·÷Database eksamen:

Vælg et Enterprise

- Streaming tjeneste ala netflix

Definer funktionelle og non-funktionelle krav

De funktionelle krav

1. Brugeren skal kunne oprette en konto med e-mail og hashed password
2. Brugeren skal have mulighed for at tilføje op til 4 brugere til én konto.
3. Der må maks være 2 brugere logget ind en konto på samme tid.
4. Der må maks være 1 enhed på en bruger på samme tid.
5. En bruger skal have mulighed for at logge ind på sin konto.
6. En bruger skal have mulighed for at logge ud af sin konto.
7. Hvis brugeren ikke er logget ind, er det ikke muligt for brugeren at interagere med streamingtjenesten.
8. Hvis en bruger er logget ind, skal brugeren kunne opdatere sine kontooplysninger
9. Hvis en bruger er logget ind, skal det være muligt at slette brugere fra kontoen.
10. Hvis en bruger er logget ind, skal brugeren have mulighed for at tilføje film til en ”watch-this-list”.
11. Som administrator er det muligt at tilføje film til streamingtjenesten.
12. Som administrator skal man kunne sætte film som aktiv eller inaktiv.
13. Hvis en bruger er logget ind, skal brugeren blive præsenteret for alle film opdelt efter genre
14. Hvis en bruger er logget ind, skal brugeren blive præsenteret for de 10 nyeste film.
15. Hvis en bruger er logget ind, skal brugeren blive præsenteret for top-10 mest sete film.
16. Hvis en bruger er logget ind, skal brugeren blive præsenteret for top-10 højest ratede film.
17. Hvis en bruger er logget ind, skal brugeren kunne blive præsenteret for anbefalede film på baggrund af tidligere sete film.
18. Hvis en bruger er logget ind, skal brugeren kunne søge efter genre og blive præsenteret for film i denne genre.
19. Hvis en bruger er logget ind, skal brugeren kunne søge efter film titel og blive præsenteret for denne film.
20. Hvis en bruger er logget ind, skal brugeren kunne søge efter skuespiller og blive præsenteret for alle film hvor skuespilleren medvirker.
21. Hvis en bruger er logget ind, skal brugeren kunne søge efter instruktør og blive præsenteret for alle film som vedkommende har instrueret.
22. Hvis en bruger er logget ind, skal brugeren kunne se en film.
23. Hvis en bruger har set en film, skal brugeren have mulighed for at rate denne.
24. Hvis en bruger er inaktiv i en halv time, udløber session og brugeren logges automatisk ud.
25. Hvis en bruger ser en film, forlænges sessionen varighed med filmens varighed plus 10 minutter, hvorefter brugeren logges automatisk ud.

De non-funktionelle krav

Availability: Graph database skal sættes op som et cluster, for at øge tilgængeligheden og håndtere store mængder af data.

Availability: Replica af alle tre typer databaser.

Plan og motiver vores valg

NoSQL database (key-value) – Redis

- Skal bruges til at håndtere session, samt sikre at maks 2 brugere kan logge ind på en konto ad gangen.
- session:{bruger id} = timestamp
- login:{konto id}: = number of logins
- active:{konto id} = set<bruger id>

Relation mellem id'er fra PostgreSQL til Redis.

Relational database – PostgreSQL

- Skal bruges til at gemme data der relaterer sig til en konto, samt opdatere kontooplysninger og slette brugere fra kontoen.
- En konto skal have: id, fornavn, efternavn, e-mail, password, fødselsdag, telefon nummer, betalings oplysninger, abonnement type (basis, standard, premium)
- En bruger skal have et id, brugernavn og vælge om det er barn (12 eller under) eller ej.

Relation mellem bruger og film: "watched", eventuelt have en rating (hvis brugeren har ratede filmen).

NoSQL database (graph) – Neo4j

- Skal holde på data der relaterer sig til film.
- Skal bruges til hente film ud på baggrund af forskellige parametre.
- Skal bruges til at gemme brugernes film historik.
- En film skal have: titel, varighed, aldersgrænse, udgivelsesår, beskrivelse
- En genre skal have et navn og en relation til film.
- Skuespillere skal have fulde navn, og en relation der indeholder navnet på karakteren i filmen.
- Instruktør skal have fulde navn og en relation til film.

Design og lav databaserne

- Deploy databaserne (enten lokalhost eller anden server)
- Lav en eller en flere simple applikationer

- Skriv op til 7 siders rapport – dokumentering / instruktioner til applikationen

Script til docker - postgreSQL

I terminal/command prompt skriv:
docker run --name flixbusterdb -p 5432:5432 -d -e POSTGRES_PASSWORD=chokobanan -e POSTGRES_USER=chokobanan -e POSTGRES_DB=flixbuster -v pgdataflixbuster:/var/lib/postgresql/data postgres

For at få adgang til container skriv følgende:
docker exec -it flixbusterdb psql -U chokobanan flixbuster

PostgreSQL tabeller

```
DROP TABLE IF EXISTS Flixbuster_User;
DROP TABLE IF EXISTS Payment;
DROP TABLE IF EXISTS Account;
drop type if EXISTS Abonnement;
```

```
create type Abonnement as enum ('basis', 'standard', 'premium');

CREATE TABLE Account (
   account_id SERIAL PRIMARY KEY,
   firstname varchar(30) NOT null,
   lastname varchar(30) not null,
   email varchar(50) unique not null,
   passwd varchar(20) not null,
   birthday date not null,
   phonenumber char(15),
   current_abonnement Abonnement
   );

create table Payment (
                payment_id int primary key references Account NOT NULL,
                cardnumber char(16) not null,
                securitycode char(3) not null,
                expirationdate char(4) not null
                );


CREATE TABLE Flixbuster_User (
   user_id SERIAL PRIMARY key,
   account_id int REFERENCES Account,
   username varchar(30) not null,
   child bool default false
   );
```

Insæt værdier

```
CREATE OR REPLACE procedure create_account (account_firstname varchar(30), account_lastname varchar(30), account_email varchar(50), account_passwd varchar(20), account_birthday date, account_phonenumber char(15), account_current_abonnement Abonnement,
```

```
account_cardnumber char(16), account_securitycode char(3), account_expirationdate char(4))

as $$
declare
                id int;
begin

INSERT INTO account (firstname, lastname, email, passwd, birthday, phonenumber,
current_abonnement) VALUES (account_firstname, account_lastname, account_email,
account_passwd, account_birthday, account_phonenumber, account_current_abonnement)
RETURNING account_id into id;

INSERT INTO payment (payment_id, cardnumber, securitycode, expirationdate) SELECT id,
account_cardnumber, account_securitycode, account_expirationdate;

INSERT INTO flixbuster_user (account_id, username) select id, account_firstname;

end
$$ language plpgsql;

call create_account('Kasper', 'Frederiksen', 'kasper_frederiksen@gmail.com', '1234', '1995-10-10',
'61271486', 'basis', '2097643167861256', '165', '0123');
call create_account('Jesper', 'Larsen', 'jesper_larsen@gmail.com', '4321', '1979-03-05', '54601814',
'standard', '2097543218765876', '912', '0522');
call create_account('Jonathan', 'Klausen', 'jonathan_klausen@gmail.com', '9876', '1965-12-24',
'61127052', 'premium', '1278479276317942', '764', '1021');

insert into flixbuster_user (account_id, username) values (3, 'Betty');
insert into flixbuster_user (account_id, username, child) values (4, 'Storm', TRUE);
```

Graph database – Neo4j

1. Åbn Neo4j -> ”new” -> ”add” -> “local DBMS” ->

   Name: FlixbusterDB
   Password: flixbuster
   Version: 4.5.2

2. “create”
3. “start” -> when “running” -> “open neo4j browser”
4. Kopier data ind I browseren:

Key-value par – Redis

Åbn en terminal/command prompt og skriv:
docker run --name FlixbusterRedis -v redis-data-flixbuster:/data -d redis:alpine

For at få adgang til containeren og åbne en cli, skriv følgende:
docker exec -it FlixbusterRedis redis-cli

Overordnet planlægning:

- I klienten:
  - Lav forbindelse til PostgreSQL
  - Lav forbindelse til Neo4j
  - Lav forbindelse til Redis
  - Lav API

- Opret/lav key-value par til redis database

-