A5: Esquema relacional, validação e melhoramento

Este artefacto contém o esquema relacional obtido através do mapeamento do modelo conceptual de dados. O esquema relacional inclui o esquema de relações, atributos, domínios, chaves primárias, chaves estrangeiras e outras regras de integridade: UNIQUE, DEFAULT, NOT NULL, CHECK.

1. Esquema Relacional

R01	User(id, username **UK NN**, password **NN**, email **UK NN**, regist_date **NN DF** Today, first_name **NN**, last_name **NN**, image_path, city_id → City **NN**)
R02	Event(id, name **NN**, date **NN** date > Today, description **NN**, owner_id → User **NN**, localization_id → Localization **NN**, image_id → Image, type **NN CK** type IN Types_of_event, category **NN CK** category IN Categories)
R03	Done(event_id → Event, rating **NN CK** rating > 0 AND rating < = 5)
R04	Not_done(event_id → Event)
R05	Paticipant(id, (user_id → User**,** event_id → Event) **UK NN**)
R06	Owner(id (user_id → User**,** event_id → Event) **UK NN**)
R07	Image(id, path **NN UK**)
R08	Localization(id, name, address **NN**, latitude, longitude, city_id → City **NN**)
R09	City(id, name **UK NN**, country_id → Country **NN**)
R10	Country(id, name **UK NN**)
R11	Post(id, description **NN**, date **NN DF** Today, event_id → Event **NN**, user_id → User **NN**, image_id → Image)
R12	Poll(id, post_id → Post **NN**)
R13	Option(id, description **NN**, poll_id → Poll **NN**)
R14	Admin(id, username **UK NN**, password **NN**, email **UK NN**)
R15	Friend_request(id, answer, sender_id → User **NN,** receiver_id → User **NN**)
R16	Friend_activity(id, sender_id → User **NN,** receiver_id → User **NN**, event_id → Event **NN**)
R17	Event_invite(id, answer, event_id → Not_done **NN**, owner_id → User **NN**, receiver_id → User **NN**)

2. Domínios

Today	DATE DEFAULT CURRENT_DATE
Types_of_event	ENUM('Public', Private')
Categories	ENUM('Music', 'Sports', 'Entertainment', 'Educational', 'Business', 'Other')

3. Dependências funcionais e validação do esquema

Table R01 (User)	
Keys: {id}, {username}, {email}	
Dependências Funcionais	
FD0101	{id} → {username, email, password, regist_date, first_name, last_name, image, city_id}
FD0102	{username} → {id, email, password, regist_date, first_name, last_name, image, city_id}
FD0103	{email} → {id, username, password, regist_date, first_name, last_name, image, city_id}
Forma Normal	BCNF

Table R02 (Event)	
Keys: {id}	
Dependências Funcionais	
FD0201	{id} → {name, date, type, category, description, localization_id, owner_id, image_id}
Forma Normal	BCNF

Table R03 (Done)	
Keys: {event_id}	
Dependências Funcionais	
FD0301	{event_id} → {rating}
Forma Normal	BCNF

Table R04 (Not_done)	
Keys: {event_id}	
Dependências Funcionais	
(none)	
Forma Normal	BCNF

Table R05 (Participant)	
Keys: {id}, {user_id, event_id}	
Dependências Funcionais	
FD0501	{id} → {user_id, event_id}
FD0502	{user_id, event_id} → {id}
Forma Normal	BCNF

Table R06 (Owner)	
Keys: {id}, {user_id, event_id}	
Dependências Funcionais	
FD0601	{id} → {user_id, event_id}
FD0602	{user_id, event_id} → {id}
Forma Normal	BCNF

Table R07 (Image)	
Keys: {id}	
Dependências Funcionais	
FD0701	{id} → {path}
Forma Normal	BCNF

Table R08 (Localization)	
Keys: {id}	
Dependências Funcionais	
FD0801	{id} → {name, address, latitude, longitude, city_id}
Forma Normal	BCNF

Table R09 (City)	
Keys: {id}	
Dependências Funcionais	
FD0901	{id} → {name, country_id}
Forma Normal	BCNF

Table R10 (Country)	
Keys: {id}	
Dependências Funcionais	
FD1001	{id} → {name}
Forma Normal	BCNF

Table R11 (Post)	
Keys: {id}	
Dependências Funcionais	
FD1101	{id} → {description, date, event_id, user_id, image_id}
Forma Normal	BCNF

Table R12 (Poll)	
Keys: {id}	
Dependências Funcionais	
FD1201	{id} → {post_id}
Forma Normal	BCNF

Table R13 (Option)	
Keys: {id}	
Dependências Funcionais	
FD1301	{id} → {description, poll_id}
Forma Normal	BCNF

Table R14 (Admin)	
Keys: {id}, {username}, {email}	
Dependências Funcionais	
FD1401	{id} → {username, email, password}
FD1402	{username} → {id, email, password}
FD1403	{email} → {id, username, password}
Forma Normal	BCNF

Table R15 (Friend_request)	
Keys: {id}	
Dependências Funcionais	
FD1501	{id} → {answer, sender_id, receiver_id}
Forma Normal	BCNF

Table R16 (Friend_activity)	
Keys: {id}	
Dependências Funcionais	
FD1601	{id} → {sender_id, receiver_id, event_id}
Forma Normal	BCNF

Table R17 (Event_invite)	
Keys: {id}	
Dependências Funcionais	
FD1701	{id} → {answer, event_id, owner_id, receiver_id }
Forma Normal	BCNF

4. SQL Código

Sql code

```sql
--Types Enums

DROP TYPE IF EXISTS categories CASCADE;
CREATE TYPE categories AS ENUM(
    'Music',
    'Sports',
    'Entertainment',
    'Educational',
    'Business',
    'Other'
);
DROP TYPE IF EXISTS types_of_event CASCADE;
CREATE TYPE types_of_event AS ENUM(
    'Public',
    'Private'
);

--Tables
DROP TABLE IF EXISTS admins CASCADE;
CREATE TABLE admins (
    id SERIAL NOT NULL,
    username text NOT NULL,
    password text NOT NULL,
    email text NOT NULL,
    CONSTRAINT admins_pk PRIMARY KEY (id),
    CONSTRAINT admins_email_uk UNIQUE (email)
);

DROP TABLE IF EXISTS cities CASCADE;
CREATE TABLE cities (
    id SERIAL NOT NULL,
    name text NOT NULL,
    country_id INTEGER NOT NULL,
    CONSTRAINT cities_pk PRIMARY KEY (id),
    CONSTRAINT cities_name_uk UNIQUE (name)
```

```
);

DROP TABLE IF EXISTS countries CASCADE;
CREATE TABLE countries (
    id SERIAL NOT NULL,
    name text NOT NULL,
    CONSTRAINT countries_pk PRIMARY KEY (id),
    CONSTRAINT countries_name_uk UNIQUE (name)
);

DROP TABLE IF EXISTS dones CASCADE;
CREATE TABLE dones (
    event_id INTEGER NOT NULL,
    rating INTEGER NOT NULL,
    CONSTRAINT dones_pk PRIMARY KEY (event_id),
    CONSTRAINT rating_ck CHECK (((rating >= 1) AND (rating <= 5)))
);

DROP TABLE IF EXISTS events CASCADE;
CREATE TABLE events (
    id SERIAL NOT NULL,
    name text NOT NULL,
    date TIMESTAMP WITH TIME zone NOT NULL,
    description text NOT NULL,
    owner_id INTEGER NOT NULL,
    localization_id INTEGER NOT NULL,
    image_id INTEGER NOT NULL,
    type event_type NOT NULL,
    category categories NOT NULL,
    CONSTRAINT events_pk PRIMARY KEY (id),
    CONSTRAINT date_ck CHECK ((date > now()))
);

DROP TABLE IF EXISTS event_invites CASCADE;
CREATE TABLE event_invites (
    id SERIAL NOT NULL,
    answer text NOT NULL,
    event_id INTEGER NOT NULL,
    owner_id INTEGER NOT NULL,
    receiver_id INTEGER NOT NULL,
    CONSTRAINT event_invites_pk PRIMARY KEY (id)
);

DROP TABLE IF EXISTS friend_activities CASCADE;
CREATE TABLE friend_activities (
    id SERIAL NOT NULL,
    sender_id INTEGER NOT NULL,
    receiver_id INTEGER NOT NULL,
    event_id INTEGER NOT NULL,
```

```
    CONSTRAINT friend_activities_pk PRIMARY KEY (id),
    CONSTRAINT friend_activities_event_id_fk FOREIGN KEY (event_id)
REFERENCES
    events(id) ON DELETE CASCADE
);

DROP TABLE IF EXISTS friend_requests CASCADE;
CREATE TABLE friend_requests (
    id SERIAL NOT NULL,
    answer text NOT NULL,
    sender_id INTEGER NOT NULL,
    receiver_id INTEGER NOT NULL,
    event_id INTEGER NOT NULL,
    CONSTRAINT friend_requests_pk PRIMARY KEY (id)
);

DROP TABLE IF EXISTS images CASCADE;
CREATE TABLE images (
    id SERIAL NOT NULL,
    path text NOT NULL,
    CONSTRAINT images_pk PRIMARY KEY (id),
    CONSTRAINT images_path_uk UNIQUE (path)
);

DROP TABLE IF EXISTS localizations CASCADE;
CREATE TABLE localizations (
    id SERIAL NOT NULL,
    name text NOT NULL,
    address text NOT NULL,
    latitude FLOAT,
    longitude FLOAT,
    city_id INTEGER NOT NULL,
    CONSTRAINT localizations_pk PRIMARY KEY (id),
    CONSTRAINT localizations_city_id_fk FOREIGN KEY (city_id) REFERENCES
    cities(id) ON DELETE SET NULL
);

DROP TABLE IF EXISTS not_dones CASCADE;
CREATE TABLE not_dones (
    event_id INTEGER NOT NULL,
    CONSTRAINT not_dones_pk PRIMARY KEY (event_id),
    CONSTRAINT not_dones_event_id_fk FOREIGN KEY (event_id) REFERENCES
    events(id) ON UPDATE CASCADE
);

DROP TABLE IF EXISTS options CASCADE;
CREATE TABLE options (
    id SERIAL NOT NULL,
    description text NOT NULL,
```

```
    poll_id INTEGER NOT NULL,
    CONSTRAINT options_pk PRIMARY KEY (id)
);

DROP TABLE IF EXISTS owners CASCADE;
CREATE TABLE owners (
    id SERIAL NOT NULL,
    user_id INTEGER NOT NULL,
    event_id INTEGER NOT NULL,
    CONSTRAINT owners_pk PRIMARY KEY (id),
    CONSTRAINT owners_user_id_event_id_uk UNIQUE (user_id, event_id),
    CONSTRAINT owners_event_id_fk FOREIGN KEY (event_id) REFERENCES
    events(id) ON DELETE SET NULL
);

DROP TABLE IF EXISTS participants CASCADE;
CREATE TABLE participants (
    id SERIAL NOT NULL,
    user_id INTEGER NOT NULL,
    event_id INTEGER NOT NULL,
    CONSTRAINT participants_pk PRIMARY KEY (id),
    CONSTRAINT participants_user_id_event_id_uk UNIQUE (user_id,
event_id),
    CONSTRAINT participants_event_id_fk FOREIGN KEY (event_id) REFERENCES
    events(id) ON DELETE SET NULL
);

DROP TABLE IF EXISTS polls CASCADE;
CREATE TABLE polls (
    id SERIAL NOT NULL,
    post_id INTEGER NOT NULL,
    CONSTRAINT polls_pk PRIMARY KEY (id)
);

DROP TABLE IF EXISTS posts CASCADE;
CREATE TABLE posts (
    id SERIAL NOT NULL,
    description text NOT NULL,
    date TIMESTAMP WITH TIME zone NOT NULL,
    event_id INTEGER NOT NULL,
    user_id INTEGER NOT NULL,
    image_id INTEGER NOT NULL,
    CONSTRAINT posts_pk PRIMARY KEY (id),
    CONSTRAINT posts_event_id_fk FOREIGN KEY (event_id) REFERENCES
    events(id) ON DELETE CASCADE,
    CONSTRAINT posts_image_id_fk FOREIGN KEY (image_id) REFERENCES
    images(id) ON DELETE SET NULL
);
```

```sql
DROP TABLE IF EXISTS users CASCADE;
CREATE TABLE users (
    id SERIAL NOT NULL,
    username text NOT NULL,
    password text NOT NULL,
    email text NOT NULL,
    regist_date TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL,
    first_name text NOT NULL,
    last_name text NOT NULL,
    image_path text NOT NULL,
    city_id INTEGER NOT NULL,
    CONSTRAINT users_pk PRIMARY KEY (id),
    CONSTRAINT users_name_uk UNIQUE (username),
    CONSTRAINT users_email_uk UNIQUE (email),
    CONSTRAINT users_city_id_fk FOREIGN KEY (city_id) REFERENCES
    cities(id) ON DELETE SET NULL
);


ALTER TABLE ONLY cities
    ADD CONSTRAINT cities_country_id_fk FOREIGN KEY (country_id)
REFERENCES
    countries(id) ON DELETE SET NULL;

ALTER TABLE ONLY dones
    ADD CONSTRAINT dones_event_id_fk FOREIGN KEY (event_id) REFERENCES
     events(id) ON UPDATE CASCADE;

ALTER TABLE ONLY events
    ADD CONSTRAINT events_owner_id_fk FOREIGN KEY (owner_id) REFERENCES
    users(id) ON UPDATE CASCADE;

ALTER TABLE ONLY events
    ADD CONSTRAINT events_localization_id_fk FOREIGN KEY
(localization_id) REFERENCES
    localizations(id) ON DELETE SET NULL;

ALTER TABLE ONLY events
    ADD CONSTRAINT events_image_id_fk FOREIGN KEY (image_id) REFERENCES
    images(id) ON DELETE SET NULL;

ALTER TABLE ONLY event_invites
    ADD CONSTRAINT event_invites_event_id_fk FOREIGN KEY (event_id)
REFERENCES
    not_dones(event_id) ON DELETE CASCADE;

ALTER TABLE ONLY event_invites
    ADD CONSTRAINT event_invites_owner_id_fk FOREIGN KEY (owner_id)
REFERENCES
```

```
    owners(id) ON DELETE SET NULL;

ALTER TABLE ONLY event_invites
    ADD CONSTRAINT event_invites_receiver_id_fk FOREIGN KEY (receiver_id)
REFERENCES
    users(id) ON DELETE SET NULL;

ALTER TABLE ONLY friend_activities
    ADD CONSTRAINT friend_activities_sender_id_fk FOREIGN KEY (sender_id)
REFERENCES
    users(id) ON DELETE SET NULL;

ALTER TABLE ONLY friend_activities
    ADD CONSTRAINT friend_activities_receiver_id_fk FOREIGN KEY
(receiver_id) REFERENCES
    users(id) ON DELETE SET NULL;

ALTER TABLE ONLY friend_requests
    ADD CONSTRAINT friend_requests_sender_id_fk FOREIGN KEY (sender_id)
REFERENCES
    participants(id) ON DELETE CASCADE;

ALTER TABLE ONLY friend_requests
    ADD CONSTRAINT friend_requests_receiver_id_fk FOREIGN KEY
(receiver_id) REFERENCES
    users(id) ON DELETE CASCADE;

ALTER TABLE ONLY options
    ADD CONSTRAINT options_poll_id_fk FOREIGN KEY (poll_id) REFERENCES
    polls(id) ON DELETE CASCADE;

ALTER TABLE ONLY owners
    ADD CONSTRAINT owners_user_id_fk FOREIGN KEY (user_id) REFERENCES
    users(id) ON UPDATE CASCADE;

ALTER TABLE ONLY participants
    ADD CONSTRAINT participants_user_id_fk FOREIGN KEY (user_id)
REFERENCES
    users(id) ON UPDATE CASCADE;

ALTER TABLE ONLY polls
    ADD CONSTRAINT polls_post_id_fk FOREIGN KEY (post_id) REFERENCES
    posts(id) ON DELETE CASCADE;

ALTER TABLE ONLY posts
    ADD CONSTRAINT posts_user_id_fk FOREIGN KEY (user_id) REFERENCES
    users(id) ON DELETE CASCADE;
```

Histórico de Revisão

Alterações feitas à primeira submissão:

1. Correção das chaves do partcipant e owner
2. Correção de algumas dependências funcionais
3. Passagem de algumas chaves e restrições para dentro das classes;

GRUPO1765, 21/03/2018

- Mariana Duarte Guimarães, up201307777@fe.up.pt
- Rui Emanuel Cabral de Almeida Quaresma, up201503005@fe.up.pt
- Rui Pedro Machado Araújo, up201403263@fe.up.pt
- Tiago Duarte Carvalho, up201504461@fe.up.pt