ΑΝΑΦΟΡΑ PROJECT ΣΤΟ ΕΡΓΑΣΤΗΡΙΟ ΒΑΣΕΩΝ ΔΕΔΟΜΈΝΩΝ 2020

ΜΕΛΗ ΟΜΑΔΑΣ

ΤΣΑΜΠΑΣ ΣΤΥΛΙΑΝΟΣ 1039884

ΣΙΑΜΟΓΛΟΥ ΧΑΡΑΛΑΜΠΟΣ 1041601

Περιγραφή

Στόχος της άσκησης ήταν η δημιουργεία μίας βάσης δεδομένων και της αντίστοιχης διεπαφής για την διαδικασία αξιολογήσεων προσλήψεων σε έναν όμηλο εταιριών.

Το σχήμα της βάσης μας βασίστηκε στο σχήμα που δινόταν σαν παράδειγμα στην εκφώνηση στο οποίο προστέθηκαν οι παρακάτω πίνακες.

admin : Καταγράφει τους χρήστες που έχουν την ιδιότητα του Administrator. Περιέχει το πεδίο username το οποίο κληρωνομεί από τον πίνακα user.
evaluationprocess : Καταγράφει την διαδικασία αξιολόγησης. Έχει ως κύριο κλειδί τον συνδιασμό των empl_username και job_id που προέρχονται από τους πίνακες employee και job αντίστοιχα. Περιέχει επίσης τα πεδία phase_1, phase_2 και phase_3 για τις βαθμολογίες στις τρεις φάσεις αξιολόγησης. το πεδίο comment για σχόλια σχετικά με τη συνέντευξη, το report για την αναφορά του manager καθώς και πεδίο mngr_username με το όνομα του manager που κατέθεσε το report.
evaluationresult : Καταγράφει τα αποτελέσμα των αξιολογήσεων. Περιέχει τα πεδία id και empl_username που αποτελούν τα πρωτεύοντα κλειδιά, πεδίο job_id ως δευτερεύων κλειδί από τον πίνακα job, grade για τον τελικό βαθμό καθώς και comments για τα σχόλια του evaluator από την συνέντευξη.
log : Καταγράφει τις ενέργειες που συμβαίνουν σε συγκεκριμένους πίνακες. Ως πρωτεύων κλειδί έχει ένα id. Τα υπόλοιπα πεδία του είναι username για το όνομα του χρήστη που έκανε την ενέργεια, action για το είδος της ενέργειας (insert/update), table για τον πίνακα στον οποίο έγινε η ενέργεια, result για τον αν η ενέργια ήταν επιτυχής ή όχι και datetime για την ημερομηνία και ώρα της ενέργειας.

Triggers

Περιγράφονται με σχόλια στον επισυναπτόμενο κώδικα.

Procudures

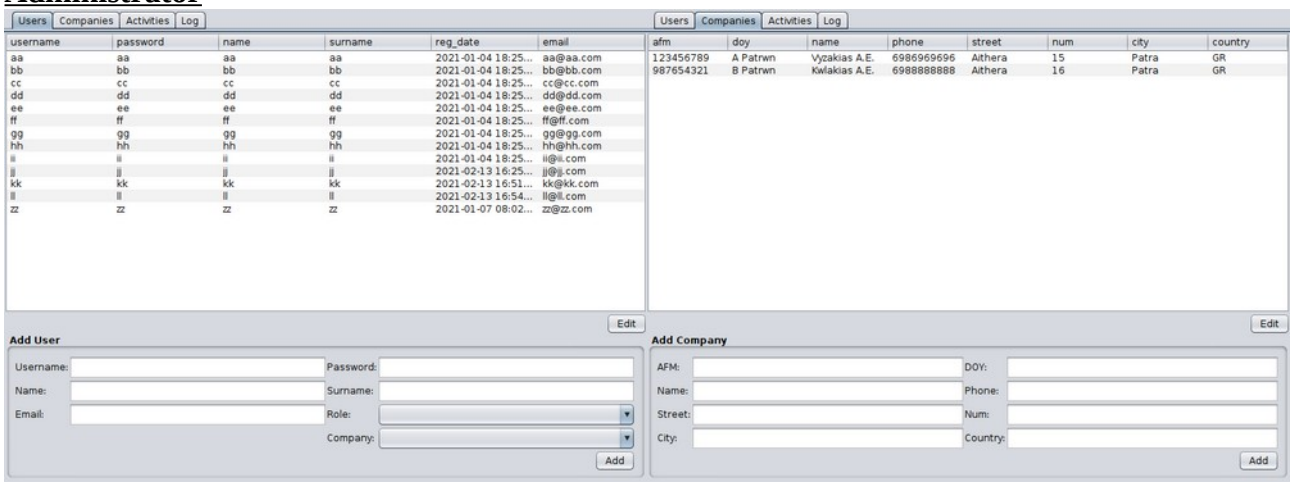
Περιγράφονται με σχόλια στον επισυναπτόμενο κώδικα.

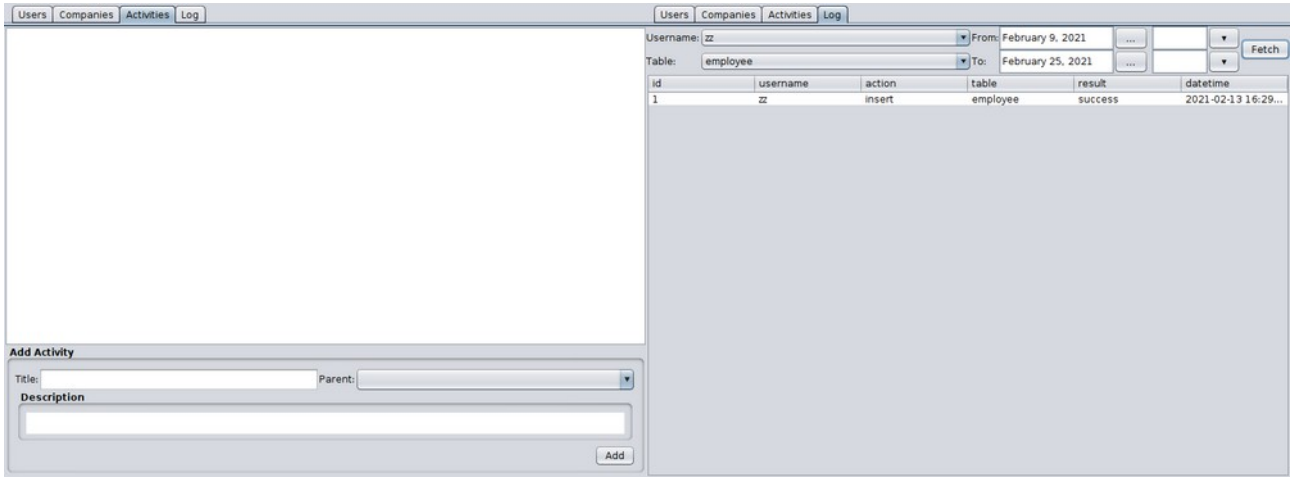
Διεπαφή Χρήστη

Το interface σε java δεν το ολοκληρώσαμε από άποψη λειτουργικότητας. Υλοποιήσαμε το login των χρηστών καθώς και την λήψη και εμφάνιση μερικών δεδομένων για τον administrator. Παρακάτω φαίνονται screenshots από το design της διεπαφής.

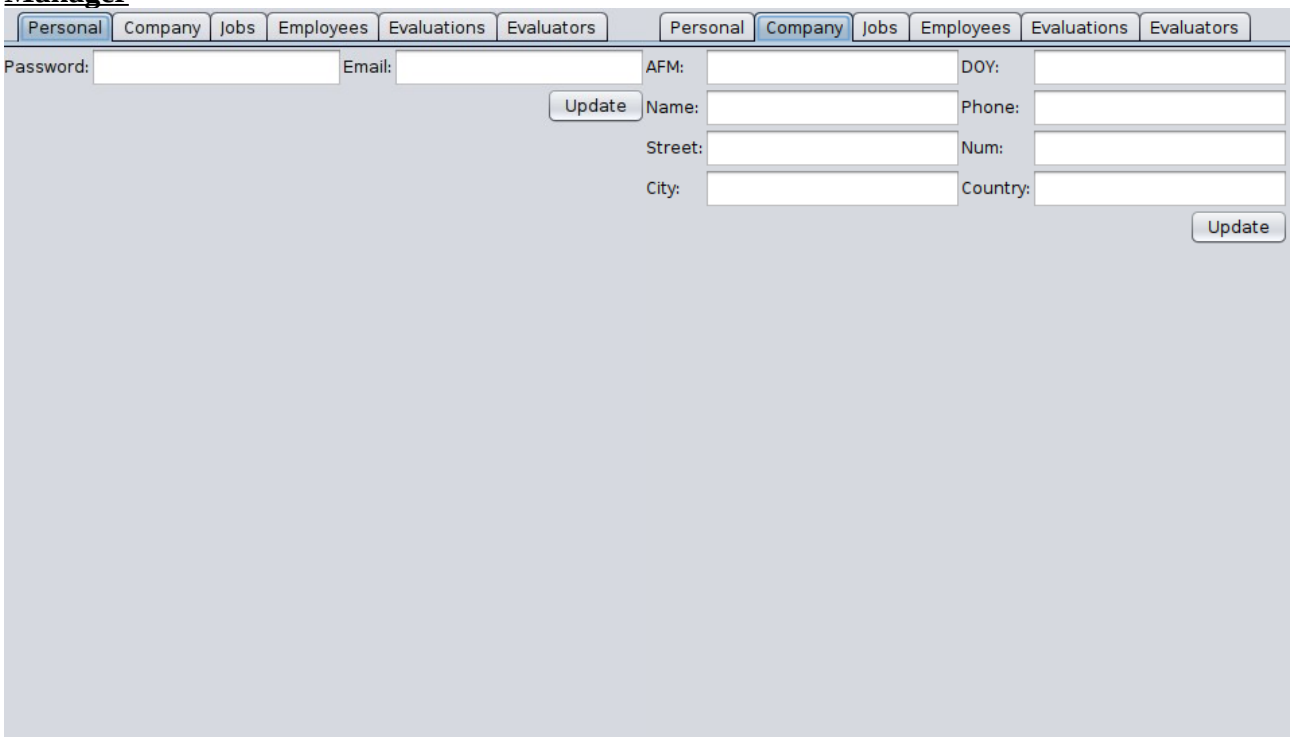
Login

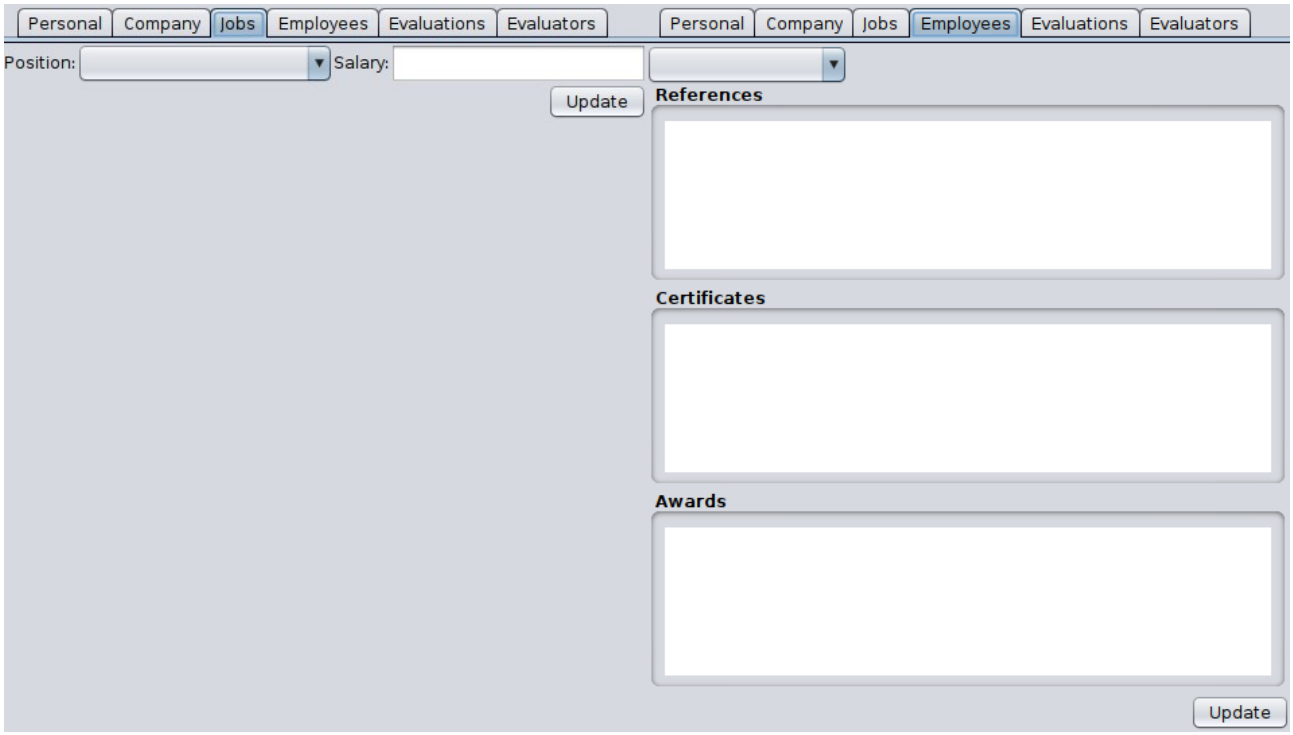
Administrator

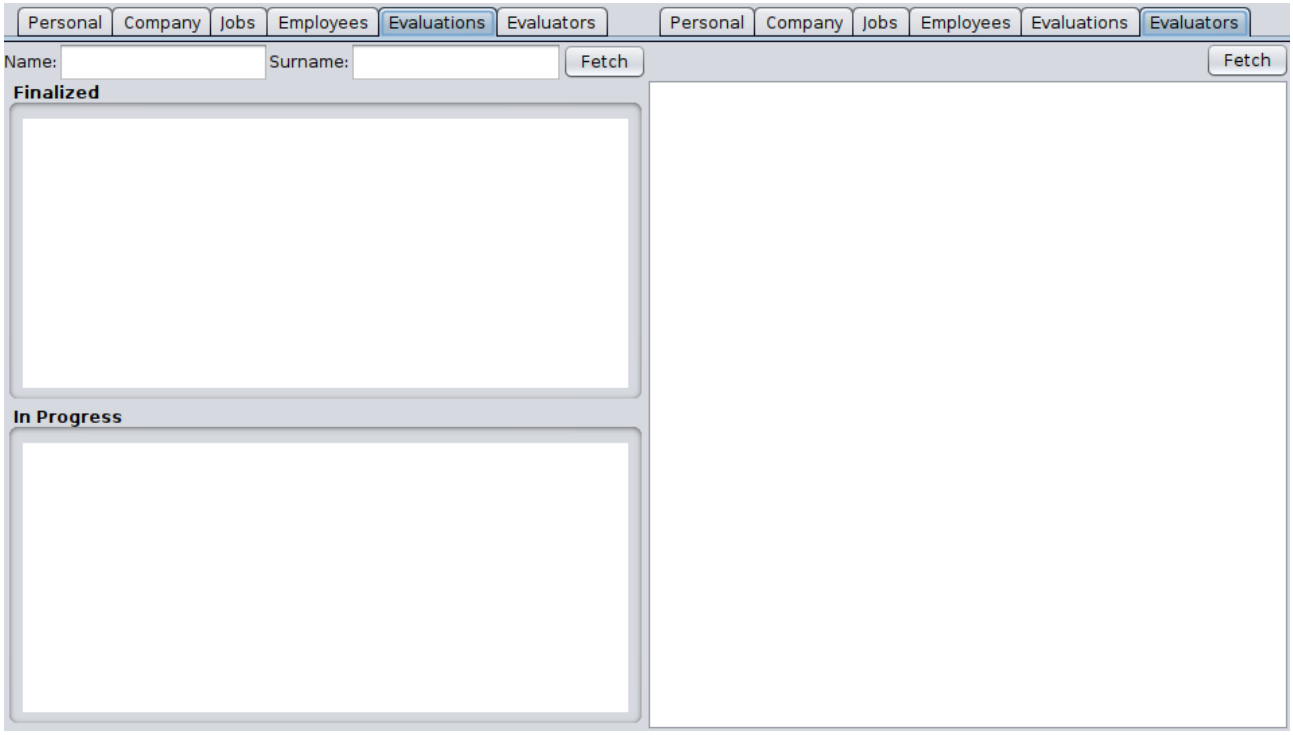




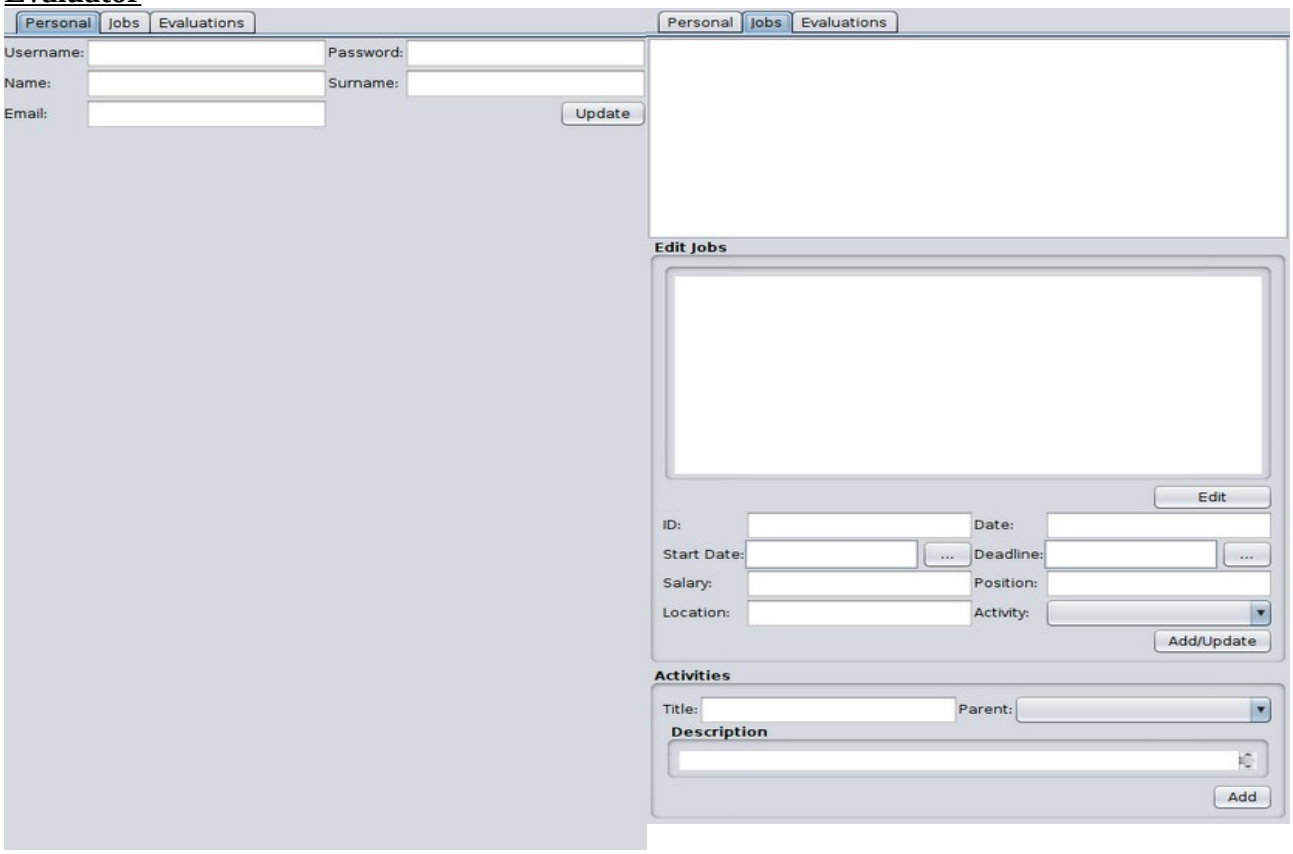
Manager

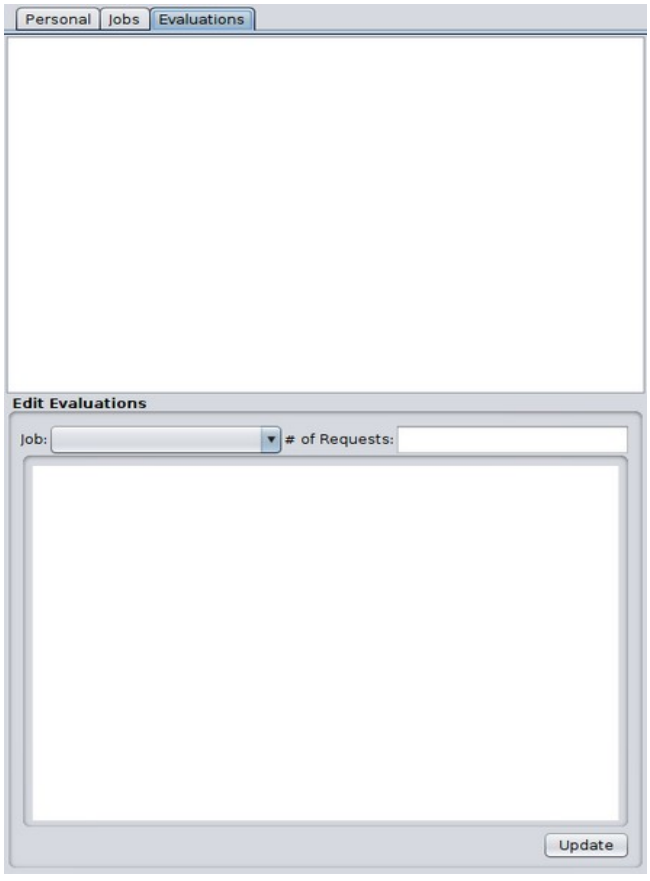

Personal
Company
Jobs
Employees
Evaluations
Evaluators
Password:
Email:
Update
AFM:
DOY:
Name:
Phone:
Street:
Num:
City:
Country:
Update


Personal
Company
Jobs
Employees
Evaluations
Evaluators
Position:
Salary:
Update
References
Certificates
Awards
Update

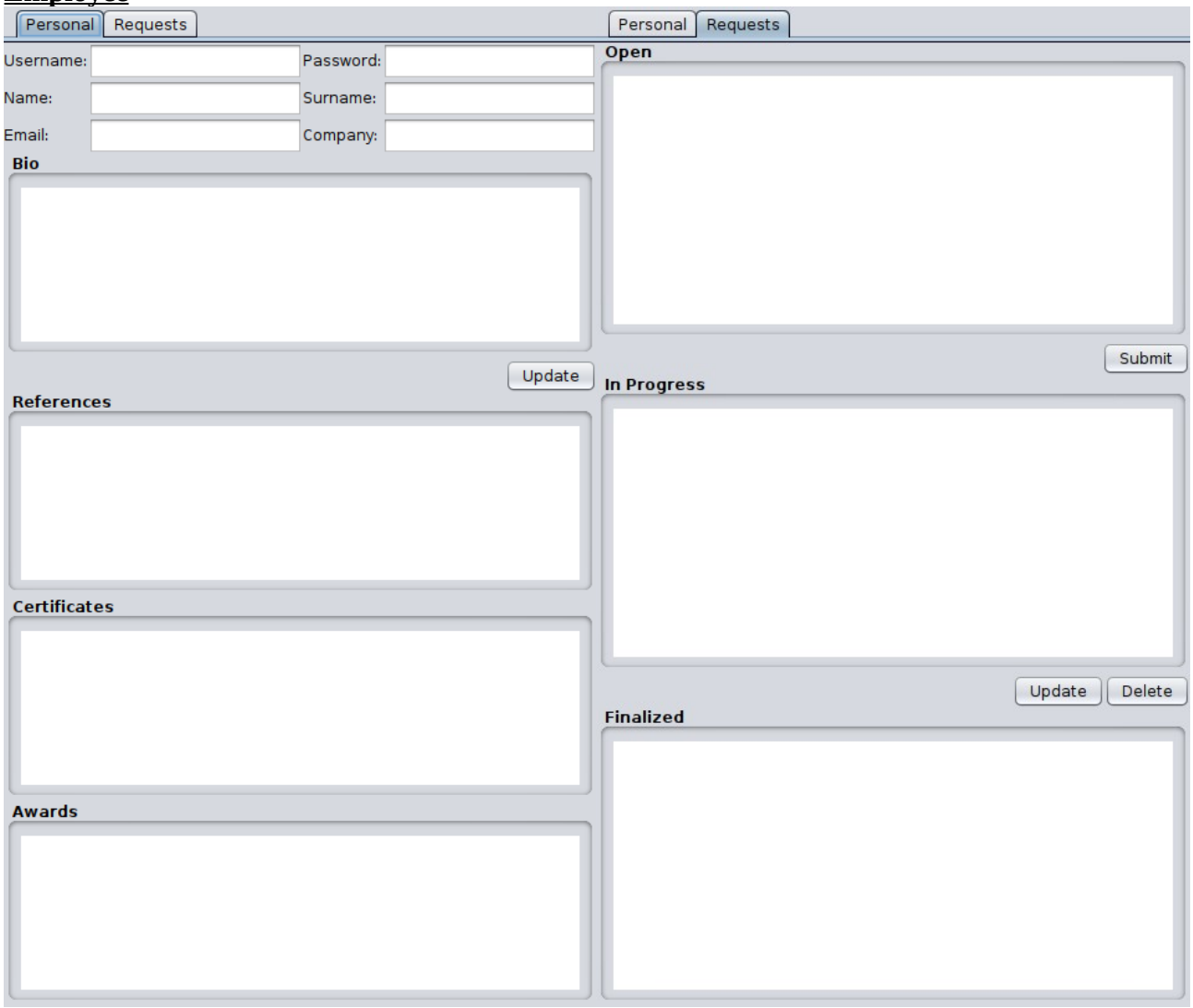

Personal
Company
Jobs
Employees
Evaluations
Evaluators
Personal
Company
Jobs
Employees
Evaluations
Evaluators
Name:
Surname:
Fetch
Fetch
Finalized
In Progress

Evaluator

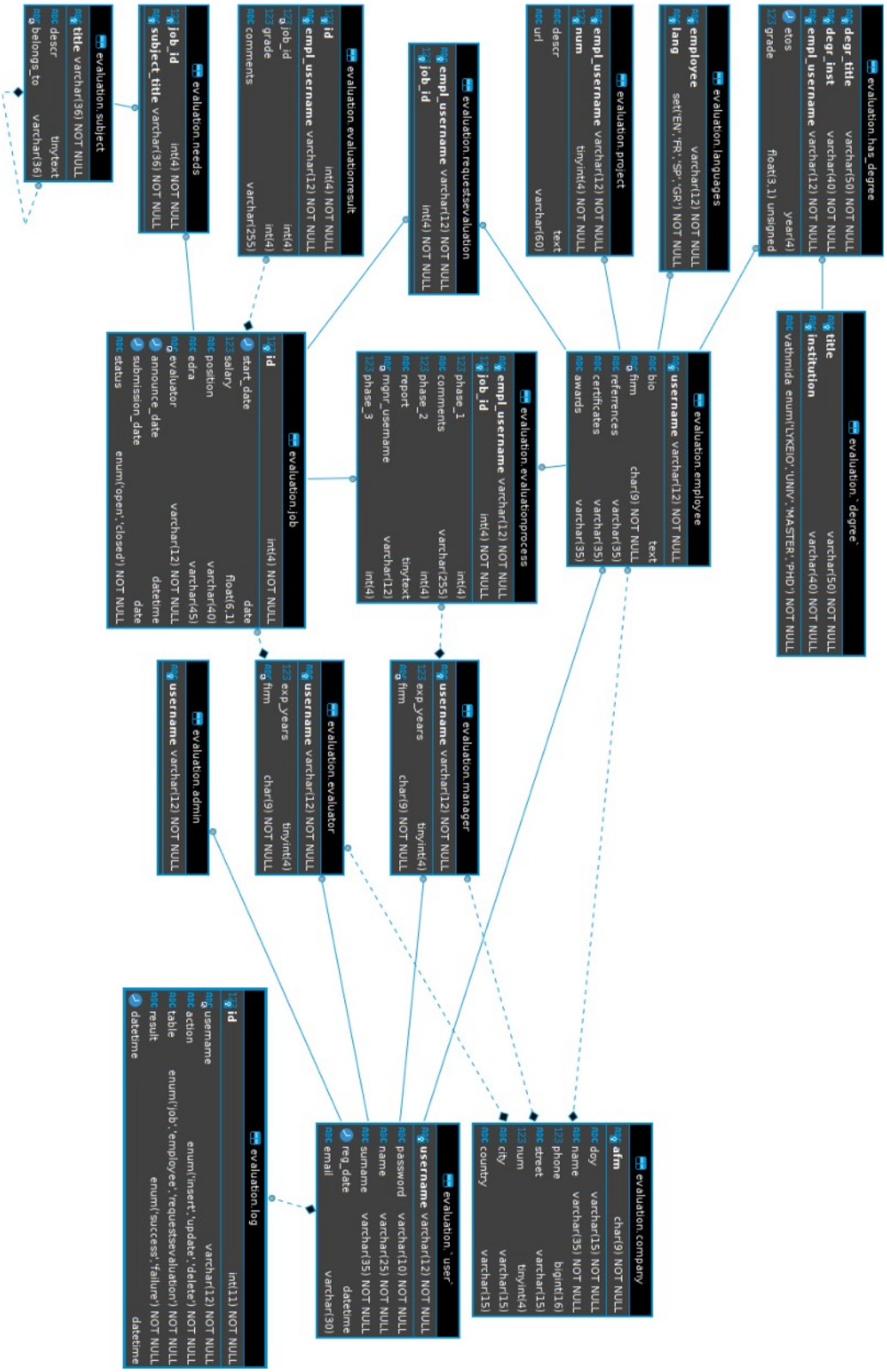

Personal
Jobs
Evaluations
Username:
Password:
Name:
Surname:
Email:
Update
Edit Jobs
Edit
ID:
Date:
Start Date:
Deadline:
Salary:
Position:
Location:
Activity:
Add/Update
Activities
Title:
Parent:
Description
Add


Personal
Jobs
Evaluations
Edit Evaluations
Job:
# of Requests:
Update

Employee


Personal
Requests
Username:
Password:
Name:
Surname:
Email:
Company:
Bio
Update
References
Certificates
Awards
Personal
Requests
Open
Submit
In Progress
Update
Delete
Finalized

Σχήμα Βάσης


evaluation.has_degree
degr_title varchar(50) NOT NULL
degr_inst varchar(40) NOT NULL
empl_username varchar(12) NOT NULL
etos year(4)
grade float(3,1) unsigned
evaluation.`degree`
title varchar(50) NOT NULL
institution varchar(40) NOT NULL
vathmida enum('LYKEIO','UNIV','MASTER','PHD') NOT NULL
evaluation.languages
employee varchar(12) NOT NULL
lang set('EN','FR','SP','GR') NOT NULL
evaluation.project
empl_username varchar(12) NOT NULL
num tinyint(4) NOT NULL
descr text
url varchar(60)
evaluation.employee
username varchar(12) NOT NULL
bio text
firm char(9) NOT NULL
references varchar(35)
certificates varchar(35)
awards varchar(35)
evaluation.evaluationresult
id int(4) NOT NULL
empl_username varchar(12) NOT NULL
job_id int(4)
grade int(4)
comments varchar(255)
evaluation.requestsevaluation
empl_username varchar(12) NOT NULL
job_id int(4) NOT NULL
evaluation.needs
job_id int(4) NOT NULL
subject_title varchar(36) NOT NULL
evaluation.subject
title varchar(36) NOT NULL
descr tinytext
belongs_to varchar(36)
evaluation.job
id int(4) NOT NULL
start_date date
salary float(6,1)
position varchar(40)
edra varchar(45)
evaluator varchar(12) NOT NULL
announce_date datetime
submission_date date
status enum('open','closed') NOT NULL
evaluation.evaluationprocess
empl_username varchar(12) NOT NULL
job_id int(4) NOT NULL
phase_1 int(4)
comments varchar(255)
phase_2 int(4)
report tinytext
mgnr_username varchar(12)
phase_3 int(4)
evaluation.manager
username varchar(12) NOT NULL
exp_years tinyint(4)
firm char(9) NOT NULL
evaluation.evaluator
username varchar(12) NOT NULL
exp_years tinyint(4)
firm char(9) NOT NULL
evaluation.admin
username varchar(12) NOT NULL
evaluation.company
afm char(9) NOT NULL
doy varchar(15) NOT NULL
name varchar(35) NOT NULL
phone bigint(16)
street varchar(15)
num tinyint(4)
city varchar(15)
country varchar(15)
evaluation.`user`
username varchar(12) NOT NULL
password varchar(10) NOT NULL
name varchar(25) NOT NULL
surname varchar(35) NOT NULL
reg_date datetime
email varchar(30)
evaluation.log
id int(11) NOT NULL
username varchar(12) NOT NULL
action enum('insert','update','delete') NOT NULL
table enum('job','employee','requestsevaluation') NOT NULL
result enum('success','failure') NOT NULL
datetime datetime

```sql
DROP DATABASE `evaluation`;
CREATE DATABASE `evaluation`;
USE `evaluation`;

--
-- Table structure for table `user`
--
DROP TABLE IF EXISTS `user`;
CREATE TABLE `user` (
  `username` varchar(12) NOT NULL,
  `password` varchar(10) NOT NULL,
  `name` varchar(25) NOT NULL,
  `surname` varchar(35) NOT NULL,
  `reg_date` datetime DEFAULT current_timestamp(),
  `email` varchar(30) NULL,
  PRIMARY KEY (`username`)
) ENGINE=InnoDB;

--
-- Table structure for table `company`
--
DROP TABLE IF EXISTS `company`;
CREATE TABLE `company` (
  `afm` char(9) NOT NULL,
  `doy` varchar(15) NOT NULL,
  `name` varchar(35) NOT NULL,
  `phone` bigint(16) NULL,
  `street` varchar(15) NULL,
  `num` tinyint(4) NULL,
  `city` varchar(15) NULL,
  `country` varchar(15) NULL,
  PRIMARY KEY (`afm`)
) ENGINE=InnoDB;

--
-- Table structure for table `degree`
--
DROP TABLE IF EXISTS `degree`;
CREATE TABLE `degree` (
  `title` varchar(50) NOT NULL,
  `institution` varchar(40) NOT NULL,
  `vathmida` enum('LYKEIO','UNIV','MASTER','PHD') NOT NULL,
  PRIMARY KEY (`title`,`institution`)
) ENGINE=InnoDB;

--
-- Table structure for table `admin`
--
DROP TABLE IF EXISTS `admin`;
CREATE TABLE `admin` (
  `username` varchar(12) NOT NULL,
  PRIMARY KEY (`username`),
  CONSTRAINT `fk_admin_user` FOREIGN KEY (`username`) REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

--
-- Table structure for table `manager`
--
DROP TABLE IF EXISTS `manager`;
CREATE TABLE `manager` (
  `username` varchar(12) NOT NULL,
  `exp_years` tinyint(4) NULL,
  `firm` char(9) NOT NULL,
  PRIMARY KEY (`username`),
  KEY `fk_manager_company` (`firm`),
  CONSTRAINT `fk_manager_company` FOREIGN KEY (`firm`) REFERENCES `company` (`afm`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_manager_users` FOREIGN KEY (`username`) REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

--
-- Table structure for table `evaluator`
--
DROP TABLE IF EXISTS `evaluator`;
CREATE TABLE `evaluator` (
  `username` varchar(12) NOT NULL,
  `exp_years` tinyint(4) NULL,
  `firm` char(9) NOT NULL,
  PRIMARY KEY (`username`),
  KEY `fk_evaluator_company` (`firm`),
  CONSTRAINT `fk_evaluator_company` FOREIGN KEY (`firm`) REFERENCES `company` (`afm`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_evaluator_user` FOREIGN KEY (`username`) REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

--
-- Table structure for table `employee`
--
DROP TABLE IF EXISTS `employee`;
CREATE TABLE `employee` (
  `username` varchar(12) NOT NULL,
  `bio` text NULL,
```

```
  `firm` char(9) NOT NULL,
  `referrences` varchar(35) NULL,
  `certificates` varchar(35) NULL,
  `awards` varchar(35) NULL,
  PRIMARY KEY (`username`),
  KEY `fk_employee_company` (`firm`),
  CONSTRAINT `fk_employee_company` FOREIGN KEY (`firm`) REFERENCES `company` (`afm`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_employee_user` FOREIGN KEY (`username`) REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

--
-- Table structure for table `job`
--
DROP TABLE IF EXISTS `job`;
CREATE TABLE `job` (
  `id` int(4) NOT NULL AUTO_INCREMENT,
  `start_date` date NULL,
  `salary` float(6,1) NULL,
  `position` varchar(40) NULL,
  `edra` varchar(45) NULL,
  `evaluator` varchar(12) NOT NULL,
  `announce_date` datetime NULL,
  `submission_date` date NULL,
  `status` enum('open','closed') NOT NULL DEFAULT 'open',
  PRIMARY KEY (`id`),
  KEY `fk_job_evaluator` (`evaluator`),
  CONSTRAINT `fk_job_evaluator` FOREIGN KEY (`evaluator`) REFERENCES `evaluator` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

--
-- Table structure for table `evaluationprocess`
--
DROP TABLE IF EXISTS `evaluationprocess`;
CREATE TABLE `evaluationprocess` (
  `empl_username` varchar(12) NOT NULL,
  `job_id` int(4) NOT NULL,
  `phase_1` int(4) DEFAULT NULL,
  `comments` varchar(255) NULL,
  `phase_2` int(4) DEFAULT NULL,
  `report` tinytext NULL,
  `mgnr_username` varchar(12) NULL,
  `phase_3` int(4) DEFAULT NULL,
  PRIMARY KEY (`empl_username`,`job_id`),
  KEY `fk_evaluationprocess_job` (`job_id`),
  KEY `fk_evaluationprocess_manager` (`mgnr_username`),
  CONSTRAINT `fk_evaluationprocess_employee` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON
UPDATE CASCADE,
  CONSTRAINT `fk_evaluationprocess_job` FOREIGN KEY (`job_id`) REFERENCES `job` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_evaluationprocess_manager` FOREIGN KEY (`mgnr_username`) REFERENCES `manager` (`username`) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE=InnoDB;

--
-- Table structure for table `evaluationresult`
--
DROP TABLE IF EXISTS `evaluationresult`;
CREATE TABLE `evaluationresult` (
  `id` int(4) NOT NULL AUTO_INCREMENT,
  `empl_username` varchar(12) NOT NULL,
  `job_id` int(4) NULL,
  `grade` int(4) NULL,
  `comments` varchar(255) NULL,
  PRIMARY KEY (`id`,`empl_username`),
  KEY `fk_evaluationresult_employee` (`empl_username`),
  KEY `fk_evaluationresult_job` (`job_id`),
  CONSTRAINT `fk_evaluationresult_employee` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE
CASCADE,
  CONSTRAINT `fk_evaluationresult_job` FOREIGN KEY (`job_id`) REFERENCES `job` (`id`) ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB;

--
-- Table structure for table `has_degree`
--
DROP TABLE IF EXISTS `has_degree`;
CREATE TABLE `has_degree` (
  `degr_title` varchar(50) NOT NULL,
  `degr_inst` varchar(40) NOT NULL,
  `empl_username` varchar(12) NOT NULL,
  `etos` year(4) NULL,
  `grade` float(3,1) unsigned NULL,
  PRIMARY KEY (`degr_title`,`degr_inst`,`empl_username`),
  KEY `fk_has_degree_employee` (`empl_username`),
  CONSTRAINT `fk_has_degree_degree` FOREIGN KEY (`degr_title`, `degr_inst`) REFERENCES `degree` (`title`, `institution`) ON DELETE
CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_has_degree_employee` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE=InnoDB;

--
-- Table structure for table `languages`
```

```sql
--
DROP TABLE IF EXISTS `languages`;
CREATE TABLE `languages` (
  `employee` varchar(12) NOT NULL,
  `lang` set('EN','FR','SP','GR') NOT NULL,
  PRIMARY KEY (`employee`,`lang`),
  CONSTRAINT `fk_languages_employee` FOREIGN KEY (`employee`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

--
-- Table structure for table `log`
--
DROP TABLE IF EXISTS `log`;
CREATE TABLE `log` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(12) NOT NULL,
  `action` enum('insert','update','delete') NOT NULL,
  `table` enum('job','employee','requestsevaluation') NOT NULL,
  `result` enum('success','failure') NOT NULL,
  `datetime` datetime DEFAULT current_timestamp(),
  PRIMARY KEY (`id`),
  KEY `fk_log_user` (`username`),
  CONSTRAINT `fk_log_user` FOREIGN KEY (`username`) REFERENCES `user` (`username`) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB;

--
-- Table structure for table `subject`
--
DROP TABLE IF EXISTS `subject`;
CREATE TABLE `subject` (
  `title` varchar(36) NOT NULL,
  `descr` tinytext NULL,
  `belongs_to` varchar(36) NULL,
  PRIMARY KEY (`title`),
  UNIQUE KEY `title_UNIQUE` (`title`),
  KEY `fk_subject_subject` (`belongs_to`),
  CONSTRAINT `fk_subject_subject` FOREIGN KEY (`belongs_to`) REFERENCES `subject` (`title`) ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB;

--
-- Table structure for table `needs`
--
DROP TABLE IF EXISTS `needs`;
CREATE TABLE `needs` (
  `job_id` int(4) NOT NULL,
  `subject_title` varchar(36) NOT NULL,
  PRIMARY KEY (`job_id`,`subject_title`),
  KEY `fk_needs_subject` (`subject_title`),
  CONSTRAINT `fk_needs_job` FOREIGN KEY (`job_id`) REFERENCES `job` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_needs_subject` FOREIGN KEY (`subject_title`) REFERENCES `subject` (`title`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

--
-- Table structure for table `requestsevaluation`
--
DROP TABLE IF EXISTS `requestsevaluation`;
CREATE TABLE `requestsevaluation` (
  `empl_username` varchar(12) NOT NULL,
  `job_id` int(4) NOT NULL,
  PRIMARY KEY (`empl_username`,`job_id`),
  KEY `fk_requestsevaluation_job` (`job_id`),
  CONSTRAINT `fk_requestsevaluation_employee` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON
UPDATE CASCADE,
  CONSTRAINT `fk_requestsevaluation_job` FOREIGN KEY (`job_id`) REFERENCES `job` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

--
-- Table structure for table `project`
--
DROP TABLE IF EXISTS `project`;
CREATE TABLE `project` (
  `empl_username` varchar(12) NOT NULL,
  `num` tinyint(4) NOT NULL,
  `descr` text NULL,
  `url` varchar(60) NULL,
  PRIMARY KEY (`empl_username`,`num`),
  CONSTRAINT `fk_project_employee` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

--
-- Triggers for table `company`
--
-- Αποτρέπει την αλλαγή συγκεκριμένων στοιχείων της εταιρίας
DELIMITER ;;
CREATE TRIGGER `company_BEFORE_UPDATE` BEFORE UPDATE ON `company` FOR EACH ROW
BEGIN
    SET NEW.afm = OLD.afm;
    SET NEW.doy = OLD.doy;
    SET NEW.name = OLD.name;
END;;
```

```
DELIMITER ;

--
-- Triggers for table `employee`
--
-- Συνδιασμός απο triggers που καταγράφουν στον `log` τις αλλαγές που συμβαίνουν
-- στον `employee`. Καλούν συγκεκριμένα procedures. Η λειτουργία τους αναλύεται εκεί
DELIMITER ;;
CREATE TRIGGER `employee_BEFORE_INSERT` BEFORE INSERT ON `employee` FOR EACH ROW
BEGIN
    CALL proc_log_before("insert", "employee", "failure");
END;;
DELIMITER ;
DELIMITER ;;
CREATE TRIGGER `employee_AFTER_INSERT` AFTER INSERT ON `employee` FOR EACH ROW
BEGIN
    CALL proc_log_after();
END;;
DELIMITER ;
DELIMITER ;;
CREATE TRIGGER `employee_BEFORE_UPDATE` BEFORE UPDATE ON `employee` FOR EACH ROW
BEGIN
    CALL proc_log_before("update", "employee", "failure");
END;;
DELIMITER ;
DELIMITER ;;
CREATE TRIGGER `employee_AFTER_UPDATE` AFTER UPDATE ON `employee` FOR EACH ROW
BEGIN
    CALL proc_log_after();
END;;
DELIMITER ;

--
-- Triggers for table `job`
--
-- Συνδιασμός απο triggers που καταγράφουν στον `log` τις αλλαγές που συμβαίνουν
-- στον `job`. Καλούν συγκεκριμένα procedures. Η λειτουργία τους αναλύεται εκεί
DELIMITER ;;
CREATE TRIGGER `job_BEFORE_INSERT` BEFORE INSERT ON `job` FOR EACH ROW
BEGIN
    CALL proc_log_before("insert", "job", "failure");
END;;
DELIMITER ;
DELIMITER ;;
CREATE TRIGGER `job_AFTER_INSERT` AFTER INSERT ON `job` FOR EACH ROW
BEGIN
    CALL proc_log_after();
END;;
DELIMITER ;
DELIMITER ;;
CREATE TRIGGER `job_BEFORE_UPDATE` BEFORE UPDATE ON `job` FOR EACH ROW
BEGIN
    CALL proc_log_before("update", "job", "failure");
END;;
DELIMITER ;
DELIMITER ;;
CREATE TRIGGER `job_AFTER_UPDATE` AFTER UPDATE ON `job` FOR EACH ROW
BEGIN
    CALL proc_log_after();
END;;
DELIMITER ;

--
-- Triggers for table `reqeustsevaluation`
--
-- Συνδιασμός απο triggers που καταγράφουν στον `log` τις αλλαγές που συμβαίνουν
-- στον `reqeustsevaluation`. Καλούν συγκεκριμένα procedures. Η λειτουργία τους αναλύεται εκεί
DELIMITER ;;
CREATE TRIGGER `requestsevaluation_BEFORE_INSERT` BEFORE INSERT ON `requestsevaluation` FOR EACH ROW
BEGIN
    CALL proc_log_before("insert", "requestsevaluation", "failure");
END;;
DELIMITER ;
DELIMITER ;;
CREATE TRIGGER `requestsevaluation_AFTER_INSERT` AFTER INSERT ON `requestsevaluation` FOR EACH ROW
BEGIN
    CALL proc_log_after();
END;;
DELIMITER ;
DELIMITER ;;
CREATE TRIGGER `requestsevaluation_BEFORE_UPDATE` BEFORE UPDATE ON `requestsevaluation` FOR EACH ROW
BEGIN
    CALL proc_log_before("update", "requestsevaluation", "failure");
END;;
DELIMITER ;
DELIMITER ;;
CREATE TRIGGER `requestsevaluation_AFTER_UPDATE` AFTER UPDATE ON `requestsevaluation` FOR EACH ROW
BEGIN
    CALL proc_log_after();
END;;
```

```
DELIMITER ;

--
-- Triggers for table `evaluationprocess`
--
-- Triggers που ελέγχουν αν υπάρχει βαθμός και στις τρεις φάσεις αξιολόγησης μετά από είσοδο ή
-- αλλαγή δεδομένων. Αν υπάρχει και στις τρεις φάσεις, καταγράφουν το αποτέλεσμα της αξιολόγησης
-- στον `evaluationresult`. Το υλοποιήσαμε πριν προχωρήσουμε στα απαιτούμενα procedures και το αφήσαμε
-- σαν δικλείδα ασφαλείας. Επίσης, αχρηστεύει το αντίστοιχο procedure.
DELIMITER ;;
CREATE TRIGGER `evaluationprocess_AFTER_INSERT` AFTER INSERT ON `evaluationprocess` FOR EACH ROW
BEGIN
    DECLARE _grade_1 INT(4);
    DECLARE _grade_2 INT(4);
    DECLARE _grade_3 INT(4);
    DECLARE _comments VARCHAR(255);
    DECLARE _username VARCHAR(12);
    DECLARE _job_id INT(4);
    SELECT
        empl_username, job_id, phase_1, comments, phase_2, phase_3
    FROM
        evaluation.evaluationprocess WHERE empl_username = NEW.empl_username AND job_id = NEW.job_id
    INTO
        _username, _job_id, _grade_1, _comments, _grade_2, _grade_3;

    IF ((_grade_1 IS NOT NULL) AND (_grade_2 IS NOT NULL) AND (_grade_3 IS NOT NULL)) THEN
        INSERT INTO evaluation.evaluationresult(empl_username, job_id, grade, comments)
        VALUES(_username, _job_id, _grade_1+_grade_2+_grade_3, _comments);
    END IF;
END;;
DELIMITER ;
DELIMITER ;;
CREATE TRIGGER `evaluationprocess_AFTER_UPDATE` AFTER UPDATE ON `evaluationprocess` FOR EACH ROW
BEGIN
    DECLARE _grade_1 INT(4);
    DECLARE _grade_2 INT(4);
    DECLARE _grade_3 INT(4);
    DECLARE _comments VARCHAR(255);
    DECLARE _username VARCHAR(12);
    DECLARE _job_id INT(4);
    SELECT
        empl_username, job_id, phase_1, comments, phase_2, phase_3
    FROM
        evaluation.evaluationprocess WHERE empl_username = NEW.empl_username AND job_id = NEW.job_id
    INTO
        _username, _job_id, _grade_1, _comments, _grade_2, _grade_3;

    IF ((_grade_1 IS NOT NULL) AND (_grade_2 IS NOT NULL) AND (_grade_3 IS NOT NULL)) THEN
        INSERT INTO evaluation.evaluationresult(empl_username, job_id, grade, comments)
        VALUES(_username, _job_id, _grade_1+_grade_2+_grade_3, _comments);
    END IF;
END;;
DELIMITER ;

--
-- Triggers for table `evaluationresult`
--
-- Trigger που ελέγχει αν μία αξιολόγηση έχει ήδη ολοκληρωθεί και δεν επιτρέπει την εισαγωγή
-- νέας εγγραφής για τον ίδιο `employee` και `job_id`.
DELIMITER ;;
CREATE TRIGGER `evaluationresult_BEFORE_INSERT` BEFORE INSERT ON `evaluationresult` FOR EACH ROW
BEGIN
    DECLARE _exists INT;

    SELECT
        EXISTS(
            SELECT * FROM `evaluation`.`evaluationresult`
            WHERE empl_username = NEW.empl_username AND job_id = NEW.job_id)
    INTO _exists;

    IF (_exists) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Evaluation already finalized';
    END IF;
END;;
DELIMITER ;
-- Δεν επιτρέπει την αλλαγή των αποτελεσμάτων των αξιολογήσεων απενεργοποιώντας
-- συνολικά τα updates στον πίνακα.
DELIMITER ;;
CREATE TRIGGER `evaluationresult_BEFORE_UPDATE` BEFORE UPDATE ON `evaluationresult` FOR EACH ROW
BEGIN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Updates are disabled. Results are final';
END;;
DELIMITER ;

--
-- Triggers for table `user`
--
-- Αποτρέπει την εισαγωγή δεδομένων στον πίνακα αν ο χρήστης δεν έχει ρόλο administrator
-- Επίσης αποτρέπει μια διαφορετικό μήνυμα την εισαγωγή αν δεν έχει δηλωθεί ρόλος
DELIMITER ;;
```

```sql
CREATE TRIGGER `user_BEFORE_INSERT` BEFORE INSERT ON `user` FOR EACH ROW
BEGIN
    IF (@evaluation_current_role IS NULL) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'evaluation_current_role is unset';
    END IF;

    IF (@evaluation_current_role != "administrator") THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'evaluation_current_role is not administrator';
    END IF;
END;;
DELIMITER ;
-- Υλοποιεί τα δικαιώματα των διαφορετικών ρόλων στην αλλαγή στοιχείων στον πίνακα `user`
-- Administrator: αλλάζει τα πάντα εκτός απο το reg_date
-- Evaluator: δεν αλλάζει το username και reg_date
-- Manager: Δεν αλλάζει τα username, name, surname, reg_date
-- Employee: Δεν αλλάζει τα username, name, surname, reg_date, email
DELIMITER ;;
CREATE TRIGGER `user_BEFORE_UPDATE` BEFORE UPDATE ON `user` FOR EACH ROW
BEGIN
    IF (@evaluation_current_role IS NULL) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'evaluation_current_role is unset';
    END IF;
    SET NEW.reg_date = OLD.reg_date;
    IF (@evaluation_current_role != "administrator") THEN
        SET NEW.username = OLD.username;
        IF (@evaluation_current_role = "manager") THEN
            SET NEW.name = OLD.name;
            SET NEW.surname = OLD.surname;
        END IF;
        IF (@evaluation_current_role = "employee") THEN
            SET NEW.name = OLD.name;
            SET NEW.surname = OLD.surname;
            SET NEW.email = OLD.email;
        END IF;
    END IF;
END;;
DELIMITER ;

--
-- Dumping routines for database 'evaluation'
--
-- Procedure που ζητήθηκε σαν ερώτημα, οριστικοποιεί τις αξιολογήσεις που έχουν βαθμούς και στις
-- τρεις φάσεις αλλά δεν υπάρχουν αντίστοιχες εγγραφες στον πίνακα `evaluationresult`.
DROP PROCEDURE IF EXISTS `proc_finalize_employee_evaluation`;
DELIMITER ;;
CREATE  PROCEDURE `proc_finalize_employee_evaluation`(IN in_job_id INT, IN in_evaluator VARCHAR(12))
BEGIN
    DECLARE _username VARCHAR(12);
    DECLARE _job_id INT(4);
    DECLARE _grade INT(4);
    DECLARE _comments VARCHAR(255);
    DECLARE _done BOOL DEFAULT FALSE;
    DECLARE _exists INT;
    DECLARE _cur CURSOR FOR
        SELECT
            empl_username, job_id, phase_1+phase_2+phase_3 as grade, comments
        FROM
            evaluationprocess INNER JOIN job ON evaluationprocess.job_id = job.id
        WHERE
            evaluator = in_evaluator AND job_id = in_job_id;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET _done = TRUE;
    -- Handler για exception που μπορεί να προκύψει από το before insert trigger του `user`
    DECLARE CONTINUE HANDLER FOR SQLSTATE '45000' BEGIN END;

    OPEN _cur;
    insert_loop: LOOP
        FETCH _cur INTO _username, _job_id, _grade, _comments;
        IF _done THEN
            LEAVE insert_loop;
        END IF;
        SELECT
            EXISTS(
                SELECT * FROM evaluationresult
                WHERE empl_username = _username AND job_id = _job_id)
        INTO _exists;
        IF ((_grade IS NOT NULL) AND (!_exists)) THEN
            INSERT INTO evaluationresult (empl_username, job_id, grade, comments)
            VALUES(_username, _job_id, _grade, _comments);
        END IF;
    END LOOP;
    CLOSE _cur;
END ;;
DELIMITER ;

-- Procedure που ζητήθηκε, βρίσκει τις αιτήσεις ενός employee, τις ολοκληρωμένες αξιολογήσεις του,
-- τον αντιστοιχο αξιολογητή, εμφανίζει και τις μη ολοκληρωμένες
DROP PROCEDURE IF EXISTS `proc_find_employee_evaluation`;
DELIMITER ;;
CREATE PROCEDURE `proc_find_employee_evaluation`
(IN in_name varchar(25), IN in_surname varchar(35))
```

```
BEGIN
    declare _username varchar(12);
    declare _num_eval_req int;
    declare _num_eval_compl int;

    SELECT username INTO _username
    FROM user
    WHERE user.name=in_name AND user.surname=in_surname;

    SELECT count(*) INTO _num_eval_req
    FROM requestsevaluation
    WHERE requestsevaluation.empl_username=_username;

    SELECT
        user.username, requestsevaluation.job_id, grade, evaluator, _evaluator.name, _evaluator.surname
    FROM user
        INNER JOIN requestsevaluation
            ON user.username = requestsevaluation.empl_username
        INNER JOIN job
            ON requestsevaluation.job_id = job.id
        INNER JOIN evaluationresult
            ON evaluationresult.empl_username = user.username AND evaluationresult.job_id = job.id
        INNER JOIN user AS _evaluator
            ON job.evaluator = _evaluator.username
    WHERE user.name = in_name AND user.surname = in_surname;
    SELECT FOUND_ROWS() INTO _num_eval_compl;

    IF (_num_eval_compl<_num_eval_req) THEN
        SELECT 'Evaluations in progress' AS message;
        SELECT
            user.username, requestsevaluation.job_id, phase_1, phase_2, phase_3, evaluator, _evaluator.name, _evaluator.surname
        FROM user
            INNER JOIN requestsevaluation
                ON user.username = requestsevaluation.empl_username
            INNER JOIN job
                ON requestsevaluation.job_id = job.id
            INNER JOIN evaluationprocess
                ON evaluationprocess.empl_username = user.username AND evaluationprocess.job_id = job.id
            INNER JOIN user AS _evaluator
                ON job.evaluator = _evaluator.username
        WHERE user.name = in_name
            AND user.surname = in_surname
            AND ((evaluationprocess.phase_1 IS NULL)
                OR (evaluationprocess.phase_2 IS NULL)
                OR (evaluationprocess.phase_1 IS NULL));
    END IF;
END ;;
DELIMITER ;

-- Procedure για το login ενός user. Παίρνει σαν όρισμα το username και το password, βρίσκει τον χρήστη καθώς
-- και την ιδιότητα του και επιστρέφει τον ρόλο του. Επίσης θέτει αντίστοιχα session variables για χρήση από triggers
-- και άλλα procedures.
DROP PROCEDURE IF EXISTS `proc_login_user`;
DROP PROCEDURE IF EXISTS `evaluation`.`proc_login_user`;
DELIMITER ;;
CREATE PROCEDURE `proc_login_user`(IN in_username varchar(12), IN in_password varchar(10), OUT out_role varchar(15))
BEGIN
    DECLARE _exists INT;
    SELECT
        EXISTS(SELECT * FROM `user`
            WHERE `user`.`username` = in_username
                AND `user`.`password` = in_password) INTO _exists;

    IF (!_exists) THEN
        SELECT NULL INTO out_role;
    ELSE
        SELECT
        CASE
            WHEN EXISTS(SELECT * FROM `admin` WHERE `admin`.`username` = in_username) THEN "administrator"
            WHEN EXISTS(SELECT * FROM `manager` WHERE `manager`.`username` = in_username) THEN "manager"
            WHEN EXISTS(SELECT * FROM `evaluator` WHERE `evaluator`.`username` = in_username) THEN "evaluator"
            WHEN EXISTS(SELECT * FROM `employee` WHERE `employee`.`username` = in_username) THEN "employee"
            ELSE NULL
        END INTO out_role;
        SET @evaluation_current_user = in_username;
        SET @evaluation_current_role = out_role;
    END IF;
END;;
DELIMITER ;

-- Triggers για την καταγραφή στον `log`
-- Για χρήση με τα αντίστοιχα before triggers. Καταγράφει στον `log` την ενέργεια ενός χρήστη.
-- το result καταγράφεται πάντα ως failure.
DROP PROCEDURE IF EXISTS `proc_log_before`;
DELIMITER ;;
CREATE  PROCEDURE `proc_log_before`(
    IN in_action ENUM('insert','update','delete'),
    IN in_table ENUM('job','employee','requestsevaluation'),
    IN in_result ENUM('success','failure'))
BEGIN
```

```
625         DECLARE _now DATETIME;
626         SELECT NOW() INTO _now;
627         INSERT INTO `evaluation`.`log`
628             (`username`, `action`, `table`, `result`, `datetime`)
629         VALUES (@evaluation_current_user,
630                 in_action,
631                 in_table,
632                 'failure',
633                 _now);
634     END ;;
635     DELIMITER ;
636     -- Για χρήση με τα αντίστοιχα after triggers. Επιλέγει το τελευταίο στοιχείο στον `log` και αλλάζει το
637     -- αποτέλεσμα απο 'failure' σε 'success'. Βασίζεται στο ότι το after trigger εκτελείται μόνο όταν ενέργεια
638     -- ολοκληρωθεί σωστά. Υποθέτει δύο συνθήκες, ότι ένας χρήστης είναι συνδεδεμένος και κάνει αλλαγές και ότι
639     -- δεν συμβαίνουν batch inserts/updates.
640     DROP PROCEDURE IF EXISTS `proc_log_after`;
641     DELIMITER ;;
642     CREATE  PROCEDURE `proc_log_after`()
643     BEGIN
644         DECLARE _id INT;
645         SELECT id FROM log ORDER BY id DESC LIMIT 1 INTO _id;
646         UPDATE log
647             SET result = "success"
648         WHERE log.id = _id;
649     END ;;
650     DELIMITER ;
651
652     -- Procedure που ζητήθηκε, εμφανίζει τις ολοκληρωμένες αξιολογήσεις για μία θέση. Αν υπάρχουν
653     -- εμφανίζει και τον αριθμό των αιτήσεων σε εκκρεμότητα.
654     DROP PROCEDURE IF EXISTS `proc_print_employee_evaluation`;
655     DELIMITER ;;
656     CREATE  PROCEDURE `proc_print_employee_evaluation`(IN in_job_id INT)
657     BEGIN
658         DECLARE _active_evaluations INT;
659         DECLARE _final_evaluations INT;
660
661         SELECT COUNT(*) FROM evaluationresult WHERE job_id = in_job_id INTO _final_evaluations;
662         SELECT COUNT(*) FROM requestsevaluation WHERE job_id = in_job_id INTO _active_evaluations;
663
664         IF (!_active_evaluations) THEN
665             SELECT "No candidates for this job." AS "Status";
666         ELSE
667             IF (_active_evaluations = _final_evaluations) THEN
668                 SELECT "Finalized evaluation." AS "Status";
669                 SELECT * FROM evaluationresult WHERE job_id = in_job_id ORDER BY grade DESC;
670             ELSEIF (_active_evaluations > _final_evaluations) THEN
671                 SELECT * FROM evaluationresult WHERE job_id = in_job_id ORDER BY grade DESC;
672                 SELECT CONCAT("Evaluation in progress. ", _active_evaluations-_final_evaluations, " requests remaining.") AS "Status";
673             ELSE
674                 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'This should not happen';
675             END IF;
676         END IF;
677     END ;;
678     DELIMITER ;
679
```