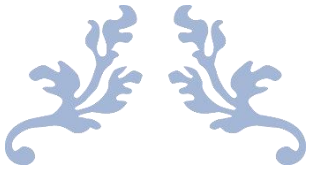


УПРАВЛЕНИЕ НА ПРОЕКТИ

База от данни и Системи за управление на бази от данни (практикуми)

January, 2025

ИЗГГОТВИЛ: ПЕТЯ ЛИЧЕВА
Факултетен номер: 3MI0700022

Съдържание

Обхват на модела. Дефиниране на задачата

Базата от данни за управление на проекти ще съхранява информацията за задачите, свързани с даден проект, както и за техните изпълнители.
Базта от данни ще съхранява информация за различни компании. Компанията ще се определя еднозначно от име на компанията, адрес, описание на дейността на компанията. Във всяка компания ще има различни екипи (минимум 1 и максимум 10).
Всеки екип ще се определя еднозначно от име на екипа, описание на дейността на екипа. Във всеки екип ще има минимум 1 служител (който ще е наследник на множеството от същности на потребителите) и всеки екип ще работи по различни проекти, а по всеки проект ще могат да работят много екипи.
Всеки потребител на системата се определя еднозначно от уникален номер, имейл, потребителско име (до 32 символа) и парола (до 32 символа). Потребителлите ще са два вида – служители и любители.
Всеки служител се определя еднозначно от телефонен номер, адрес, заплата, екип, в който работят.
Всеки любител се определя еднозначно от описание на дейността, с която се занимават любителски.
Всеки проект се определя еднозначно от име, версия, описание, дата на пускане в употреба. Проектът ще се състои от много задачи като една задача ще може да участва в точно един проект.
Всяка задача се определя еднозначно от уникален номер, начална дата, статус (завършена или блокирана) и текстово описание. За някои задачи е дефиниран и краен срок (който не може да е преди датата на създаване на задачата). Задължително се съхранява информация за това кой потребител и на коя дата е дефинирал задачата. Бъговете са специални задачи, за които задължително се съхранява сценарий за възпроизвеждането им, както и номер на версията на приложението, в която са били забелязани за първи път.
Задсачите биват – изпълними задачи или бъгове.
Всяка изпълнима задача се определя еднозначно от педполагаема крайна дата.
Всеки бъг се определя еднозначно от сценарий за възпроизвеждането им, номер на версията на приложението, в което са били забелязани за първи път.
Всяка блокирана задача се определя еднозначно от дата на блокиране, блокираща задача.
Един потребител може да работи по няколко задачи, а по една задача могат да работят няколко потребителя едновременно. Дадена задача може да бъде блокирана от други задачи (тоест необходимо е те да бъдат завършени, за да може да се работи по нея) и една задача може да блокира няколко други задачи.

Множества от същности и техните атрибури

Компании – име на компанията, адрес на компанията, описание на дейността на компанията

Екипи – име на компанията, име на екипа, описание на дейността на екипа
Потребители – уникален номер, имейл, потребителско име и парола
Служители – телефонен номер, адрес, заплата, екип, в който работят
Любители – описание на дейността, с която се занимават любителски
Проекти – име на провкта, верисия на проекта, описание, дата на пускане в употреба
Задачи – уникален номер на задачата, начална дата, статус, описание
Изпълними задачи – уникален номер на задачата , педполагаема крайна дата
Бъгове – уникален номер на задачата , сценарий за възпроизвеждането им, номер на версията на приложението, в което са били забелязани за първи път

Домейн на атрибутите

Компании – име на компанията: низ, адрес: низ, описание на дейността на компанията: низ
Екипи – име на компанията: низ, име на екипа: низ, описание на дейността на екипа: низ
Потребители – уникален номер: цяло положително число, потребителско име: низ, имейл: низ и парола: низ
Служители – уникален номер: цяло положително число, телефонен номер: низ, адрес: низ, заплата: число с плаваща запетая
Любители – уникален номер: цяло положително число, описание на дейността, с която се занимават любителски: низ
Проекти – име: низ, версия: низ, описание: низ, дата на пускане в употреба: дата
Задачи - уникален номер: число, начална дата: дата, описание: низ
Изпълними задачи - педполагаема крайна дата: дата
Бъгове – сценарий за възпроизвеждането им: низ

Връзки

Потребители – Служители: Множеството от потребители е корен на йерархия на наследственост, а множеството от служители е негов наследник (isa връзка)
Потребители – Любители: Множеството от потребители е корен на йерархия на наследственост, а множеството от любители е негов наследник (isa връзка)
Потребители – Задачи: Един потребител може да дефинира много задачи, но една задача може да е дефинирана от точно един потребител.
Компании – Екипи: Една компания може да има между 1 и 10 екипа, а един екип може да участва в точно една компания (не се допуска екип без компания, за която да работи)
Екипи – Служители: В един екип може да има много служители, но не може да няма нито един служител. Един служител може да работи в точно един екип.
Проекти – Задачи: Един проект може да се състои от 1 или от много задачи, а една задача, може да е част само от един проект.
Екипи – Проекти: Един екип може да работи по много проекти и по един проект могат да работят много екипи.
Любители – Задачи: Един любител може да работи по много задачи и по една задача могат да работят много любители.

Задачи – Изпълними задачи: Множеството от задачи е корен на йерархия на наследственост, а множеството от изпълними задачи е негов наследник (isa връзка)
Задачи – Бъгове: Множеството от задачи е корен на йерархия на наследственост, а множеството от бъгове е негов наследник (isa връзка)
Бъгове – Проекти: Един бъг може да се забележи за пръв път в точно една версия на даден проект, но в една версия на проект може да се забележат много бъгове.

Ключове

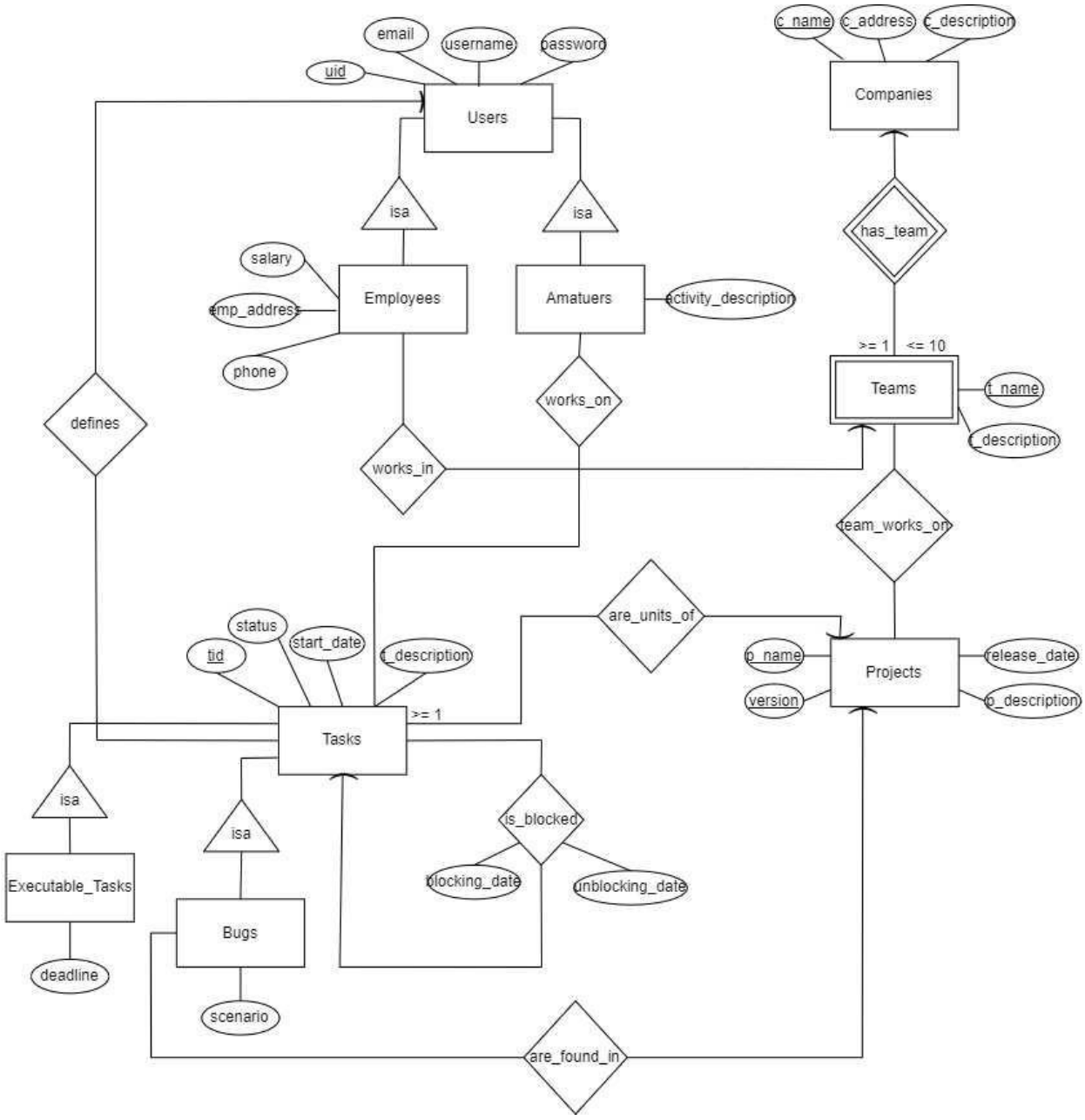
Компании – име на компанията: низ, адрес на компанията: низ уникално определят компанията.
Екипи – име на компанията: низ, име на екипа: низ уникално определят екипа.
Потребители – уникален номер: цяло число уникално определя потребителя.
Служители – уникален номер: цяло число уникално определя служителя.
Любители – уникален номер: цяло число уникално определя любителя.
Проекти – име на провкта: низ, верисия на проекта: низ уникално определят проекта.
Задачи – уникален номер на задачата: цяло число уникално определя задачата.
Изпълними задачи – уникален номер на задачата: цяло число уникално определя изпълнимата задача.
Бъгове – уникален номер на задачата: цяло число уникално определя бъга.

Правила и проверки

Компании – име на компанията (до 20 символа, не може да е NULL), адрес на компанията (до 100 символа, адресът да е записан в български стандарт за адреси – компаниите ще са само на територията на страната, не може да е NULL), описание на дейността на компанията (до 450 символа, може да е NULL)
Екипи – име на екипа (до 20 символа, не може да е NULL), описание на дейността на екипа (до 300 символа, може да е NULL)
Потребители – уникален номер (последователно, уникално, цяло число, не може да е NULL), потребителско име (до 32 символа – да съдържа букви и цифри и да започва с буква, не може да е NULL), имейл (валиден имейл адрес до 100 символа, не може да е NULL) и парола (до 32 символа – да съдържа букви, цифри и специални символи като _-.!#@, не може да е NULL)
Служители –телефонен номер (валиден телефонен номер до 20 символа, не може да е NULL), адрес (до 100 символа, адресът да е записан в български стандарт адреси – служителите ще са само на територията на страната, може да е NULL), заплата (положително число с плаваща запетая, може да е NULL)
Любители – описание на дейността, с която се занимават любителски (до 100 символа, може да е NULL)
Проекти – име на провкта (до 30 символа, не може да е NULL), верисия на проекта (до 20 символа, не може да е NULL), описание (до 300 символа, може да е NULL), дата на пускане в употреба (датата на пускане на проекта да не е преди 01.01.2023 година, не може да е NULL)

Задачи – уникален номер на задачата (поредно, уникално, цяло число, не може да е NULL), начална дата (датата да не е преди 01.01.2023 година, може да е NULL), статус (завършена или блокирана, може да е NULL в случаите, когато все още се изпълнява), описание (до 300 символа, може да е NULL), статус (завършена/незавършена, може да е NULL)
Изпълними задачи – педполагаема крайна дата (предполагаемата крайна дата да е поне месец след началната дата на задачата, може да е NULL)
Бъгове – сценарий за възпроизвеждането им (до 600 символа, не може да е NULL), номер на версията на приложението, в което са били забелязани за първи път (до 20 символа, не може да е NULL).

E/R модел на данни



Релационен модел на днни

Таблици

ISA йерархия Потребители

Релационен подход:

Users (uid: int (> 0), email: string (320), username: string (10), password: string (64))
Employees (uid: int (> 0), salary: double (> 0.0), emp_address: string (100), phone: string (>= 10 && <= 13)) Amateurs (uid: int (> 0), activity_description: string (350))

ООП подход:

Users (uid: int (> 0), email: string (320), username: string (10), password: string (64))

PK: uid (unique, not null)

Employees (uid: int (> 0), salary: double (> 0.0), emp_address: string (100), phone: string (>= 10 && <= 13)) – таблицата отпада

Amateurs (uid: int (> 0), activity_description: string (100)) – таблицата отпада

Users_Employees (uid: int (> 0), email: string (320), username: string (10), password: string (64), salary: double (> 0.0), emp_address: string (100), phone: string (>= 10 && <= 13), team_id: int (> 0))

PK: uid (unique, not null)

FK: Users (team_id) -> Teams (tid)

Users_Amateurs (uid: int (> 0), email: string (320), username: string (10), password: string (64), activity_description: string (350))

PK: uid (unique, not null)

NULL подход:

Users (uid: int (> 0), email: string (320), username: string (10), password: string (64), team_id: int (> 0), salary: double (> 0.0), emp_address: string (100), phone: string (>= 10 && <= 13), activity_description: string (350))

Избирам ООП подхода за йерархията на потребителите със съкращаване на две от таблиците (Employees и Amateurs), защото те са нефункционални и не носят достатъчно информация за въпросните роли на потребителя. Не избирам релационния подход или NULL подхода, защото при бъдещи разширения на систа те биха били лош вариант поради излишество (повторения) на данни. С оглед на възможните транзакции към базата този подход би бил най-удачен, защото в повечето случаи ще се обръщаме към само една от таблиците в модела. За потребителите ще генерираме 3 таблици.

ISA йерархия Задачи

Релационен подход:

Tasks (tid: int (> 0), status: string (10 – unfinished), start_date: date (>= 01.01.2023), tdescription: string (350), uid: int (> 0), project_id: int (> 0), btid: int (> 0))

PK: tid (unique, not null)

FK: Tasks (uid) -> Tasks (uid)

FK: Tasks (project_id) -> Projects (pid)

FK: Tasks (btid) -> Tasks (tid)

Executable_Tasks (tid: int (> 0), deadline: date (>= 01.01.2023))

PK: tid (unique, not null)

Bugs_Tasks (tid: int (> 0), scenario: string (600), project_id: int (> 0), blocking_date: date (>= 01.01.2023), unblocking_date: date (>= blocking_date))

PK: tid (unique, not null)

FK: Bugs_Tasks (project_id) -> Projects (pid)

ООП подход:

Tasks (tid: int (> 0), status: string (10 – unfinished), start_date: date (>= 01.01.2023), tdescription: string (350))

Executables (tid: int (> 0), deadline: date (>= 01.01.2023))

Bugs (tid: int (> 0), scenario: string (600), project_id: int (> 0))

Executable_Tasks (tid: int (> 0), status: string (10 – unfinished), start_date: date (>= 01.01.2023), tdescription: string (350), deadline: date (>= start_date))

Bugs_Tasks (tid: int (> 0), status: string (10 – unfinished), start_date: date (>= 01.01.2023), tdescription: string (350), scenario: string (600), project_id: int (> 0), blocking_date: date (>= 01.01.2023), unblocking_date: date (>= blocking_date))

NULL подход:

Tasks (tid: int (> 0), status: string (10 – unfinished), start_date: date (>= 01.01.2023), tdescription: string (350), deadline: date (>= start_date), scenario: string (600), project_id: int (> 0), blocking_date: date (>= 01.01.2023), unblocking_date: date (>= blocking_date))

Избирам релационния подход, защото за удобство бих могла да си позволя да разделя атрибутите, необходими за описанието на различни типове задачи, в три таблици. С оглед на възможните транзакции към базата, обаче, този подход не би бил най-удачен, защото в част от случаи ще се обръщаме към повече от една таблица в модела, за да извлечем цялостно информацията, от която се интересуваме. Но от гледна точка на сравнително малкото количество данни в таблиците, това не би било от особено значение за бързодействието на модела. За задачите ще генерираме 3 таблици.

Таблици на останалите множества от същности и взаимоотношенията им

Companies (cid: int (> 0), cname: string (50), caddress: string (100), cdescription: string (500))

PK: cid (unique, not null)

Забележка: Тъй като името на компанията ще е низ с променлива дължина, не би било удачно да е ключ, заради проблеми с индексацията на такива полета. Това налага въвеждането на ново поле за ключ, а именно cid: int (> 0).

Teams (team_id: int (> 0), cid: int (> 0), tname: string (50), tdescription: string (350))

PK: team_id (unique, not null)

FK: Teams (cid) -> Companies (cid)

Забележка: Тъй като имената на екипа и компанията ще са низове с променлива дължина, не би било удачно да сформират ключ, заради проблеми с индексацията 4 на такива полета. Това налага въвеждането на ново поле за ключ, а именно team_id: int (>= 0).

Projects (project_id: int (> 0), pname: string (50), version: string (20), release_date: date (>= 01.01.2023), pdescription: string (350))

PK: project_id (unique, not null)

Забележка: Тъй като името на проекта ще е низ с променлива дължина, не би било удачно да е ключ, заради проблеми с индексацията на такива полета. Това налага въвеждането на ново поле за ключ, а именно pid: int (>= 0).

Defines (uid: int (> 0), tid: int (> 0)) – таблицата отпада

AreFoundIn (tid: int (> 0), project_id: int (> 0)) – таблицата отпада

AreUnitsOf (tid: int (> 0), project_id: int (> 0)) – таблицата отпада

IsBlocked (blocking_task_id: int (> 0), blocked_task_id: int (> 0), blocking_date: date (>= 01.01.2023), unblocking_date: date (>= blocking_date)) – таблицата отпада WorksIn (uid: int (> 0), team_id: int (> 0)) – таблицата отпада

WorksOn (uid: int (> 0), tid: int (> 0))

PK: uid (unique, not null), tid (unique, not null)

FK: WorksOn (uid) -> Users (uid)

WorksOn (tid) -> Tasks (tid)

TeamsWorkOn (team_id: int (> 0), project_id: int (> 0))

PK: team_id (not null), project_id (unique, not null)

FK: TeamsWorkOn (team_id) -> Teams (team_id) TeamsWorkOn (project_id) -> Projects (project_id)

Забележка: дължината на полетата от тип string са проверявани от различни онлайн източници по спецификациите на SQL с цел максимална прецизност и доближаване до изискванията на задачата.

Извод

След всички преобразувания до момента стигаме до следните релации.

Таблици

Users (uid: int (>= 0), email: string (320), username: string (10), password: string (64))
Users_Employees (uid: int (> 0), email: string (320), username: string (10), password: string (64), salary: double (> 0.0), emp_address: string (100), phone: string (>= 10 && <= 13), team_id: int (> 0))

Users_Amateurs (uid: int (> 0), email: string (320), username: string (10), password: string (64), activity_description: string (350))

Tasks (tid: int (> 0), status: string (10 – unfinished), start_date: date (>= 01.01.2023), tdescription: string (350), uid: int (> 0), project_id: int (> 0), btid: int (> 0)) Executables (tid: int (> 0), deadline: date (>= 01.01.2023))

Bugs (tid: int (> 0), scenario: string (600), project_id: int (> 0), blocking_date: date (>= 01.01.2023), unblocking_date: date (>= blocking_date))

Companies (cid: int (> 0), cname: string (50), caddress: string (100), cdescription: string (500))

Teams (team_id: int (> 0), cid: int (> 0), tname: string (50), tdescription: string (350))
Projects (project_id: int (> 0), pname: string (50), version: string (20), release_date: date (>= 01.01.2023), pdescription: string (350))

WorksOn (uid: int (> 0), tid: int (> 0))

TeamsWorkOn (team_id: int (> 0), project_id: int (> 0))

Първични ключове (PK)

PK: Users -> uid (unique, not null)

PK: Users_Employees -> uid (unique, not null)

PK: Users_Amateurs -> uid (unique, not null)

PK: Tasks -> tid (unique, not null)

PK: Executable_Tasks -> tid (unique, not null)

PK: Bugs -> tid (unique, not null)

PK: Companies -> cid (unique, not null)

PK: Teams -> tid (unique, not null)

PK: Projects -> pid (unique, not null)

PK: WorksOn -> uid (unique, not null), tid (unique, not null)

PK: TeamsWorkOn -> team_id (not null), project_id (unique, not null)

Външни ключове (FK)

FK: Users (tname, cname) -> Teams (tname, cname)

FK: Tasks (uid) -> Tasks (uid)

FK: Tasks (pname, pversion) -> Projects (pname, version)

FK: Tasks (btid) -> Tasks (tid)

FK: Bugs (found_in_pname, found_in_pversion) -> Projects (pname, version)

FK: Teams (cname) -> Companies (cname)

FK: WorksOn (uid) -> Users (uid)

FK: WorksOn (tid) -> Tasks (tid)

FK: TeamsWorkOn (team_id) -> Teams (team_id)

FK: TeamsWorkOn (project_id) -> Projects (project_id)

Ограничения

Users (email - _@_.)

Users (username - _%)

Users (password - _%)

Users_Employees (email - _@_.)

Users_Employees (username - _%)

Users_Employees (password - _%)

Users_Employees (salary - > 0.0)

Users_Amateurs (email - _@_.)

Users_Amateurs (username - _%)

Users_Amateurs (password - _%)

Tasks (status – finished / unfinished)

Tasks (start_date >= 01.01.2023)

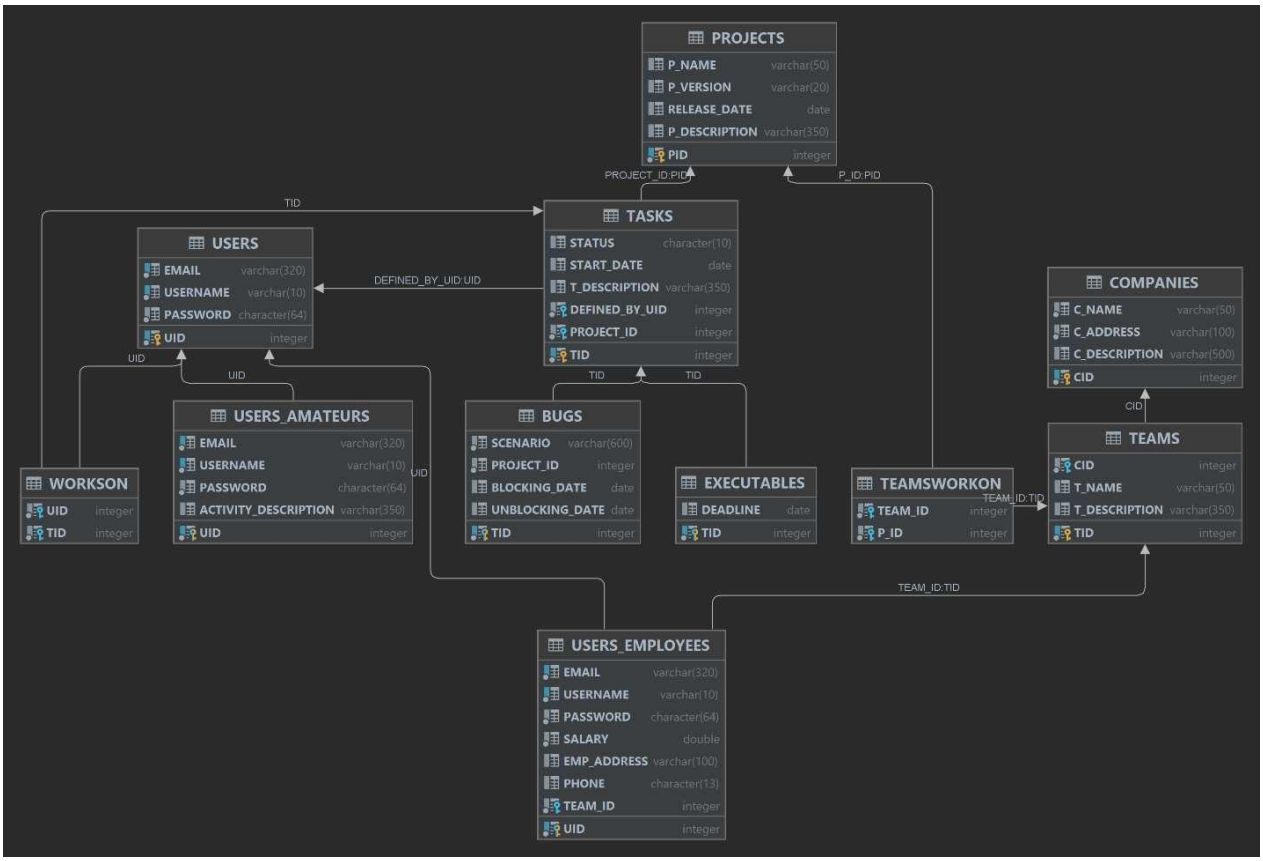
Executable_Tasks (deadline >= start_date)

Bugs (blocking_date >= 01.01.2023)

Bugs (unblocking_date >= blocking_date)

Projects (release_date - >= 01.01.2023)

Схема на базат от данни



Функции

Първа функция:

```sql
SET SCHEMA FN3MI0700022;

-- A function that returns all bugs in a project full data as a table.
-- This function can be used to make an audit of the bugs in a definite
-- project by its ID.
CREATE OR REPLACE FUNCTION F_GET_ALL_BUGS_IN_PROJECT(PID INT)
    RETURNS TABLE
            (
                TID             INT,
                SCENARIO        VARCHAR(600),
                PROJECT_ID      INT,
                BLOCKING_DATE   DATE,
                UNBLOCKING_DATE DATE
            )
    LANGUAGE SQL
    RETURN SELECT B.TID, B.SCENARIO, B.PROJECT_ID, B.BLOCKING_DATE, B.UNBLOCKING_DATE
            FROM FN3MI0700022.BUGS B
            WHERE B.PROJECT_ID = PID;

-- execution of the function
GRANT EXECUTE ON FUNCTION FN3MI0700022.F_GET_ALL_BUGS_IN_PROJECT TO PUBLIC;
SELECT * FROM TABLE(FN3MI0700022.F_GET_ALL_BUGS_IN_PROJECT(10));
```

- Първата функция, която съм имплементирала, връща пълна информация за бъговете в даден проект като таблица.

Втора функция:

```
SET SCHEMA FN3MI0700022;

-- A function that returns all teams in a definite company tasks. This function can be
-- used for audit of all tasks that a definite company works on.
CREATE OR REPLACE FUNCTION F_GET_ALL_TEAMS_IN_A_COMPANY_TASKS(P_CID INT)
    RETURNS TABLE
            (
                TEAM_ID          INT,
                TEAM_NAME        VARCHAR(50),
                TASK_STATUS      CHAR(10),
                TASK_START_DATE  DATE,
                TASK_DESCRIPTION VARCHAR(350),
                DEFINED_BY_USER  VARCHAR(10),
                PROJECT_NAME     VARCHAR(50),
                PROJECT_VERSION  VARCHAR(20)
            )
    RETURN
        SELECT T.TID,
               T.T_NAME,
               TSK.STATUS,
               TSK.START_DATE,
               TSK.T_DESCRIPTION,
               U.USERNAME,
               P.P_NAME,
               P.P_VERSION
        FROM FN3MI0700022.TEAMS T
                 LEFT JOIN FN3MI0700022.TEAMSWORKON TW
                           1<->0..n: ON T.TID = TW.TEAM_ID
                 LEFT JOIN FN3MI0700022.PROJECTS P
                           1..n<->1: ON TW.P_ID = P.PID
                 LEFT JOIN FN3MI0700022.TASKS TSK
                           1<->0..n: ON TSK.PROJECT_ID = P.PID
                 LEFT JOIN FN3MI0700022.USERS U
                           1..n<->1: ON TSK.DEFINED_BY_UID = U.UID
        WHERE TW.P_ID = P_CID;

-- execution of the function
GRANT EXECUTE ON FUNCTION FN3MI0700022.F_GET_ALL_TEAMS_IN_A_COMPANY_TASKS TO PUBLIC;
SELECT * FROM TABLE(FN3MI0700022.F_GET_ALL_TEAMS_IN_A_COMPANY_TASKS(10));
```

- Втората функция, която съм имплментирала, връща пълна информация за задачите, по които работят екипите в дадена компания като таблица.

Тригери

Първи тригер:

```
SET SCHEMA FN3MI0700022;

-- A trigger that validates the user data before insert.
CREATE OR REPLACE TRIGGER TRIG_BEFORE_USER_INSERT
    BEFORE INSERT ON FN3MI0700022.USERS
    REFERENCING NEW AS N
    FOR EACH ROW
BEGIN
    DECLARE V_IS_VALID_DATA BOOLEAN;

    -- Call the validation procedure
    CALL FN3MI0700022.USER_MOD.P_VALIDATE_USER_DATA( U_EMAIL N.EMAIL, U_USERNAME N.USERNAME,
                                                     V_IS_VALID V_IS_VALID_DATA);

    -- Signal an error if data is invalid
    IF V_IS_VALID_DATA = FALSE THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Validation failed: Email or Username is not unique.';
    END IF;
END;

-- trigger execution - signal error that the username or email is not unique
INSERT INTO FN3MI0700022.USERS(EMAIL, USERNAME, PASSWORD)
VALUES ( EMAIL 'dummyaddress@outlook.com', USERNAME 'speedy423', PASSWORD '123456');

-- trigger execution - inserts successfully a user
INSERT INTO FN3MI0700022.USERS(EMAIL, USERNAME, PASSWORD)
VALUES ( EMAIL 'dummy@outlook.com', USERNAME 'new423', PASSWORD '123456');
```

- Първият тригер, който съм имплементирала, валидира данните на потребител, който се опитва да се регистрира преди реално да го направи.

Втори тригер:

```
SET SCHEMA FN3MI0700022;

-- A trigger that validates the user date before update.
CREATE OR REPLACE TRIGGER TRIG_BEFORE_USER_UPDATE
    BEFORE UPDATE ON FN3MI0700022.USERS
    REFERENCING OLD AS O NEW AS N
    FOR EACH ROW
BEGIN
    DECLARE V_IS_VALID_DATA BOOLEAN;

    SET V_IS_VALID_DATA = O.USERNAME != N.USERNAME OR O.EMAIL != N.EMAIL;

    -- Signal an error if data is invalid
    IF V_IS_VALID_DATA = FALSE THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Validation failed: Email or Username is not unique.';
    END IF;

    -- Optionally, check for specific fields being updated
    IF O.EMAIL = N.EMAIL AND O.USERNAME = N.USERNAME AND O.PASSWORD = N.PASSWORD THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No changes detected in the update operation.';
    END IF;
END;

-- trigger execution - successfully updates the user with a definite username
UPDATE FN3MI0700022.USERS U SET U.EMAIL = 'newaddress@gmail.com' WHERE U.USERNAME = 'new423';
```

- Вторият тригер, който съм имплементирала, валидира данните на потребител, който се опитва да си актуализира данните преди реално да настъпи прмяната.

Трети тригер:

```sql
SET SCHEMA FN3MI0700022;

-- A trigger that logs deletions from Projects table
-- (after a definite project is deleted).
CREATE TABLE DELETED_PROJECTS_LOG (
    PID INTEGER GENERATED ALWAYS AS IDENTITY
    CONSTRAINT PK_DEL_PROJECTS
    PRIMARY KEY,
    P_NAME VARCHAR(50),
    P_VERSION VARCHAR(20),
    RELEASE_DATE DATE DEFAULT CURRENT_DATE,
    P_DESCRIPTION VARCHAR(350),
    CHECK (P_VERSION LIKE '%_'),
    CHECK (RELEASE_DATE >= DATE('2023-01-01'))
);

SET SCHEMA FN3MI0700022;

TRIG_LOG_AFTER_PROJECT_DELETION...;

-- deletes all connections to the project before executing the next line,
-- then executes the project deletion line and after the deletion in the
-- FN3MI0700022.DELETED_PROJECTS_LOG table will appear a log for this
-- deleted project
DELETE FROM FN3MI0700022.PROJECTS P WHERE P.PID = 1;
```

- Третият тригер, който съм имплементирала, записва текущо изтрит проект в таблицата DELETED_PROJECTS_LOG с цел одит на вече изтрити проекти.

Изгледи

Employees view

```
CREATE VIEW "Employees" AS
SELECT FN3MI0700022.USERS_EMPLOYEES.EMAIL,
       FN3MI0700022.USERS_EMPLOYEES.USERNAME,
       FN3MI0700022.USERS_EMPLOYEES.EMP_ADDRESS,
       FN3MI0700022.USERS_EMPLOYEES.PHONE,
       FN3MI0700022.USERS_EMPLOYEES.SALARY,
       FN3MI0700022.TEAMS.T_NAME,
       FN3MI0700022.COMPANIES.C_NAME
FROM FN3MI0700022.USERS_EMPLOYEES, FN3MI0700022.TEAMS, FN3MI0700022.COMPANIES
WHERE FN3MI0700022.USERS_EMPLOYEES.TEAM_ID = FN3MI0700022.TEAMS.TID AND
      FN3MI0700022.TEAMS.CID = FN3MI0700022.COMPANIES.CID;

SELECT * FROM "Employees" WHERE C_NAME = 'Nexus Dynamics';
```

Tasks view

```
CREATE VIEW "Tasks" AS
SELECT * FROM FN3MI0700022.EXECUTABLES
WHERE DEADLINE > CURRENT_DATE;

SELECT * FROM "Tasks";
DROP VIEW "Tasks";
```

Bugs view

```
CREATE VIEW "Bugs" AS
SELECT * FROM "Bugs"
WHERE FN3MI0700022."Bugs".UNBLOCKING_DATE > FN3MI0700022."Bugs".BLOCKING_DATE;

SELECT * FROM Bugs;
INSERT INTO Bugs(SCENARIO, PROJECT_ID, BLOCKING_DATE, UNBLOCKING_DATE)
VALUES('new bug scenario', 10, '2023-04-22', '2023-05-31');
SELECT * FROM Bugs;
```

Процедури

Първа процедура (с прихващане на грешки):

```
SET SCHEMA FN3MI0700022;

CREATE OR REPLACE MODULE USER_MOD;

-- A procedure that validates user data. This procedure can be used in the triggers
-- that will validate the user data before insert and before update.
ALTER MODULE USER_MOD PUBLISH PROCEDURE P_VALIDATE_USER_DATA(IN U_EMAIL VARCHAR(320),
                                                             IN U_USERNAME VARCHAR(10),
                                                             OUT V_IS_VALID BOOLEAN);
ALTER MODULE USER_MOD ADD PROCEDURE P_VALIDATE_USER_DATA(IN U_EMAIL VARCHAR(320),
                                                         IN U_USERNAME VARCHAR(10),
                                                         OUT V_IS_VALID BOOLEAN)
BEGIN
    DECLARE V_UID INT;
    DECLARE V_IS_VALID BOOLEAN;
    SET V_UID = (SELECT U.UID FROM FN3MI0700022.USERS U WHERE U.USERNAME = U_USERNAME OR U.EMAIL = U_EMAIL);

    IF V_UID IS NOT NULL
        THEN
            SET V_IS_VALID = FALSE;
            SIGNAL SQLSTATE '70001'
            SET MESSAGE_TEXT = 'Username or email is not unique!';
    ELSE
        SET V_IS_VALID = TRUE;
    END IF;
END;

-- this procedure is tested in the before insert trigger for the FN3MI0700022.USERS table
```

- Първата процедура, която съм имплементирала, валидира данните на даден потребител.

Втора процедура (с курсор и while цикъл):

```
SET SCHEMA FN3MI0700022;
CREATE OR REPLACE MODULE COMPANIES_MOD;

-- A procedure that will return information about all companies in which amateurs
-- and experienced employees work, the count of amateurs and the count of experienced
-- employees. This information can be used for statistical researches.
ALTER MODULE COMPANIES_MOD PUBLISH PROCEDURE P_GET_ALL_COMPANIES_IN_WHICH_AMATEURS_WORKS_INFO();
ALTER MODULE COMPANIES_MOD ADD PROCEDURE P_GET_ALL_COMPANIES_IN_WHICH_AMATEURS_WORKS_INFO()
BEGIN
    DECLARE V_C_ID INT;
    DECLARE V_C_NAME VARCHAR(50);
    DECLARE V_C_ADDRESS VARCHAR(100);
    DECLARE V_C_DESCRIPTION VARCHAR(500);
    DECLARE V_AMATEURS_EMPLOYEES_CNT INT;
    DECLARE V_EXPERIENCED_EMPLOYEES_CNT INT;

    DECLARE SQLSTATE CHAR(5) DEFAULT '00000';

    DECLARE CURSOR_COMPANIES CURSOR FOR (
        SELECT C.*,
               AMATEURS.AMATEURS_EMPLOYEES_COUNT,
               EXPERIENCED.EXPERIENCED_EMPLOYEES_COUNT
        FROM FN3MI0700022.COMPANIES C
                 LEFT JOIN FN3MI0700022.TEAMS T 1<->0..n: ON C.CID = T.CID
                 LEFT JOIN (
            SELECT T2.TID, COUNT(*) AS AMATEURS_EMPLOYEES_COUNT
            FROM FN3MI0700022.USERS_AMATEURS UA
                     LEFT JOIN FN3MI0700022.WORKSON ET ON UA.UID = ET.UID
                     LEFT JOIN FN3MI0700022.TEAMS T2 ON ET.TID = T2.TID
            GROUP BY T2.TID
        ) AS AMATEURS ON T.TID = AMATEURS.TID
                 LEFT JOIN (
            SELECT UE.TEAM_ID, COUNT(*) AS EXPERIENCED_EMPLOYEES_COUNT
            FROM FN3MI0700022.USERS_EMPLOYEES UE
            GROUP BY UE.TEAM_ID
        ) AS EXPERIENCED ON T.TID = EXPERIENCED.TEAM_ID
    );

    CALL DBMS_OUTPUT.PUT_LINE('CID, C_NAME, C_ADDRESS, C_DESCRIPTION, AMATEURS_EMPLOYEES_COUNT, ' ||
                              'EXPERIENCED_EMPLOYEES_COUNT');
```

```
    OPEN CURSOR_COMPANIES;
    FETCH CURSOR_COMPANIES INTO V_C_ID, V_C_NAME, V_C_ADDRESS, V_C_DESCRIPTION,
        V_AMATEURS_EMPLOYEES_CNT, V_EXPERIENCED_EMPLOYEES_CNT;
    WHILE SQLSTATE = '00000'
        DO
            CALL DBMS_OUTPUT.PUT_LINE(V_C_ID || ', ' || V_C_NAME || ', ' || V_C_ADDRESS || ', '
                || V_C_DESCRIPTION || ', ' || V_AMATEURS_EMPLOYEES_CNT || ', ' || V_EXPERIENCED_EMPLOYEES_CNT);
            FETCH CURSOR_COMPANIES INTO V_C_ID, V_C_NAME, V_C_ADDRESS, V_C_DESCRIPTION,
                V_AMATEURS_EMPLOYEES_CNT, V_EXPERIENCED_EMPLOYEES_CNT;
        END WHILE;
    CLOSE CURSOR_COMPANIES;
END;

-- executes the procedure
CALL FN3MI0700022.COMPANIES_MOD.P_GET_ALL_COMPANIES_IN_WHICH_AMATEURS_WORKS_INFO();
```

- Втората процедура, която съм имплементирала, връща таблица, в която се съдържа информация за всички компании, в които работят както начинаещи, така и служители със дългогодишен опит, както и броят на едната група и броят на другата група служители.

Трета процедура (с курсор, входни и изходни данни):

```
SET SCHEMA FN3MI0700022;

-- A procedure that returns the history of a definite employee on the base of his / her
-- username. This procedure can be used for employees audit.
ALTER MODULE USER_MOD PUBLISH PROCEDURE P_GET_EMPLOYEE_HISTORY_BY_EMPLOYEE_USERNAME(
    IN P_UNM VARCHAR(10), OUT P_CURSOR_EMPLOYEE_HISTORY CURSOR);
ALTER MODULE USER_MOD ADD PROCEDURE P_GET_EMPLOYEE_HISTORY_BY_EMPLOYEE_USERNAME(
    IN P_UNM VARCHAR(10), OUT P_CURSOR_EMPLOYEE_HISTORY CURSOR)
BEGIN
    DECLARE P_CURSOR_EMPLOYEE_HISTORY CURSOR WITH RETURN FOR
        SELECT * FROM FN3MI0700022.USERS_EMPLOYEES E WHERE E.USERNAME = P_UNM;

    OPEN P_CURSOR_EMPLOYEE_HISTORY;
END;
```

- Третата процедура, която съм имплементирала, връща таблица, съдържаща информация за историята на даден служител по неговия псевдоним (уникално поле в таблиците USERS, USERS_AMATEURS и EMPLOYEES_USERS.

Приложение за достъп до базата от данни

main() method

- стартовата точка на приложението

```java
public static void main(String[] args) {
    ProjectManagementApp db2Obj = new ProjectManagementApp();
    String option = "";
    db2Obj.execute(db2Obj.printMenu(option));
}
```

printMenu() method

– принтира менюто и въвежда потребителя в базата данни като му дава опции, от които да си избере каква заявка иска да изпълни върху таблицата Employees

```
public String printMenu(String option) {
    Scanner input = new Scanner(System.in);
    do {
        System.out.println("""
                Welcome to Project Management DataBase!
                Choose a command to be executed!
                MENU:
                1. SELECT
                2. INSERT
                3. DELETE
                4. EXIT
                """);
        option = input.next();
    } while (!option.equals("SELECT") && !option.equals("INSERT")
            && !option.equals("UPDATE") && !option.equals("DELETE")
            && !option.equals("EXIT"));

    return option;
}
```

execute()

– извиква и изпълнява конкретен метод, който се грижи за даден тип заявки

```java
public void execute(String option) {
    String statement = ""; this.openConnection();
    switch (option) {
        case "SELECT": {
            statement = option + " * FROM FN3MI0700022.USERS";
            this.select(statement,  column: 4);
            break;
        } case "INSERT": {
            Scanner input = new Scanner(System.in);
            String mail = input.next();
            String username = input.next();
            String pass = input.next();

            statement = option + " INTO FN3MI0700022.USERS (EMAIL, USERNAME, PASSWORD)"
                    + " VALUES ('" + mail + "','" +
                    username + "','" + pass + "')";
            this.insert(statement);
        } case "DELETE": {
            Scanner input = new Scanner(System.in);
            String username = input.next();
            statement = option + " FROM FN3MI0700022.USERS WHERE FN3MI0700022.USERS.USERNAME = '"
                    + username + "'";
            this.delete(statement);
        } case "EXIT": default: System.exit( status: 0);
    } this.closeConnection();
}
```

Конзола с резултатните множества от заявките

SELECT

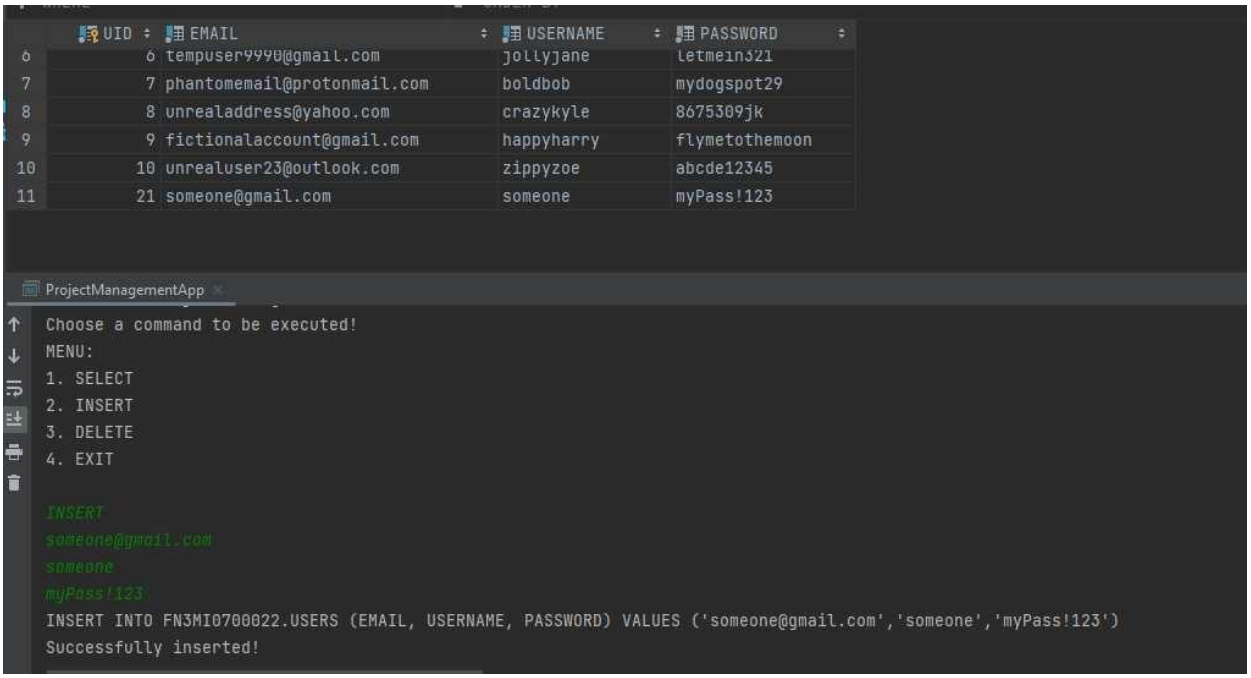
– показва всички записи в таблизата Employees с всичките им атрибути и съответните им стойности

```
Welcome to Project Management DataBase!
Choose a command to be executed!
MENU:
1. SELECT
2. INSERT
3. DELETE
4. EXIT

SELECT
1           fakeemail1@yahoo.com            coolguy56           abcd1234
2           wtpfakeemail@gmail.com          hotsauce            qwertyuiop
3           notreal@email.co.uk         lilypad44           password123
4           dummyaddress@outlook.com            speedy423           ilovemom1
5           fictitiousemail@yahoo.com           cloudberry          123456789
```

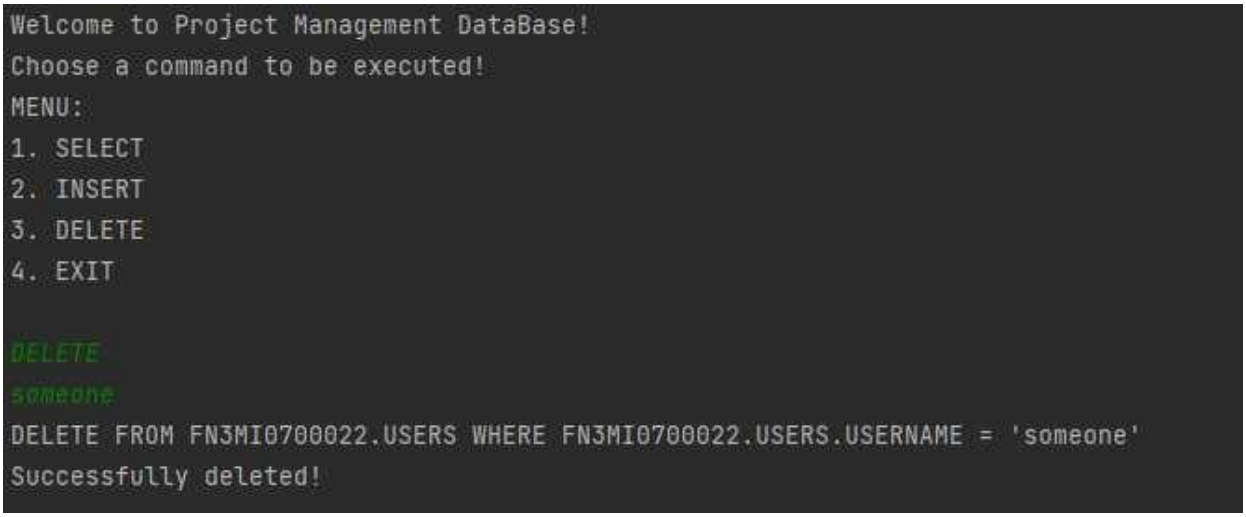
INSERT INTO

– добавя запис в таблицата Employees като потребителя трябва ръщно да въведе стойност в конзолата на приложението за всяко необходимо поле



DELETE

– изтрива запис за служител от таблицата Employees по подадено име на служителя, ако такъв съществува



EXIT

– излиза от приложението и терминира работата на конзолата