Project Management

A5: Relational schema, validation and schema refinement

1. Relational Schema

R01	user (id, name **NN**, username **UK NN**, email **UK NN**, image, password **NN**)
R02	project (id, name **NN**, description **NN**, isPublic **NN**)
R03	sprint (id, name **NN**, deadline **NN CK** deadline > Today, effort **NN**, project_id -> project **NN**, user_creator_id -> user **NN**)
R04	task (id, name **NN**, description **NN**, effort **NN**, project_id -> project **NN**, sprint_id -> sprint)
R05	thread (id, name **NN**, description, date **NN DF** Today, project_id -> project **NN**, user_creator_id -> user **NN**)
R06	comment (id, content **NN**, date **NN DF** Today, user_id -> user **NN**, task_id -> task, thread_id -> thread) (*There is one more restriction, but it's only in the sql file for reasons of perceptibility*)
R07	category (id, name **NN**)
R08	project_members(user_id -> User, project_id -> Project, isCoordinator **NN**, date **NN DF** Today)
R09	administrator(id, username **NN**, password **NN**)
R10	report(id, date **NN DF** Today, type **NN CK** type **IN** ReportTypes, summary **NN**, author_id->user, comment_reported_id -> comment, user_reported_id -> user) (*There is one more restriction, but it's only in the sql file for reasons of perceptibility*)
R11	notification(id, date **NN DF** Today, notificationType **NN CK** notificationType **IN** NotificationType, user_id -> user **NN**, project_id -> project, comment_id -> comment, user_action_id -> user) (*There is one more restriction, but it's only in the sql file for reasons of perceptibility*)
R12	invite(id, date **NN DF** Today, user_invited_id -> user **NN**, project_id -> project **NN**, user_who_invited_id -> user)
R13	taskStateRecord(id, date **NN DF** Today, state **NN CK** state **IN** State, user_completed_id -> user **NN**, task_id -> task **NN**)
R14	sprintStateRecord(id, date **NN DF** Today(), state **NN CK** state **IN**

	SprintState, sprint_id -> sprint **NN**)
R15	project_categories(project_id -> project **NN**, category_id -> category **NN**)

where UK means UNIQUE KEY, NN means NOT NULL, DF means DEFAULT and CK means CHECK.

Note: The administrator is a separate unit from the user because they are different entities and it would make more sense for them to be separate.

2. Domains

Today	DATE DEFAULT CURRENT_DATE
ReportTypes	ENUM('CommentReported','UserReported') (*It is an ENUM because it is more perceptible this way, instead of a boolean)*
State	ENUM('Completed', 'Assigned', 'Created')
SprintState	ENUM('Completed','Created','Outdated')
NotificationType	ENUM('Comment','CommentReported','Promotion','RemovedFromProject','Invite','Request')

3. Functional Dependencies and Schema Validation

Table R01 (user)	
Keys: {id,username,email}	
Functional Dependencies	
FD0101	{id}->{name,username,email,image,password}
FD0102	{username}->{id,name,email,image,password}
FD0103	{email}->{id,name,username,image,password}
Normal Form	BCNF

Table R02 (project)	
Keys: {id}	
Functional Dependencies	
FD0201	{id}->{name,description,isPublic}
Normal Form	BCNF

Table R03 (sprint)	
Keys: {id}	
Functional Dependencies	
FD0301	{id}->{name, deadline, effort, project_id, user_creator_id}
Normal Form	BCNF

Table R04 (task)	
Keys: {id}	
Functional Dependencies	
FD0401	{id}->{name, description, effort, project_id, sprint_id}
Normal Form	BCNF

Table R05 (thread)	
Keys: {id}	
Functional Dependencies	
FD0501	{id}->{name, description, date, project_id, user_creator_id}
Normal Form	BCNF

Table R06 (comment)	
Keys: {id}	
Functional Dependencies	
FD0601	{id}->{content, date, user_id, task_id, thread_id}
Normal Form	BCNF

Table R07 (category)	
Keys: {id}	
Functional Dependencies	
FD0701	{id}->{name}
Normal Form	BCNF

Table R08 (project_members)	
Keys: {user_id, project_id}	
Functional Dependencies	
FD0801	{user_id, project_id}->{name, date}
Normal Form	BCNF

Table R09 (administrator)	
Keys: {id,username}	
Functional Dependencies	
FD0901	{id}->{username, password}
FD0902	{username}->{id,password}
Normal Form	BCNF

Table R10 (report)	
Keys: {id}	
Functional Dependencies	
FD1001	{id}->{date, type, summary, user_id, commnt_reported_id, user_reported_id}
Normal Form	BCNF

Table R11 (notification)	
Keys: {id}	
Functional Dependencies	
FD1101	{id}->{date,notificationType, user_id, project_id, comment_id, user_action_id}
Normal Form	BCNF

Table R12 (invite)	
Keys: {id}	
Functional Dependencies	
FD1201	{id}->{date, user_invited_id, project_id, user_who_invited_id}
Normal Form	BCNF

Table R13 (taskStateRecord)	
Keys: {id}	
Functional Dependencies	
FD1301	{id}->{date, state, user_completed_id, task_id}
Normal Form	BCNF

Table R14 (sprintStateRecord)	
Keys: {id}	
Functional Dependencies	
FD1401	{id}->{date, state, sprint_id}
Normal Form	BCNF

Table R15 (project_categories)	
Keys: {project_id, category_id}	
Functional Dependencies	
(none)	
Normal Form	BCNF

4. SQL Code

```
CREATE TABLE User (
      id SERIAL NOT NULL,
      name text NOT NULL,
      username text NOT NULL,
      email text NOT NULL,
      image text,
      password text NOT NULL
);

CREATE TABLE Project (
      id SERIAL NOT NULL,
      name text NOT NULL,
      description text NOT NULL,
      isPublic boolean NOT NULL
);

CREATE TABLE Sprint (
      id SERIAL NOT NULL,
      name text NOT NULL,
      deadline TIMESTAMP WITH TIME zone NOT NULL,
      effort INTEGER NOT NULL,
      project_id INTEGER NOT NULL,
      user_creator_id INTEGER NOT NULL,
      CONSTRAINT deadline CHECK (deadline > now())
);

CREATE TABLE Task (
      id SERIAL NOT NULL,
      name text NOT NULL,
      description text,
      effort INTEGER NOT NULL,
      project_id INTEGER NOT NULL,
      sprint_id INTEGER
);

CREATE TABLE Thread (
      id SERIAL NOT NULL,
      name text NOT NULL,
      description text,
```

```
      "date" TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL,
      project_id INTEGER NOT NULL,
      user_creator_id INTEGER NOT NULL
);

CREATE TABLE Comment (
      id SERIAL NOT NULL,
      content text NOT NULL,
      "date" TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL,
      user_id INTEGER NOT NULL,
      task_id INTEGER,
      thread_id INTEGER,
      CONSTRAINT belongs CHECK ((task_id != NULL AND thread_id =
NULL) OR (task_id = NULL AND thread_id != NULL))
);

CREATE TABLE Category (
      id SERIAL NOT NULL,
      name text NOT NULL
);

CREATE TABLE Project_members (
      user_id INTEGER NOT NULL,
      project_id INTEGER NOT NULL,
      isCoordinator boolean NOT NULL,
      "date" TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL
);

CREATE TABLE Administrator (
      id SERIAL NOT NULL,
      username text NOT NULL,
      password text NOT NULL
);

CREATE TABLE Report (
      id SERIAL NOT NULL,
      "date" TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL,
      type text NOT NULL,
      summary text NOT NULL,
      user_id INTEGER NOT NULL,
      comment_reported_id INTEGER,
```

```
      user_reported_id INTEGER,
      CONSTRAINT reportType CHECK ((type =
ANY(ARRAY['CommentReported'::text, 'UserReported'::text]))),
      CONSTRAINT typeConstraint CHECK ((type = 'CommentReported'
AND comment_reported_id != NULL AND user_reported_id = NULL) OR
                                                                (type
= 'UserReported' AND user_reported_id != NULL AND
comment_reported_id = NULL))
);

CREATE TABLE Notification (
      id SERIAL NOT NULL,
      "date" TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL,
      notification_type text NOT NULL,
      user_id INTEGER NOT NULL,
      project_id INTEGER,
      comment_id INTEGER,
      user_action_id INTEGER,
      CONSTRAINT notificationType CHECK ((notification_type =
ANY(ARRAY['Comment'::text, 'CommentReported'::text,
'Promotion'::text, 'RemovedFromProject'::text, 'Invite'::text,
'Request'::text]))),
      CONSTRAINT notificationConstraint CHECK ((notification_type =
'Comment' AND comment_id != NULL) OR

(notification_type = 'CommentReported' AND comment_id != NULL) OR

(notification_type = 'Promotion' AND project_id != NULL AND
user_action_id != NULL) OR

(notification_type = 'RemovedFromProject' AND project_id != NULL)
OR

(notification_type = 'Invite' AND project_id != NULL AND
user_action_id != NULL) OR

(notification_type = 'Request' AND project_id != NULL))
);

CREATE TABLE Invite (
      id SERIAL NOT NULL,
```

```
      "date" TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL,
      user_invited_id INTEGER NOT NULL,
      project_id INTEGER NOT NULL,
      user_who_invited_id INTEGER NOT NULL,
);

CREATE TABLE TaskStateRecord(
      id SERIAL NOT NULL,
      "date" TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL,
      state text NOT NULL,
      user_completed_id INTEGER NOT NULL,
      task_id INTEGER NOT NULL,
      CONSTRAINT state CHECK ((state = ANY(ARRAY['Completed'::text,
'Assigned'::text, 'Created'::text])))
);

CREATE TABLE SprintStateRecord(
      id SERIAL NOT NULL,
      "date" TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL,
      state text NOT NULL,
      sprint_id INTEGER NOT NULL,
      CONSTRAINT state CHECK ((state = ANY(ARRAY['Completed'::text,
'Outdated'::text, 'Created'::text])))
);

CREATE TABLE Project_categories (
      project_id INTEGER NOT NULL,
      category_id INTEGER NOT NULL
);


/* Primary Keys and Uniques*/

ALTER TABLE ONLY User
      ADD CONSTRAINT user_pkey PRIMARY KEY (id);

ALTER TABLE ONLY User
    ADD CONSTRAINT user_email_key UNIQUE (email);

ALTER TABLE ONLY User
    ADD CONSTRAINT user_username_key UNIQUE (username);
```

```
ALTER TABLE ONLY Project
      ADD CONSTRAINT project_pkey PRIMARY KEY (id);

ALTER TABLE ONLY Sprint
      ADD CONSTRAINT sprint_pkey PRIMARY KEY (id);

ALTER TABLE ONLY Task
      ADD CONSTRAINT task_pkey PRIMARY KEY (id);

ALTER TABLE ONLY Thread
      ADD CONSTRAINT thread_pkey PRIMARY KEY (id);

ALTER TABLE ONLY Comment
      ADD CONSTRAINT comment_pkey PRIMARY KEY (id);

ALTER TABLE ONLY Category
      ADD CONSTRAINT category_pkey PRIMARY KEY (id);

ALTER TABLE ONLY Project_members
      ADD CONSTRAINT project_members_pkey PRIMARY KEY (user_id,
project_id);

ALTER TABLE ONLY Administrator
      ADD CONSTRAINT administrator_pkey PRIMARY KEY (id);

ALTER TABLE ONLY Report
      ADD CONSTRAINT report_pkey PRIMARY KEY (id);

ALTER TABLE ONLY Notification
      ADD CONSTRAINT notification_pkey PRIMARY KEY (id);

ALTER TABLE ONLY Invite
      ADD CONSTRAINT invite_pkey PRIMARY KEY (id);

ALTER TABLE ONLY TaskStateRecord
      ADD CONSTRAINT taskstaterecord_pkey PRIMARY KEY (id);

ALTER TABLE ONLY SprintStateRecord
      ADD CONSTRAINT sprintstaterecord_pkey PRIMARY KEY (id);
```

```
ALTER TABLE ONLY Project_categories
      ADD CONSTRAINT project_categories_pkey PRIMARY KEY
(project_id, category_id);


/* Foreign Keys */

ALTER TABLE ONLY Sprint
      ADD CONSTRAINT task_id_project_fkey FOREIGN KEY (project_id)
REFERENCES Project(id) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE ONLY Sprint
      ADD CONSTRAINT sprint_id_user_creator_fkey FOREIGN KEY
(user_creator_id) REFERENCES User(id) ON UPDATE CASCADE;

ALTER TABLE ONLY Task
      ADD CONSTRAINT task_id_user_project_fkey FOREIGN KEY
(project_id) REFERENCES Project(id) ON UPDATE CASCADE ON DELETE
CASCADE;

ALTER TABLE ONLY Task
      ADD CONSTRAINT task_id_user_sprint_fkey FOREIGN KEY
(sprint_id) REFERENCES Sprint(id) ON UPDATE CASCADE;

ALTER TABLE ONLY Thread
      ADD CONSTRAINT thread_id_project_fkey FOREIGN KEY
(project_id) REFERENCES Project(id) ON UPDATE CASCADE ON DELETE
CASCADE;

ALTER TABLE ONLY Thread
      ADD CONSTRAINT thread_id_user_creator_fkey FOREIGN KEY
(user_creator_id) REFERENCES User(id) ON UPDATE CASCADE;

ALTER TABLE ONLY Comment
      ADD CONSTRAINT comment_id_user_fkey FOREIGN KEY (user_id)
REFERENCES User(id) ON UPDATE CASCADE;

ALTER TABLE ONLY Comment
      ADD CONSTRAINT comment_id_task_fkey FOREIGN KEY (task_id)
REFERENCES Task(id) ON UPDATE CASCADE ON DELETE CASCADE;
```

```
ALTER TABLE ONLY Comment
     ADD CONSTRAINT comment_id_thread_fkey FOREIGN KEY (thread_id)
REFERENCES Thread(id) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE ONLY Project_members
     ADD CONSTRAINT members_id_user_fkey FOREIGN KEY (user_id)
REFERENCES User(id) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE ONLY Project_members
     ADD CONSTRAINT members_id_project_fkey FOREIGN KEY
(project_id) REFERENCES Project(id) ON UPDATE CASCADE ON DELETE
CASCADE;

ALTER TABLE ONLY Report
     ADD CONSTRAINT report_id_user_fkey FOREIGN KEY (user_id)
REFERENCES User(id) ON UPDATE CASCADE;

ALTER TABLE ONLY Report
     ADD CONSTRAINT report_id_comment_reported_fkey FOREIGN KEY
(comment_reported_id) REFERENCES Comment(id) ON UPDATE CASCADE;

ALTER TABLE ONLY Report
     ADD CONSTRAINT report_id_user_reported_fkey FOREIGN KEY
(user_reported_id) REFERENCES User(id) ON UPDATE CASCADE;

ALTER TABLE ONLY Notification
     ADD CONSTRAINT notification_id_user_fkey FOREIGN KEY
(user_id) REFERENCES User(id) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE ONLY Notification
     ADD CONSTRAINT notification_id_project_fkey FOREIGN KEY
(project_id) REFERENCES Project(id) ON UPDATE CASCADE ON DELETE
CASCADE;

ALTER TABLE ONLY Notification
     ADD CONSTRAINT notification_id_comment_fkey FOREIGN KEY
(comment_id) REFERENCES Comment(id) ON UPDATE CASCADE ON DELETE
CASCADE;

ALTER TABLE ONLY Notification
     ADD CONSTRAINT notification_id_user_action_fkey FOREIGN KEY
```

```
(user_action_id) REFERENCES User(id) ON UPDATE CASCADE;

ALTER TABLE ONLY Invite
     ADD CONSTRAINT invite_id_user_fkey FOREIGN KEY
(user_invited_id) REFERENCES User(id) ON UPDATE CASCADE ON DELETE
CASCADE;

ALTER TABLE ONLY Invite
     ADD CONSTRAINT invite_id_user_who_invited_fkey FOREIGN KEY
(user_who_invited_id) REFERENCES User(id) ON UPDATE CASCADE ON
DELETE CASCADE;

ALTER TABLE ONLY Invite
     ADD CONSTRAINT invite_id_project_fkey FOREIGN KEY
(project_id) REFERENCES Project(id) ON UPDATE CASCADE ON DELETE
CASCADE;

ALTER TABLE ONLY TaskStateRecord
     ADD CONSTRAINT taskStateRecord_id_user_fkey FOREIGN KEY
(user_completed_id) REFERENCES User(id) ON UPDATE CASCADE;

ALTER TABLE ONLY TaskStateRecord
     ADD CONSTRAINT taskStateRecord_id_task_fkey FOREIGN KEY
(task_id) REFERENCES Task(id) ON UPDATE CASCADE;

ALTER TABLE ONLY SprintStateRecord
     ADD CONSTRAINT SprintStateRecord_id_sprint_fkey FOREIGN KEY
(sprint_id) REFERENCES Sprint(id) ON UPDATE CASCADE;

ALTER TABLE ONLY Project_categories
     ADD CONSTRAINT project_categories_id_project_fkey FOREIGN KEY
(project_id) REFERENCES Project(id) ON UPDATE CASCADE ON DELETE
CASCADE;

ALTER TABLE ONLY Project_categories
     ADD CONSTRAINT project_categories_id_category_fkey FOREIGN
KEY (category_id) REFERENCES Category(id) ON UPDATE CASCADE ON
DELETE CASCADE;
```

Link para o script SQL

Revision History

1. Changing of ENUM in Project and project_members to boolean
2. Removal of tables project_tasks, sprint_tasks, task_comments, thread_comments, image and contains_image
3. Removal of constraint on the attribute name in the User table
4. Removal of Access and Role Domain
5. Removal of the Effort Domain and changing the attribute effort in Sprint and Task to an integer
6. Addition of attribute sprint_id to the Task table
7. Addition of attribute task_id and thread_id to the Task table
8. Addition of attribute ReportTypes and the respective ENUM in the Domain, addition of attributes comment_reported_id and user_reported_id and the renaming of the attribute user_id to author_id
9. Changing of table of functional dependencies in project_members to fix the error about the keys
10. Update the SQL code

Grupo 1717, 18/3/2018

Ana Margarida Oliveira Pinheiro da Silva, up201505505@fe.up.pt
Luís Miguel Cardoso Lopes Correia, up201503342@fe.up.pt
Pedro Daniel dos Santos Reis, up201506046@fe.up.pt
Vicente Fernandes Ramada Caldeira Espinha, up201503764@fe.up.pt