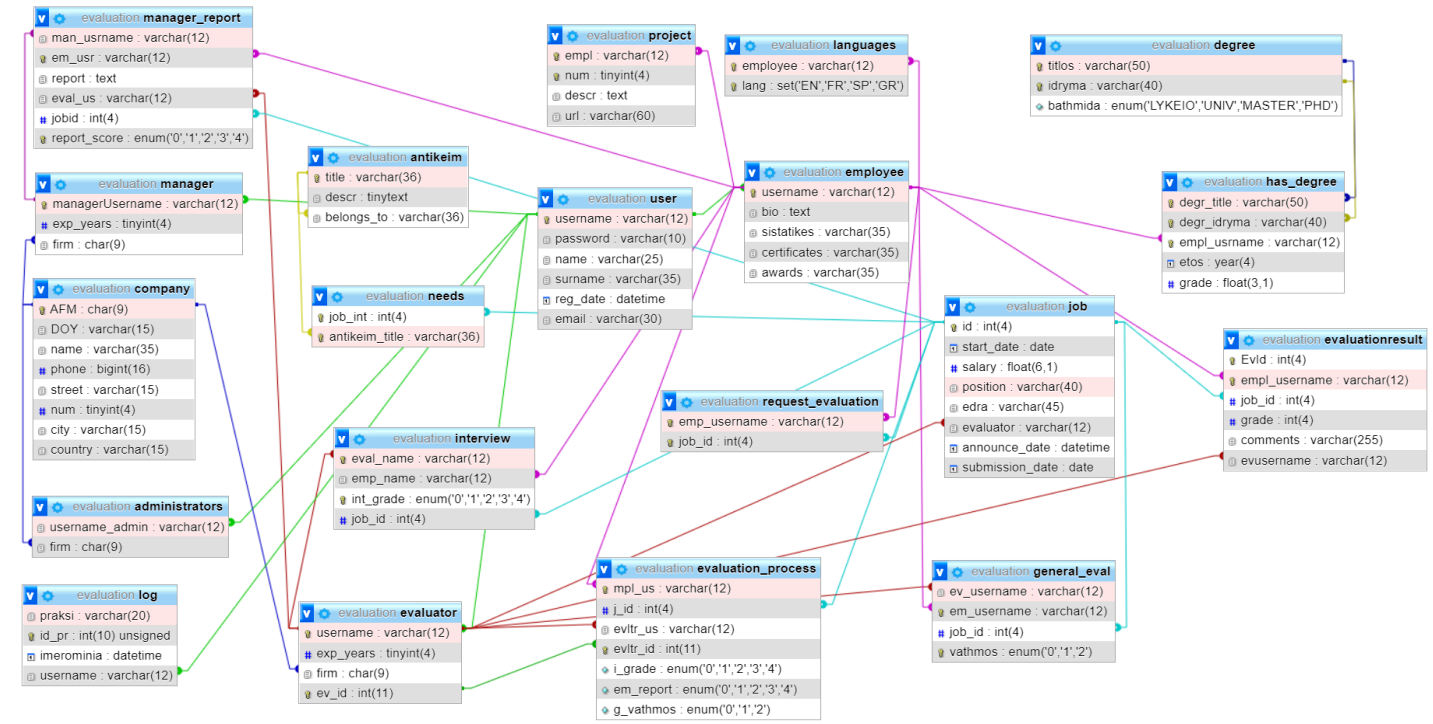
ΜΠΙΡΜΠΑΣ ΠΑΝΑΓΙΩΤΗΣ ΑΜ:1069365
ΡΟΔΗ ΑΣΗΜΙΝΑ ΑΜ:1059693

Α΄ ΜΕΡΟΣ PROJECT ΕΡΓΑΣΤΗΡΙΟΥ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

1)



Στην παραπάνω βάση υπάρχουν οι εξής παραδοχές :

- Οι βαθμοί στους πίνακες manager_report,general_eval και interview είναι ακέραιες τιμές από 0 έως 4.
- Στον πίνακα general_eval αποθηκεύεται η αξιολόγηση που υλοποιεί ο αξιολογητής με βάσει τα πτυχία, τις συστατικές και τον αριθμό των πρότζεκτ του εργαζομένου,
- Στον πίνακα evaluationresult στη στήλη comments αποθηκεύονται τα τα σχόλια που προέκυψαν συνολικά κατα τη διαδικασία της αξιολόγησης και όχι τα επιμέρους των αξιολογήσεων των πρώτων σταδίων.
- Στον πίνακα evaluation_process αποθηκεύνται οι βαθμοί των επιμέρους αξιολογήσεων με σκοπό την ευκολότερη χρήση του procedure που υπολογίζει τον τελικό βαθμό των εργαζομένων.

Για τον έλεγχο και τη δημιουργία της βάσης evaluation χρησιμοποιήσαμε το πρόγραμμα xampp.

2)

```
CREATE TABLE company (
AFM CHAR(9) DEFAULT 'unknown' NOT NULL,
DOY VARCHAR(15) DEFAULT 'unknown' NOT NULL,
name VARCHAR(35) DEFAULT 'unknown' NOT NULL,
```

```
phone BIGINT(16) NOT NULL,
street VARCHAR(15) DEFAULT 'unknown' NOT NULL,
num TINYINT(4) NOT NULL,
city VARCHAR(15) DEFAULT 'unknown' NOT NULL,
country VARCHAR(15) DEFAULT 'unknown' NOT NULL,
PRIMARY KEY(AFM)
);

CREATE TABLE user (
username VARCHAR(12) DEFAULT 'unknown' NOT NULL,
password VARCHAR(10) NOT NULL,
name VARCHAR(25) DEFAULT 'unknown' NOT NULL,
surname VARCHAR(35) DEFAULT 'unknown' NOT NULL,
reg_date DATETIME NOT NULL,
email VARCHAR(30) DEFAULT 'unknown' NOT NULL,
PRIMARY KEY(username)
);

CREATE TABLE manager (
managerUsername VARCHAR(12) DEFAULT 'unknown' NOT NULL,
exp_years TINYINT(4) NOT NULL,
firm CHAR(9) DEFAULT 'unknown' NOT NULL,
PRIMARY KEY(managerUsername),
CONSTRAINT a1
FOREIGN KEY(managerUsername) REFERENCES user(username)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT a2
FOREIGN KEY(firm) REFERENCES company(AFM)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE evaluator (
username VARCHAR(12) NOT NULL,
ev_id INT NOT NULL,
exp_years TINYINT(4) NOT NULL,
firm CHAR(9) DEFAULT 'unknown' NOT NULL,
PRIMARY KEY (username,ev_id),
CONSTRAINT EVALUATES
FOREIGN KEY(username) REFERENCES user(username)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT EVALUATES2
FOREIGN KEY(firm) REFERENCES company(AFM)
);

CREATE TABLE employee(
username VARCHAR(12) NOT NULL,
bio TEXT NOT NULL,
sistatikes VARCHAR(35) DEFAULT 'unknown',
```

```
certificates VARCHAR(35) DEFAULT 'unknown' ,
awards VARCHAR(35) DEFAULT 'unknown' ,
PRIMARY KEY(username),
CONSTRAINT h1
FOREIGN KEY (username) REFERENCES user(username)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE languages(
employee VARCHAR(12) NOT NULL,
lang SET('EN','FR','SP','GR') NOT NULL,
PRIMARY KEY (employee,lang),
CONSTRAINT e1
FOREIGN KEY (employee) REFERENCES employee(username)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE project(
empl VARCHAR(12) NOT NULL,
num TINYINT(4) NOT NULL AUTO_INCREMENT,
descr TEXT NOT NULL,
url VARCHAR(60) NOT NULL,
PRIMARY KEY(num, empl),
CONSTRAINT fk4
FOREIGN KEY (empl) REFERENCES employee(username)
ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE degree(
titlos VARCHAR(50) NOT NULL,
idryma VARCHAR(40) NOT NULL,
bathmida ENUM('LYKEIO','UNIV','MASTER','PHD'),
PRIMARY KEY(titlos,idryma)
);


CREATE TABLE has_degree (
degr_title VARCHAR(50) NOT NULL,
degr_idryma VARCHAR(40) NOT NULL,
empl_usrname VARCHAR(12) NOT NULL,
etos year(4) ,
grade float (3,1) NOT NULL,
PRIMARY KEY (degr_title, degr_idryma, empl_usrname),
CONSTRAINT d1
FOREIGN KEY(degr_title,degr_idryma) REFERENCES degree(titlos,idryma)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT d2
```

```
FOREIGN KEY(empl_usrname) REFERENCES employee(username)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE job (
id int(4) NOT NULL,
start_date DATE  NOT NULL,
salary float(6,1) NOT NULL,
position VARCHAR(40) NOT NULL,
edra VARCHAR(45) NOT NULL,
evaluator VARCHAR(12) NOT NULL,
announce_date DATETIME NOT NULL,
submission_date DATE NOT NULL ,
PRIMARY KEY(id),
CONSTRAINT j1
FOREIGN KEY(evaluator) REFERENCES evaluator(username)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE request_evaluation(
emp_username VARCHAR (12) NOT NULL,
job_id INT(4) NOT NULL,
CONSTRAINT k1
PRIMARY KEY(emp_username,job_id),
FOREIGN KEY (emp_username) REFERENCES employee(username)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT k2
FOREIGN KEY (job_id) REFERENCES job(id)
ON DELETE CASCADE ON UPDATE CASCADE
);


CREATE TABLE antikeim(
title VARCHAR(36) NOT NULL,
descr TINYTEXT NOT NULL,
belongs_to VARCHAR(36) ,
PRIMARY KEY (title),
CONSTRAINT q1
FOREIGN KEY (belongs_to) REFERENCES antikeim(title)
ON DELETE CASCADE ON UPDATE CASCADE
);


CREATE TABLE needs (
job_int int(4) NOT NULL,
antikeim_title VARCHAR(36) DEFAULT 'unknown' NOT NULL,
PRIMARY KEY (job_int, antikeim_title),
CONSTRAINT b1
```

```
FOREIGN KEY(job_int) REFERENCES job(id)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT b2
FOREIGN KEY(antikeim_title) REFERENCES antikeim(title)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE interview(
eval_name VARCHAR(12) NOT NULL,
emp_name VARCHAR(12) NOT NULL,
int_grade ENUM('0','1','2','3','4') ,
job_id INT(4) NOT NULL,
PRIMARY KEY(eval_name,int_grade),
CONSTRAINT l1
FOREIGN KEY (eval_name) REFERENCES evaluator(username)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT l2
FOREIGN KEY (emp_name) REFERENCES employee(username)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT l3
FOREIGN KEY(job_id) REFERENCES job(id)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE general_eval(
ev_username VARCHAR(12) NOT NULL,
em_username VARCHAR(12) NOT NULL,
job_id INT(4) NOT NULL,
vathmos ENUM('0','1','2'),
PRIMARY KEY (vathmos, em_username),
CONSTRAINT o1
FOREIGN KEY (ev_username) REFERENCES evaluator(username)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT o2
FOREIGN KEY (em_username) REFERENCES employee(username)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT o3
FOREIGN KEY (job_id) REFERENCES job(id)
ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE manager_report(
man_usrname VARCHAR(12) NOT NULL,
em_usr VARCHAR(12)NOT NULL,
report TEXT,
eval_us VARCHAR(12),
jobid INT(4),
report_score ENUM('0','1','2','3','4'),
PRIMARY KEY(em_usr,report_score),
```

```
CONSTRAINT p1
FOREIGN KEY (man_usrname) REFERENCES manager(managerUsername)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT p2
FOREIGN KEY (em_usr) REFERENCES employee(username)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT p3
FOREIGN KEY (eval_us) REFERENCES evaluator(username)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT p4
FOREIGN KEY (jobid) REFERENCES job(id)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE evaluationresult(
EvId int NOT NULL,
empl_username VARCHAR(12) NOT NULL,
job_id int(4) NOT NULL,
grade int(4) NOT NULL,
comments VARCHAR(255) DEFAULT 'unknown' ,
evusername VARCHAR(12) NOT NULL,
PRIMARY KEY(evid,empl_username),
CONSTRAINT c1
FOREIGN KEY(empl_username) REFERENCES employee(username)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT c2
FOREIGN KEY(job_id) REFERENCES job(id)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT c3
FOREIGN KEY(evusername) REFERENCES evaluator(username)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE log (
praksi SET('eisagogi','diagrafi','enimerosi') ,
id_pr INT UNSIGNED NOT NULL AUTO_INCREMENT,
imerominia DATETIME,
username VARCHAR (12) NOT NULL,
PRIMARY KEY (id_pr),
CONSTRAINT w1
FOREIGN KEY (username) REFERENCES user(username)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE evaluation_process(
mpl_us VARCHAR(12) NOT NULL,
j_id INT(4) NOT NULL,
evltr_us VARCHAR(12) NOT NULL,
```

```
evltr_id INT NOT NULL,
i_grade ENUM('0','1','2','3','4'),
em_report ENUM('0','1','2','3','4'),
g_vathmos ENUM('0','1','2'),
PRIMARY KEY(mpl_us,evltr_id),
CONSTRAINT z1
FOREIGN KEY (mpl_us) REFERENCES employee(username)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT z2
FOREIGN KEY (j_id) REFERENCES job(id)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT z3
FOREIGN KEY (evltr_us,evltr_id) REFERENCES evaluator(username,ev_id)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

3.1)

```
DELIMITER $
CREATE PROCEDURE aithsh(IN u_name VARCHAR(25),IN u_surname VARCHAR(25))
BEGIN
DECLARE ev_name VARCHAR(12);
DECLARE ev_surname VARCHAR(12);
DECLARE ait_username VARCHAR(12);
DECLARE finidhedFlag1 INT;
DECLARE finishedFlag2 INT;
DECLARE grade INT;
SELECT username INTO ait_username
FROM user
WHERE u_name=name AND u_surname=surname;
SELECT job_id,grade FROM requestevaluation
WHERE empl_usname=ait_username;
IF (evaluationresult.grade IS NULL)
THEN
SELECT ‘Evaluation in progress’;
SELECT grade,comments,EvId FROM evaluationresult
WHERE empl_username=ait_username;
DECLARE obi_wan_kenobi CURSOR FOR
SELECT evusername FROM evaluationresult  WHERE ait_username=empl_usname;
DECLARE CONTINUE FOR NOT FOUND SET finishedFlag=1;
OPEN obi_wan_kenobi;
SET finishedFlag=0;
FETCH obi_wan_kenobi INTO new.evusername;
WHILE (finishedFlag1=0) DO
SELECT new.evaluator AS ’Aksiologitis’;
FETCH obi-wan_kenobi new.evusername;
```

```
BEGIN
DECLARE kylo_ren CURSOR FOR
SELECT name ,surname FROM user WHERE new.evusername=username;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedFlag2=1;
OPEN kylo_ren;
SET finishedFlag2=0;
FETCH kylo_ren INTO ev_name,ev_surname;
WHILE (finishedFlag2=0) DO
SELECT ev_name AS ‘Onoma Aksiologiti’,ev_surname AS ‘Eponimo Aksiologiti’;
FETCH kylo_ren INTO ev_name,ev_surname;
END WHILE;
CLOSE kylo_ren;
END;
END WHILE;
CLOSE obi_wan_kenobi;
END;
ELSE
SELECT grade,comments,EvId FROM evaluationresult
WHERE ait_username=empl_username;
BEGIN
DECLARE obi_wan_kenobi CURSOR FOR
SELECT evusername FROM evaluationresult
WHERE ait_username=empl_username;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedFlag1=1;
OPEN obi_wan_kenobi;
SET finishedFlag1=0;
FETCH obi_wan_kenobi INTO new.evaluator;
WHILE (finisheFlag1=0) DO
SELECT new.evusername AS ‘Aksiologitis’;
FETCH obi_wan_kenobi INTO new.evusername;
BEGIN
DECLARE kylo_ren CURSOR FOR
SELECT name,surname FROM user
WHERE new.evusername=username;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedFlaf2=1;
OPEN kylo_ren;
SET finishedFlag2=0;
FETCH kylo_ren INTO ev_name,ev_surname;
WHILE (finishedFlag2=0) DO
SELECT ev_name AS ‘Onoma Aksiologiti’,ev_surname AS ‘Eponimo Aksiologiti’;
FETCH kylo_ren INTO ev_name,ev_surname;
END WHILE;
CLOSE kylo_ren;
END;
END WHILE;
CLOSE obi_wan_kenobi;
END;
END$
```

DELIMITER;

Η παραπάνω procedure παίρνει ως είσοδο το όνομα και το επώνυμο του εργαζομένου, εμφανίζει τις αιτήσεις του μέσω του πίνακα requestevaluation. Στην συνέχεια αντιστοιχεί με το username του εργαζομένου τον βαθμό του στον πίνακα evaluationresults. Αν ο βαθμός είναι κενός τότε εμφανίζει το μήνυμα ‘Evaluation in progress’ και μέσω του obi_wan_kenobi φορτώνει τα usernames των αξιολογητών και μετά με τον kylo_ren τα ονόματα και τα επώνυμα τους. Διαφορετικά, εμφανίζει τον βαθμό και τα σχόλια και πάλι μέσω των cursors αρχικά τα usernames και μετά το όνομα και το επώνυμο των αξιολογητών.
Η παραπάνω stored procedure εμφάνιζε σφάλματα κατα την προσπάθειά μας να την υλοποιήσουμε στο xampp.

3.2)

```
DELIMITER $
CREATE PROCEDURE telikos_bathmos(IN jb_ID INT(4), IN evl_ID INT(4))
BEGIN
DECLARE grade1 INT;
DECLARE grade2 INT;
DECLARE grade3 INT;
DECLARE sum INT;
SELECT i_grade,em_report,g_vathmos
INTO grade1,grade2,grade3
FROM evaluation_process
WHERE j_id=jb_ID AND evltr_id=evl_ID;
SELECT grade INTO f_grade FROM evaluationresult;
SET sum=grade1+grade2+grade3;
IF (grade1 IS NULL OR grade2 IS NULL OR grade3 IS NULL )
THEN
SELECT ‘This evaluation is not finished.’;
ELSE
INSERT INTO evaluationresult(grade) values (@sum);
END IF;
END $
DELIMITER ;
```

Η παραπάνω procedure δέχεται ως είσοδο τον κωδικό μιας θέσης και τον κωδικό του αξιολογητή και με τη χρήση του βοηθητικού πίνακα evaluation_process εισάγει στις αντίστοιχες μεταβλητές τους βαθμούς των επιμέρους αξιολογήσεων και τους αθροίζει, αποθηκεύει το αποτέλεσμα σε μια μεταβλητή sum και την αποθηκεύει στο αντίστοιχο grade του πίνακα evaluationresult και την εμφανίζει με το κατάλληλο μήνυμα. Αν κάποια από τις τιμές των βαθμών είναι NULL τότε εμφανίζει ‘This evaluation is not finished.’.
Για την λειτουργία του παραπάνω procedure κρίναμε απαραίτητη την δημιουργία triggers για την εισαγωγή των νέων βαθμών στον πίνακα evaluation_process.

```
DELIMITER $
CREATE TRIGGER eisagogi_igrd
AFTER INSERT ON interview
FOR EACH ROW
BEGIN
```

```
INSERT INTO evaluation_process(i_grade) SELECT int_grade FROM interview;
END $
DELIMITER ;

DELIMITER $
CREATE TRIGGER eisagogi_gengrd
AFTER INSERT ON general_eval
FOR EACH ROW
BEGIN
INSERT INTO evaluation_proces(g_vathmos) SELECT vathmos FROM general_eval;
END $
DELIMITER ;


DELIMITER $
CREATE TRIGGER eisagogi_rgrd
AFTER INSERT ON manager_report
FOR EACH ROW
BEGIN
INSERT INTO evaluation_process(em_report) SELECT report_score FROM manager_report;
END $
DELIMITER ;
```

3.3)

```
DELIMITER $

CREATE PROCEDURE eleghos_aksiologisis(IN idjob INT(4))
BEGIN
SET @evaluations=(SELECT COUNT(*) FROM evaluationresult);
SELECT @evaluations WHERE job_id=idjob;
SET @requests=(SELECT COUNT(*) FROM request_evaluation);
SELECT @requests WHERE job_id=idjob;
IF (@evaluations=@requests)
THEN
SELECT empl_username,grade FROM evaluationresult AS Teliki_Aksiologisi WHERE job_id=idjob
ORDER BY grade DESC;
ELSEIF (@evaluations < @requests)
THEN
SELECT empl_username,grade FROM evaluationresult WHERE job_id=idjob
ORDER BY grade DESC;
SET @evaluations_in_proc=@requests-@evaluations;
SELECT ‘Apomenoun:’, @evaluations_in_proc, ‘aksiologiseis.’;
ELSE
SELECT ‘Den iparxoun ipopsifioi.’;
END IF;
END$
DELIMITER ;
```

Η παραπάνω procedure έχει ως είσοδο το id μιας θέσης και συγκρίνει το πλήθος των αιτήσεων μέσω του πίνακα request_evaluation (αποθηκευμένο στη μεταβλητή requests) και των αξιολογήσεων μέσω του πίνακα evaluationresult (αποθηκευμένο στη μεταβλητή eavaluations). Αν το πλήθος είναι ίδιο εμφανίζει τον πίνακα Teliki_Aksiologisi τα grades με φθίνουσα σειρά. Αν οι αιτήσεις είναι περισσότερες τότε εμαφανίζει το πλήθος των αξιολογήσεων που απομένουν και αν δεν υπάρχουν υποψήφιοι τότε εμφανίζει 'Den iparxoun ipopsifioi.'.

Για την λειτουργία της procedure δημιουργήσαμε 2 triggers όπου αυξάνονται οι μεταβλητές requests και evaluations με την προσθήκη νέων εγγραφών στους αντίστοιχους πίνακες.

```
DELIMITER $
CREATE TRIGGER  request_counter
AFTER INSERT ON request_evaluation
FOR EACH ROW
BEGIN
SET @requests =@requests +1;
END$
DELIMITER ;



DELIMITER $
CREATE TRIGGER  evaluation_counter
AFTER INSERT ON evaluationresult
FOR EACH ROW
BEGIN
SET @evaluations =@evaluations +1;
END$
DELIMITER ;
```

4.1)

```
DELIMITER $
CREATE TRIGGER job_eisagogi
AFTER INSERT ON job
FOR EACH ROW
BEGIN
INSERT INTO log (praksi,id_pr,imerominia,username) VALUES('eisagogi',NEW.id_pr,NOW(),NEW.username);
END$

DELIMITER ;

DELIMITER $
CREATE TRIGGER job_enimerosi
AFTER UPDATE ON job
FOR EACH ROW
BEGIN
INSERT INTO log (praksi,id_pr,imerominia,username) VALUES('enimerosi',NEW.id_pr,NOW(),NEW.username);
END$

DELIMITER ;
```

```
DELIMITER $
CREATE TRIGGER job_diagrafi
AFTER DELETE ON job
FOR EACH ROW
BEGIN
INSERT INTO log (praksi,id_pr,imerominia,username) VALUES('diagrafi',OLD.id_pr,NOW(),OLD.username);
END$

DELIMITER ;

DELIMITER $
CREATE TRIGGER employee_eisagogi
AFTER INSERT ON employee
FOR EACH ROW
BEGIN
INSERT INTO log (praksi,id_pr,imerominia,username) VALUES('eisagogi',NEW.id_pr,NOW(),NEW.username);
END$

DELIMITER ;

DELIMITER $
CREATE TRIGGER employee_enimerosi
AFTER UPDATE ON employee
FOR EACH ROW
BEGIN
INSERT INTO log (praksi,id_pr,imerominia,username) VALUES('enimerosi',NEW.id_pr,NOW(),NEW.username);
END$

DELIMITER ;

DELIMITER $
CREATE TRIGGER employee_diagrafi
AFTER DELETE ON employee
FOR EACH ROW
BEGIN
INSERT INTO log (praksi,id_pr,imerominia,username) VALUES('diagrafi',OLD.id_pr,NOW(),OLD.username);
END$

DELIMITER ;

DELIMITER $
CREATE TRIGGER req_eval_eisagogi
AFTER INSERT ON requestevaluation
FOR EACH ROW
BEGIN
INSERT INTO log (praksi,id_pr,imerominia,username) VALUES('eisagogi',NEW.id_pr,NOW(),NEW.username);
END$
```

```
DELIMITER ;

DELIMITER $
CREATE TRIGGER req_eval_enimerosi
AFTER UPDATE ON requestevaluationi
FOR EACH ROW
BEGIN
INSERT INTO log (praksi,id_pr,imerominia,username) VALUES('enimerosi',NEW.id_pr,NOW(),NEW.username);
END$

DELIMITER ;

DELIMITER $
CREATE TRIGGER req_eval_diagrafi
AFTER DELETE ON requestevaluation
FOR EACH ROW
BEGIN
INSERT INTO log (praksi,id_pr,imerominia,username) VALUES('diagrafi',OLD.id_pr,NOW(),OLD.username);
END$

DELIMITER ;
```

Με τα παραπάνω triggers δημιουργούμε καινούργιες εγγραφές κάθε φορά που υπάρχει αλλαγή στους πίνακες request_evaluation, job και employee.

4.3)

```
DELIMITER $
CREATE TRIGGER epeksergasia_profil
BEFORE UPDATE ON manager
FOR EACH ROW
BEGIN
IF (manager.managerUsername <> administrators.username_admin)
THEN
SIGNAL SQLSTATE VALUE '45000'
SET MESSAGE_TEXT= 'Attempted change is not allowed due to authorization protocol.';
END IF;
END $
DELIMITER ;

DELIMITER $
CREATE TRIGGER epeksergasia_profil1
BEFORE UPDATE ON evaluator
FOR EACH ROW
BEGIN
IF (evaluator.evaluator.username <> administrators.username_admin)
THEN
```

```
SIGNAL SQLSTATE VALUE ‘45000’
SET MESSAGE_TEXT= ‘Attempted change is not allowed due to authorization protocol.’;
END IF;
END $
DELIMITER ;

DELIMITER $
CREATE TRIGGER epeksergasia_profil2
BEFORE UPDATE ON employee
FOR EACH ROW
BEGIN
IF (employee.username <> administrators.username_admin)
THEN
SIGNAL SQLSTATE VALUE ‘45000’
SET MESSAGE_TEXT= ‘Attempted change is not allowed due to authorization protocol.’;
END IF;
END $
DELIMITER ;

DELIMITER $
CREATE TRIGGER epeksergasia_profil3
BEFORE UPDATE ON user
FOR EACH ROW
BEGIN
IF (user.username <> administrators.username_admin)
THEN
SIGNAL SQLSTATE VALUE ‘45000’
SET MESSAGE_TEXT= ‘Attempted change is not allowed due to authorization protocol.’;
END IF;
END $
DELIMITER ;
```

Τα παραπάνω triggers ελέγχουν την ταυτότητα των χρηστών που επιχειρούν οποιαδήποτε αλλαγή σε αυτούς τους πίνακες πριν την εκτέλεση της πράξης και αν δεν είναι κάποιος administrator ακυρώνει την πράξη και εμφανίζει error 4500.