggery do zbierania statystyk, oraz widoki ułatwiające wyszukiwanie i wyświetlanie danych. Dodatkowo baza danych sprawdza poprawność wprowadzonych danych, pomimo że niektóre wartości są już sprawdzane za pomocą JavaScriptu.

3. Zaprojektowanie funkcji: określenie podstawowych funkcji realizowanych w bazie danych.

- **Dodawanie do bazy danych**
- **Usuwanie z bazy danych**
- **Gromadzenie statystyki**
- **Ocenianie konkretnych gier**

II. Projekt diagramów (konceptualny)

4. Budowa i analiza diagramu przepływu danych (DFD):



Podstawowy diagram DFD dla mojej bazy. Opisane są tylko główne funkcjonalności takie jak wprowadzanie, wyświetlanie danych, generowanie statystyk, i usuwanie oraz ocenianie zawartości.

5. Zdefiniowanie encji (obiektów) oraz ich atrybutów i zaprojektowanie relacji pomiędzy encjami

Obrazek z diagramem i definicjami encji nie zmieścił się w dokumentacji więc można go znaleźć pod adresem:

http://pascal.fis.agh.edu.pl/~3hujdus/projekt_bazy/baza_danych.PNG

Jest to diagram użyty przy początkowym projektowaniu bazy danych. W czasie realizacji projektu nastąpiło kilka zmian:
-Dodana tabela statystyka(ilosc_gier integer, ilosc_kategorii iteger,ilosc_platform integer, ilosc_ocen integer, srednia double precision) do przechowania statystyk.

Modyfikowane klucze obce w tabeli kategorie i platformy by umożliwić łatwe wyszukiwanie.

Czego nie widać na tym diagramie to:
-warunki:

- Gra.Ocena musi mieć wartość 0-10;
- Gra.Nazwa i Gra.Ocena nie może być pusta
- Gra.ilosc_ocen ma domyślną wartość 1
- Gra.nazwa musi być unique
- Kategoria.nazwa musi być unique
- Platforma.nazwa musi być unique

  Każda tabela poza „statystyka” posiada id, numerowane za pomocą sekwencji.

- Wykorzystane są relacje 1:n

III. Projekt logiczny

7. Projektowanie tabel, kluczy, indeksów

Cały dump bazy danych z wszystkimi funkcjami i tabelami dostępny jest pod adresem:

http://pascal.fis.agh.edu.pl/~3hujdus/projekt_bazy/bazadanych_dump.backup

W bazie sporządzonych jest kilka widoków pozwalających na łatwe wyszukiwanie, jeden widok który zawiera wszystkie możliwe informacje. Widoki zrealizowane są z użyciem JOIN,DISTINCT, ORDER BY, oraz funkcji agg_array(). Napisana jest funkcja pozwalająca dodać nową ocenę do danego rekordu z tablicy gra. Napisane jest też 6 triggerów zbierających statystyki z bazy danych. Każda tabela (z wyjątkiem tej z statystykami) posiada własną sekwencję do iterowania numerów ID. Relacje pomiędzy tabelami to 1:n. Baza posiada też 3 tabele łączące: Grę z Kategoriami, Grę z Platformami, Grę z Twórcami.

8. Słowniki danych: wypisać słownik danych, określić dziedziny oraz ich ograniczenia.

Projekt nie zawiera żadnych słowników danych ponieważ wszystkie tabele są modyfikowalne

9. Analiza zależności funkcyjnych i normalizacja tabel

Wszystkie tabele są maksymalnie znormalizowane wg postaci normalnych.

IV. Projekt funkcjonalny

10.Interfejsy do prezentacji, edycji i obsługi danych: zdefiniowanie struktury poszczególnych formularzy do wprowadzania danych oraz powiązań między formularzami.

Interfejs do obsługi bazy danych został zrealizowany przy użyciu php i JavaScriptu, każde wprowadzenie do bazy danych ma osobny widok z formularzem odpowiadającym odpowiedniej funkcji. Funkcje usuwania i wyszukiwania mają praktycznie ten sam formularz różniący się tylko podpiętymi pod niego funkcjami. Wyszukiwania wg kategorii, platformy i oceny jako rezultat zwracają tabelę gier i ich ocen. Samo „Wyszukaj grę” zwraca sformatowane wszystkie informacje jakie tylko dostępne są w bazie dodatkowo włącza się po wyszukaniu formularz do ocenienia właśnie wyszukanej gry. „Zawartość bazy” wyświetla wszystkie gry z bazy w formacie tabelki (tylko podstawowe informacje). Formularze dodawania do bazy mają zdefiniowany przycisk „Dodaj kolejny” który odświeża stronę.

11.Wizualizacja danych: określenie formy i struktury raportów które będą generowane przez bazę danych.

Większość danych jest wyświetlanych jako tabele, za wyjątkiem „Wyszukaj grę” gdzie jest przeformatowana odpowiedź. Raport z statystykami bazy danych wyświetlany jest na stronie głównej. Jest on generowany przy użyciu triggerów nałożonych na każdą tabelę przy dodawaniu lub usuwaniu do niej. Wiele zapytań zostało wpisanych w pliki php ponieważ były na tyle krótkie iż nie

widziałem sensu by tworzyć z nich funkcje.

12.Zdefiniowanie panelu sterowania aplikacji.

Całe sterowanie aplikacji odbywa się za pomocą panelu znajdującego się po lewej stronie. Wyjątkiem jest wejście w tryb administratora, domyślnie ukryty przed użytkownikiem.

V. Dokumentacja

13.Wprowadzanie danych: zdefiniowanie sposobu wprowadzania danych

Wprowadzanie danych jest zawsze ręczne, nie ma możliwości importu danych. Dane do wyszukiwania muszą być dokładnie takie same jak w bazie danych inaczej nie zostaną wyświetlone wyniki. Brak jakiejkolwiek odpowiedzi po naciśnięciu „Szukaj” oznacza brak rekordu w bazie.

14.Dokumentacja użytkownika

Aplikacja jest bardzo prosta w użytkowaniu. Wszystkie funkcje są praktycznie opisane więc nie sądzę żeby mogła sprawić problem w obsłudze. Poruszanie się po funkcjach odbywa się za pomocą menu z lewej strony.

15.Opracowanie dokumentacji technicznej

Dokumentacja do plików php znajduje się w na serwerze pascal pod linkiem.
Funkcje w JavaScript opisane są w pliku baza.js

Funkcje SQL zapisane są w pliku http://pascal.fis.agh.edu.pl/~3hujdus/projekt_bazy/funkcje.sql

Wszystkie pliki projektu znajdują się też pod adresem https://github.com/hades13542/projekt2

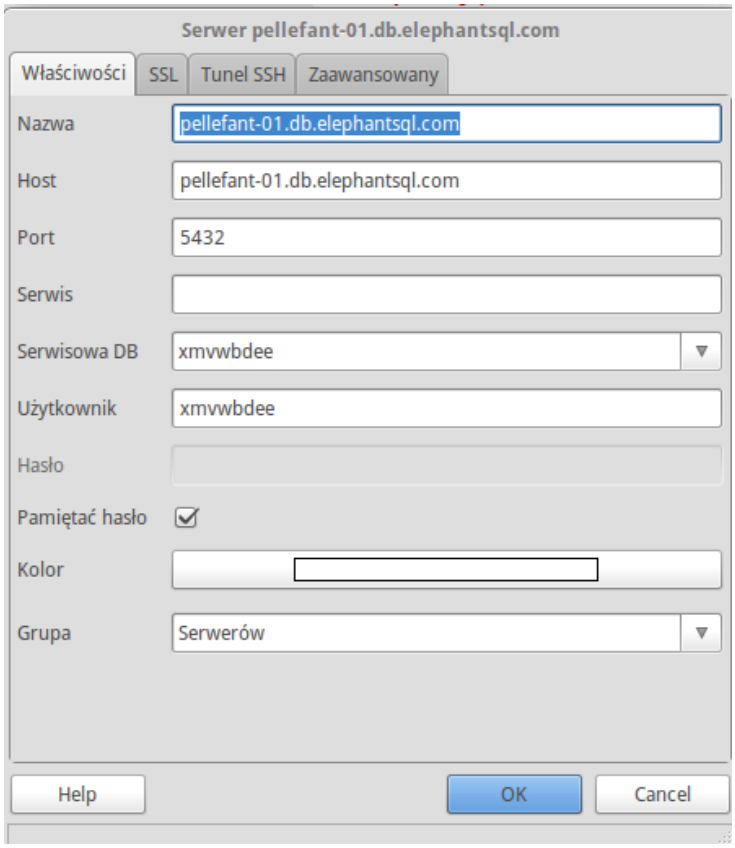
19.Wykaz literatury.

W czasie realizacji projektu korzystałem głównie z materiałów z laboratoriów oraz dokumentacji PostgreSQL oraz SQL

Dodatkowe:

Połączenie z bazą za pomocą pgAdmin

hasło:ge3juV_jk0VgFE2VrDgD-ZOi9D2J5hHp



Wszystkie funkcje użyte w projekcie, dostępne również w pliku funkcje.sql

--Funkcja do usuwania z tabeli Gry

-- TOC entry 306 (class 1255 OID 3031389)

-- Name: delete_gra(character varying); Type: FUNCTION; Schema: public; Owner: xmvwbdee

--

```
CREATE FUNCTION delete_gra(character varying) RETURNS void
    LANGUAGE plpgsql
    AS $_$
declare
x integer;
begin
select into x idgra from gra where nazwa=$1 ;
delete from kategorie where gra_idgra = x;
delete from tworcy where gra_idgra = x;
```

```
delete from gra where idgra= x;

return;

end$_$;



ALTER FUNCTION public.delete_gra(character varying) OWNER TO xmvwbdee;


--Funkcja do dodawania zaawansowanego

-- TOC entry 289 (class 1255 OID 2961967)

-- Name: insert_advanced(character varying, date, character varying, double precision, boolean, character varying, character varying, character varying, text, text); Type: FUNCTION; Schema: public; Owner: xmvwbdee

--


CREATE FUNCTION insert_advanced(character varying, date, character varying, double precision, boolean, character varying, character varying, character varying, text, text) RETURNS void

    LANGUAGE plpgsql

    AS $_$

begin

        insert into gra(nazwa,data_wydania,opis,ocena,multiplayer) values ($1,$2,$3,$4,$5);

        insert into producenci(nazwa) values ($6);

        insert into wydawca(nazwa) values ($7);

        insert into wydawca_pl(nazwa) values ($8);

        insert into
tworcy(producenci_idproducenci,wydawca_idwydawca,wydawca_pl_idwydawca_pl,gra_idgra)
values ((SELECT last_value from producenci_idproducenci_seq),(SELECT last_value from
wydawca_idwydawca_seq),(SELECT last_value from wydawca_pl_idwydawca_pl_seq),(SELECT
last_value from gra_idgra_seq));

        INSERT INTO kategorie(gra_idgra,kategoria_idkategoria)

        SELECT (SELECT last_value from gra_idgra_seq),

        cast(id AS INT)

        FROM unnest(string_to_array($9, ',')) AS dt(id);
```

```
        INSERT INTO platformy(gra_idgra,platforma_idplatforma)
        SELECT (SELECT last_value from gra_idgra_seq),
        cast(id AS INT)
        FROM unnest(string_to_array($10, ',')) AS plat(id);
end;
$_$;
```

ALTER FUNCTION public.insert_advanced(character varying, date, character varying, double precision, boolean, character varying, character varying, character varying, text, text) OWNER TO xmvwbdee;

```
--Funkcja do usuwania kategorii z statystyk (Trigger)
-- TOC entry 875 (class 1255 OID 3039100)
-- Name: kategorie_delete(); Type: FUNCTION; Schema: public; Owner: xmvwbdee
--

CREATE FUNCTION kategorie_delete() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
declare
kategorie integer;
begin
select into kategorie ilosc_kategorii from statystyka;
update statystyka set ilosc_kategorii=kategorie-1;
return new ;
end$$;
```

ALTER FUNCTION public.kategorie_delete() OWNER TO xmvwbdee;

```
--Funkcja do zmiany oceny
-- TOC entry 305 (class 1255 OID 2894593)
-- Name: ocena_change(integer, integer); Type: FUNCTION; Schema: public; Owner: xmvwbdee
--

CREATE FUNCTION ocena_change(integer, integer) RETURNS void
    LANGUAGE plpgsql
    AS $_$
declare x float;
declare count integer;
declare count_new integer;
declare wynik float;
begin
select ocena into x from gra where idgra = $1;
select ilosc_ocen into count from gra where idgra = $1;
wynik=((x*count)+$2)/(count+1);
update gra set ocena=wynik where idgra = $1;
count_new=count+1;
update gra set ilosc_ocen=count_new where idgra = $1;
return;
end;
$_$;


ALTER FUNCTION public.ocena_change(integer, integer) OWNER TO xmvwbdee;

--Funkcja do usuwania platformy z statystyk (Trigger)
-- TOC entry 877 (class 1255 OID 3039111)
-- Name: platforma_delete(); Type: FUNCTION; Schema: public; Owner: xmvwbdee
```

```
--

CREATE FUNCTION platforma_delete() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
declare
platforma integer;
begin
select into platforma ilosc_platform from statystyka;
update statystyka set ilosc_platform=platforma-1;
return new ;
end$$;


ALTER FUNCTION public.platforma_delete() OWNER TO xmvwbdee;


--Funkcja tworzaca statystyki z tablicy gra (Trigger)
-- TOC entry 878 (class 1255 OID 3039092)
-- Name: statystyka(); Type: FUNCTION; Schema: public; Owner: xmvwbdee
--

CREATE FUNCTION statystyka() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
declare
gry integer;
oceny integer;
srednia_new double precision;
begin
```

```
select into gry ilosc_gier from statystyka;
update statystyka set ilosc_gier=gry+1;
select into oceny ilosc_ocen from statystyka;
update statystyka set ilosc_ocen=oceny+1;
select into srednia_new avg(ocena) from wszystko;
update statystyka set srednia = srednia_new;
return new ;
end$$;


ALTER FUNCTION public.statystyka() OWNER TO xmvwbdee;

--Funkcja do usuwania gier z statystyk (trigger)
-- TOC entry 879 (class 1255 OID 3039096)
-- Name: statystyka_delete(); Type: FUNCTION; Schema: public; Owner: xmvwbdee
--

CREATE FUNCTION statystyka_delete() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
declare
gry integer;
oceny integer;
srednia_new double precision;
begin
select into gry ilosc_gier from statystyka;
update statystyka set ilosc_gier=gry-1;
select into oceny ilosc_ocen from statystyka;
update statystyka set ilosc_ocen=oceny-1;
select into srednia_new avg(ocena) from wszystko;
```

```
update statystyka set srednia = srednia_new;
return new ;
end$$;


ALTER FUNCTION public.statystyka_delete() OWNER TO xmvwbdee;

--Funkcja do dodawania kategorii do statystyk (Trigger)
-- TOC entry 874 (class 1255 OID 3039099)
-- Name: statystyka_kategorie(); Type: FUNCTION; Schema: public; Owner: xmvwbdee
--

CREATE FUNCTION statystyka_kategorie() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
declare
kategorie integer;
begin
select into kategorie ilosc_kategorii from statystyka;
update statystyka set ilosc_kategorii=kategorie+1;
return new ;
end$$;


ALTER FUNCTION public.statystyka_kategorie() OWNER TO xmvwbdee;

--Funkcja do dodawania platform do statystyk (Trigger)
-- TOC entry 876 (class 1255 OID 3039110)
-- Name: statystyka_platforma(); Type: FUNCTION; Schema: public; Owner: xmvwbdee
--
```

```
CREATE FUNCTION statystyka_platforma() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
declare
platforma integer;
begin
select into platforma ilosc_platform from statystyka;
update statystyka set ilosc_platform=platforma+1;
return new ;
end$$;


ALTER FUNCTION public.statystyka_platforma() OWNER TO xmvwbdee;
```