

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Σύστημα αξιολόγησης υπάλληλων

ΥΛΟΠΟΙΗΣΗ:

ΒΕΝΟΣ ΑΝΑΣΤΑΣΙΟΣ 1067536

ΖΕΡΒΟΥ ΠΟΥΛΧΕΡΙΑ 1067511

ΠΟΡΤΟΚΑΛΟΓΛΟΥ ΑΝΔΡΟΝΙΚΗ 1067539

Περιεχόμενα

1. Το ER της βάσης

2. Μετάβαση στο σχεσιακό μοντέλο

3. Υλοποίηση σε SQL

3.1. Δημιουργία πινάκων

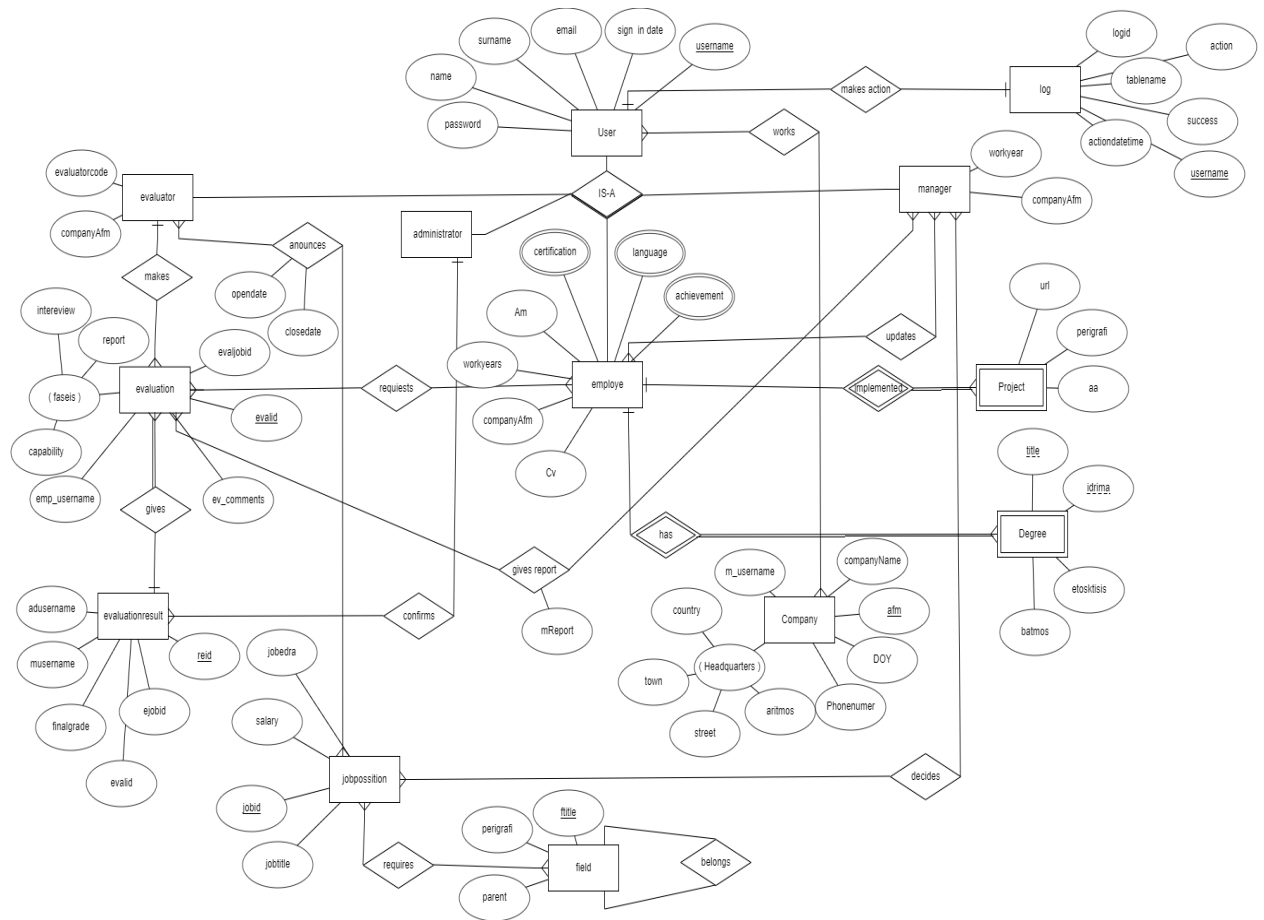
3.2. Εισαγωγή στοιχείων

3.3 Stored Procedures της βάσης

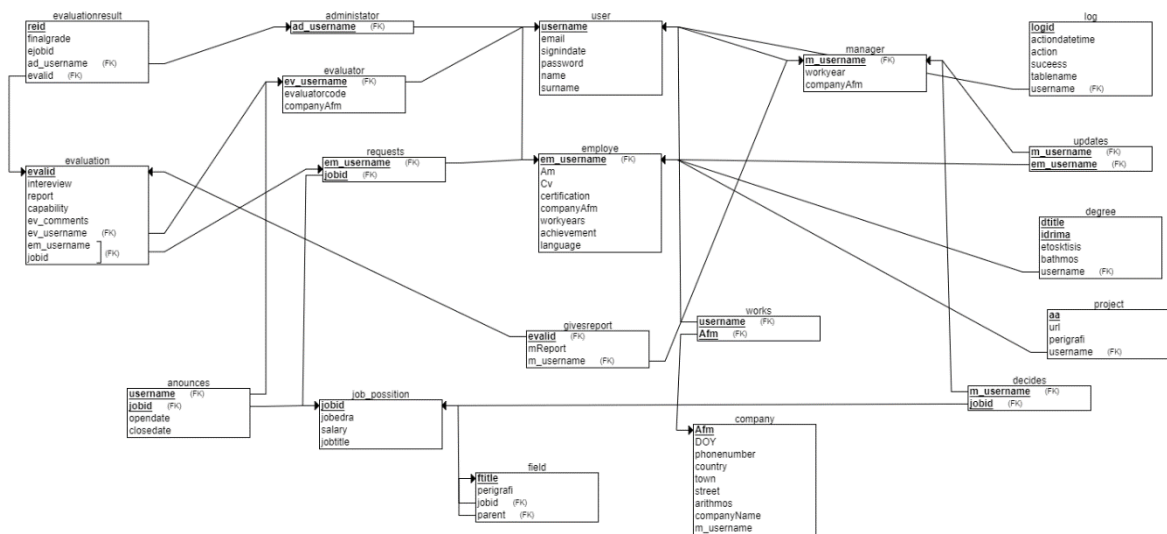
3.4 Triggers

4. Υλοποίηση GUI με την βοήθεια της Python

1.Το ER της βάσης



2.Μεταβαση στο σχεσιακό μοντέλο



Το διάγραμμα οντοτήτων και το σχεσιακό έχουν επισυνάπτεi στο φάκελο για πιο λεπτομερειακή παρατήρηση .

3.Υλοποίηση σε SQL

3.1. Δημιουργία πινάκων

Έχουν δημιουργηθεί 20 πίνακες οι οποίοι είναι οι:

user, log, company, employe, administrator, evaluator, manager, updates, works, degree, project, evaluation, givesreport, evaluationresult, job_postition, announces, requests, field ,decides.

Οι πίνακες είναι βασισμένοι στην περιγραφή της εκφώνησης ,στο ER και στο σχεσιακό που έχουμε δημιουργήσει .Ωστόσο υπάρχουν αρκετά πράγματα τα οποία προσθέσαμε που βρίσκονται πιο κοντά στη προσωπική μας αντίληψη της βάσης

Τα επιπλέον γνωρίσματα που διαθέτουμε είναι ο πίνακας degree που έχει το γνώρισμα username το οποίο κληρονομεί από τον πίνακα employe . Ο πίνακας Job_position που κρατάει το τον τίτλο της δουλείας ως το γνώρισμα jobtitle .Το γνώρισμα logid στον πίνακα log που αποτελεί το primary key του όπως και αντίστοιχα στον πίνακα evaluation το evalid και επιπλέον το γνώρισμα en_username που διατηρείται το ονομα του evaluator που κάνει την αξιολόγηση. Τέλος στον πίνακα evaluationresult όπου όλα τα γνωρίσματα ήταν δικιά μας πρωτοβουλία έχουμε το primary key reid , το finalgrade που είναι ο συνολικός βαθμός της αξιολόγησης (δηλαδή το άθροισμα των βαθμών του interview,report και capability) και τα ad_username ,evalid που κληρονομούνται από τον πίνακα administrator και evaluator αντίστοιχα.

3.2. Εισαγωγή στοιχείων

Βάλαμε τιμές για κάθε γνώρισμα των πινάκων με την εντολή INSERT INTO table_name VALUES(values);

Για να γίνει ευκολότερη η διαδικασία της εισαγωγής βασιστήκαμε σε πολλές τιμές στη σειρά “The office”.

3.3 Stored Procedures της βάσης

3.3.1

Procedure η οποία δέχεται ως είσοδο το όνομα και επώνυμο ενός εργαζόμενου.

```
CREATE PROCEDURE employee(IN em_name VARCHAR(20), IN em_surname VARCHAR(20))
```

Επιλέγεται με την εντολή Select να εμφανίζονται το username, αριθμός αξιολόγησης και αριθμός εργασίας και τα δίνω ένα ψευδώνυμο και με Inner Join τα συνδέω στον πίνακα User, έτσι ώστε η είσοδος που δίνω να βρει το αντίστοιχο όνομα και επώνυμο του χρήστη που θέλω.

```
SELECT requests.em_username AS Username, requests.evalid AS Evaluation_ID,  
requests.jobid AS Job_ID FROM requests  
INNER JOIN employe ON requests.em_username=employe.em_username  
INNER JOIN user ON employe.em_username=user.username  
WHERE em_name=user.name AND em_surname=user.surname;
```

Επιλέγεται να εμφανιστεί η αναφορά του manager, η ικανότητα του εργαζόμενου, η συνέντευξη που έδωσε και τα σχόλια του αξιολογητή. Με Inner Join τα συνδέονται στον πίνακα User, έτσι ώστε η είσοδος που δίνεται να βρει το αντίστοιχο όνομα και επώνυμο του χρήστη που θέλω.

```
SELECT evaluation.report AS Report, evaluation.capability AS Capability, evaluation.interview AS Interview,  
evaluation.ev_comments AS comments FROM evaluation  
INNER JOIN requests ON evaluation.evalid=requests.evalid  
INNER JOIN employe ON requests.em_username=employe.em_username  
INNER JOIN user ON employe.em_username=user.username  
WHERE em_name=user.name AND em_surname=user.surname;
```

Επιλέγεται να εμφανιστεί το όνομα και επώνυμο του αξιολογητή που βαθμολόγησε τον εργαζόμενο. Ύστερα υπάρχουν εμφωλευμένες Select. Στην εσωτερική, μου εμφανίζει το username του αντίστοιχου αξιολογητή. Στην εξωτερική, εμφανίζει το όνομα και επώνυμο του αξιολογητή με το username που εμφανίζεται στην εσωτερική select.

```
SELECT user.name, user.surname FROM user
INNER JOIN evaluator ON user.username=evaluator.ev_username
INNER JOIN evaluation ON evaluator.ev_username=evaluation.ev_username
WHERE evaluation.ev_username IN
(SELECT evaluation.ev_username FROM evaluation
INNER JOIN requests ON evaluation.evalid=requests.evalid
INNER JOIN employe ON requests.em_username=employe.em_username
INNER JOIN user ON employe.em_username=user.username
WHERE em_name=user.name AND em_surname=user.surname);
```

3.3.2

Δημιουργείται την stored procedure sumofevaluation η οποία παίρνει είσοδο το id του evaluator και το id του job_position.

Δηλώνονται μεταβλητές μέσα στο σώμα της procedure την total, id, inter, rep και cap και τους αναθέτονται τις τιμές των γνωρισμάτων : το άθροισμα των τιμών των γνωρισμάτων interview , report και capability ,το evalid ,το interview , report ,capability αντίστοιχα κάνοντας inner join από τον πίνακα evaluation, στον evaluator, στον πίνακα announces και στον job_position με limit 1 .

Μετά μέσω της if ελέγχεται αν οι βαθμολογίες είναι μεγαλύτερες του μηδενός ή όχι .Αν είναι τότε το total αποθηκεύεται στο finalgrade αλλιώς εκτυπώνεται ένδειξη “Not all phases completed”.

3.3.3

Δημιουργείται `procedure evaluationStatus` η οποία δέχεται ως είσοδο ένα `jobid`.

Η `store procedure evaluationStatus` ενημερώνει για την κατάσταση των αξιολογήσεων για μια θέση εργασίας.

Δημιουργούνται δυο μεταβλητές l και k στις οποίες αποθηκεύονται ο αριθμός αιτήσεων που έχουν γίνει και ο αριθμός αποτελεσμάτων που έχουν οριστικοποιηθεί αντίστοιχα (για το συγκεκριμένη θέση εργασίας που δόθηκε στην είσοδο). Αν λοιπόν $l=k$ τότε όλες οι αιτήσεις έχουν βαθμολογηθεί και εμφανίζονται ο βαθμός και το όνομα του υπάλληλου. Αν $l>k$ τότε εκκρεμούν βαθμολογίες εμφανίζονται όσες έχουν ολοκληρωθεί. Ενώ αν $l=0$ δεν υπάρχουν αιτήσεις.

Σε κάθε περίπτωση εμφανίζεται μήνυμα το οποίο ενημερώνει για την κατάσταση των αξιολογήσεων.

3.3.4

Στο πλαίσιο δημιουργίας του GUI προέκυψε μια άλλη `procedure` τη οποία ονομάσαμε `mesos`.

Η `procedure` αυτή παίρνει ως είσοδο ένα `username` ενός `evaluator` και εμφανίζει το μέσος όρο των βαθμών που έχει βάλει γενικά ο αξιολογητής που έδωσε ο χρήστης.

3.4 Triggers

3.4.1

Επιλέγεται μετά την εισαγωγή στον κατάλληλο πίνακα :

```
AFTER INSERT ON job_position
```

```
AFTER INSERT ON employe
```

```
AFTER INSERT ON requests
```

Δημιουργείται μια μεταβλητή στην οποία ορίζεται η συνάρτηση ημερομηνίας, έτσι ώστε να μας αναφέρει ακριβής ημερομηνία μετά από κάθε ενέργεια.

```
DECLARE currDate DATETIME;  
SET currDate=CURRENT_TIMESTAMP();
```

Τέλος, εισάγονται οι τιμές που πρέπει να μπουν στον πίνακα log έτσι ώστε να μας αναφέρει τις ενέργειες που έγιναν.

```
INSERT INTO log VALUES(DEFAULT, 'ABernard', currDate, 'insert', '1', 'job_position');
```

Για τις υπόλοιπες ενέργειες (update, delete) ο κώδικας είναι αντίστοιχος. Το μόνο που αλλάζει είναι τα ονόματα των πινάκων και οι ενέργειες.

3.4.2

Το trigger companyUpdate αποτρέπει την αλλαγή ΑΦΜ, ονόματος και ΔΟΥ από τον χρήστη. Κάθε φορά που ο χρήστης πάει να κάνει update ένα company, η ενημέρωση γίνεται εκτός όμως από τα τρία αυτά γνωρίσματα. Το trigger λοιπόν γίνεται before update on company και με την χρήση new και old. Συγκεκριμένα κάθε φορά κάνει το NEW. να ισούται με το OLD. για καθένα από τα παραπάνω.

3.4.3

Το trigger changeprofile ενεργοποιείται όταν κάποιος χρήστης εκτός του administrator προσπαθεί να κάνει κάποια αλλαγή στα στοιχεία του προφίλ του. Μέσω μιας If ελέγχεται αν το username είναι το ABernard (το username του administrator της βάσης μας) αν δεν είναι αυτό τότε ορίζονται τα καινούρια username,name,surname ,email και sing_in_date ίσα με τα προϋπάρχοντα.

4. Υλοποίηση GUI με την βοήθεια της Python

ΜΕΡΟΣ Β

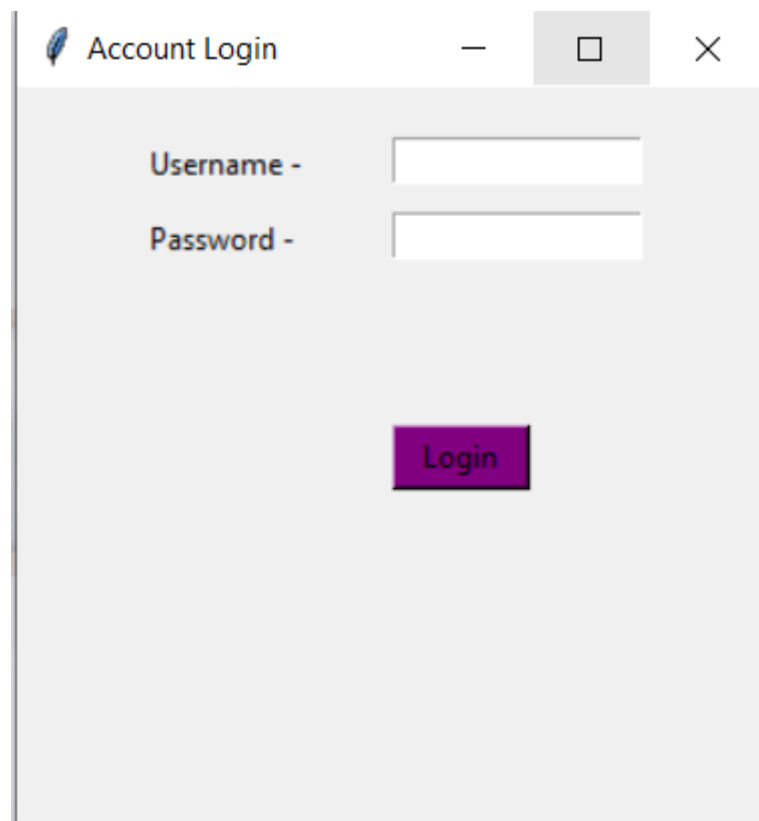
Το user interface της βάσης μας έχει υλοποιηθεί με python ,MYSQL python και με την βοήθεια της βιβλιοθήκης tkinter.

Αποτελείται όπως ζητείται από 4 υπό διεπαφές για καθένα από τα είδη χρήστη.

Η εκτέλεση του έγινε σε jupyter notebook.

LOGIN IN SCREEN

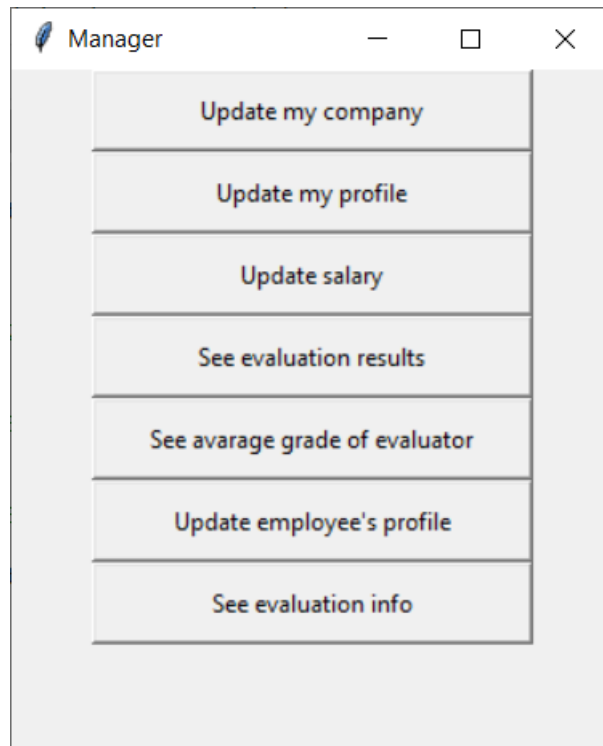
Ο χρήστης όταν συνδέεται στη βάση βλέπει το παρακάτω παράθυρο όπου συμπληρώνοντας το username και το κωδικό αυτόματα θα του εμφανιστεί το αντίστοιχο interface της κατηγορίας χρήστη που ανήκει.



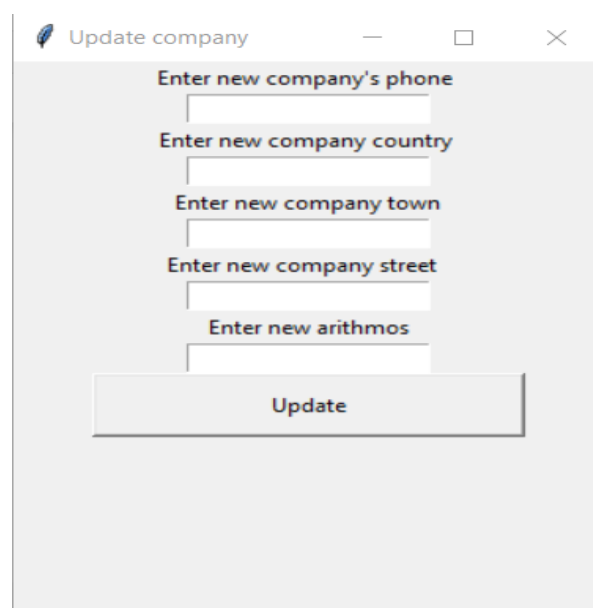
The image shows a window titled "Account Login" with a feather icon on the left and standard window controls (minimize, maximize, close) on the right. The window has a light gray background. It contains two labels, "Username -" and "Password -", each followed by a white text input field. Below these fields is a purple button with the text "Login" in white. The window is centered on the screen.

MANAGER

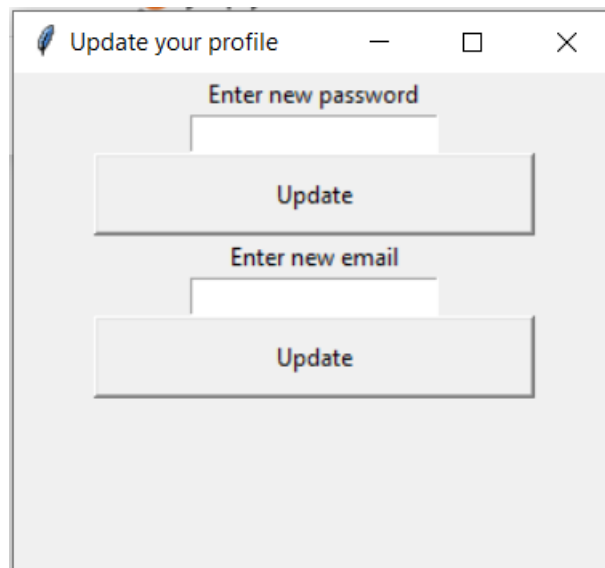
Ο manager όταν συνδέεται του εμφανίζεται μια οθόνη γραφεί Manager και η οποία αποτελείται από 7 κουμπιά.



Το κουμπί update company εμφανίζει ένα παράθυρο στο οποίο ο manager μπορεί να ενημερώσει στα στοιχεία της εταιρίας που διοικεί.

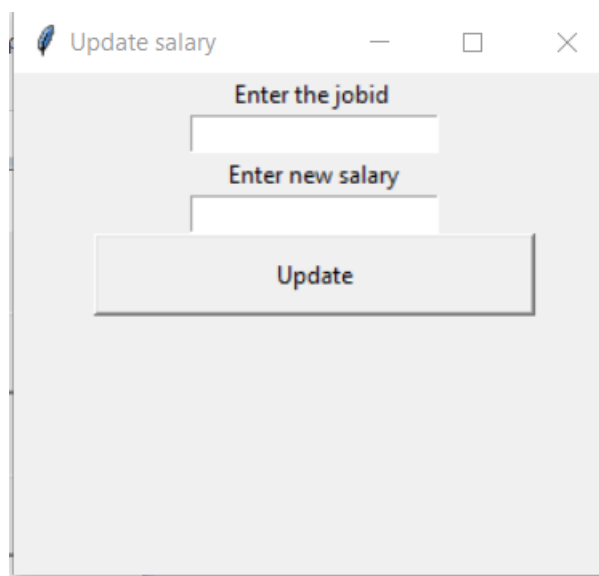
A screenshot of a web application window titled "Update company". The window contains a form with five input fields, each preceded by a label: "Enter new company's phone", "Enter new company country", "Enter new company town", "Enter new company street", and "Enter new arithmos". Below these fields is a large rectangular button labeled "Update". The window has standard OS controls (minimize, maximize, close) in the top right corner.

Με το update profile εμφανίζονται τα παρακάτω όπου μπορεί να επιλέξει να αλλάξει το κωδικό του ή το email ή και τα δυο αν επιθυμεί.



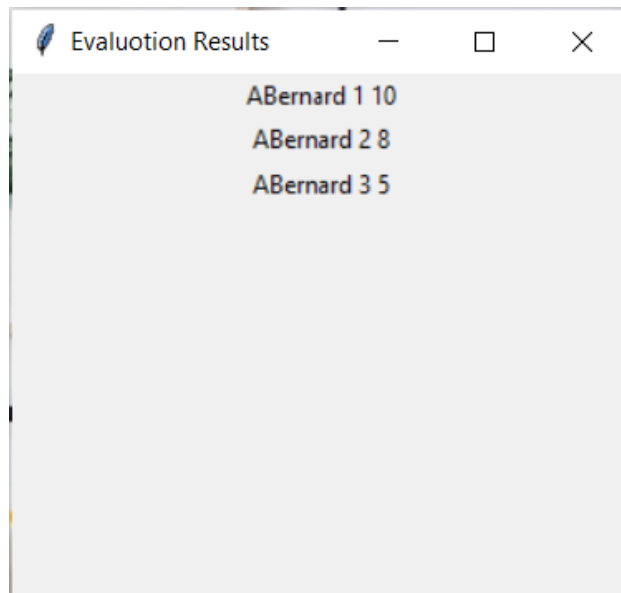
The screenshot shows a window titled "Update your profile" with a feather icon. It contains two input sections. The first section is labeled "Enter new password" and has a text input field followed by an "Update" button. The second section is labeled "Enter new email" and also has a text input field followed by an "Update" button.

Εδώ ο manager μπορεί να αλλάξει το μισθό μιας θέσης.

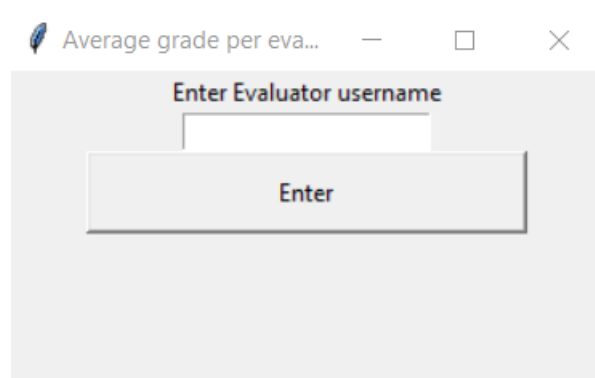


The screenshot shows a window titled "Update salary" with a feather icon. It contains two input fields: "Enter the jobid" and "Enter new salary", each with a corresponding text input field. Below these fields is a single "Update" button.

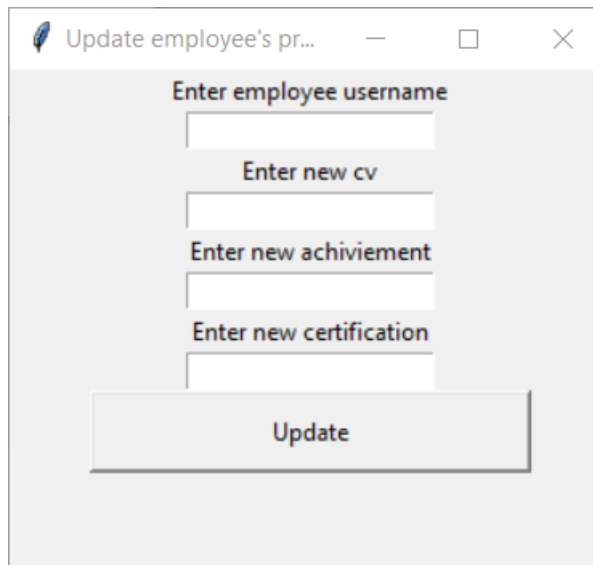
Το κουμπί see evaluation results εμφανίζει στη οθόνη του μάνατζερ της οριστικοποιημένες αξιολογήσεις. Για να βρίσκονται τα αποτελέσματα στο πίνακα evaluationresults στο txt με την procedure sumofevaluation βρίσκονται μερικές γραμμές κώδικας που πρέπει να εκτελεσθεί μετά την εισαγωγή του στη βάση.



Σε αυτό το σημείο ο μπορεί να βάλει το username ενός αξιολογητή και να δει το μέσο ορό βαθμολογίας του κρίνοντας αν ένας αξιολογητής είναι αυστηρός ή όχι στις αξιολογήσεις του.

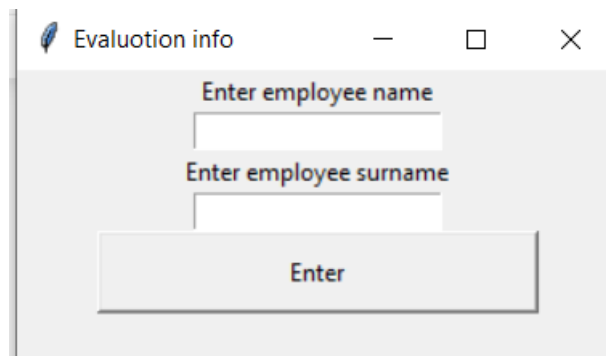


Με το κουμπί update employee's info ο manager έχει την δυνατότητα να αλλάξει τα στοιχεία ενός υπάλληλου.



A screenshot of a software dialog box titled "Update employee's pr...". The dialog box has a standard Windows-style title bar with a feather icon, a minus button, a maximize button, and a close button. The main content area is light gray and contains four text input fields stacked vertically, each with a label above it: "Enter employee username", "Enter new cv", "Enter new achivement", and "Enter new certification". Below these fields is a large, light gray button labeled "Update".

Ο manager ανοίγοντας αυτό το παράθυρο μπορεί να βάλει το όνομα και το επίθετο ενός υπάλληλου και να δει όλες οι αιτήσεις του, οι τελικές αξιολογήσεις του που έχουν ολοκληρωθεί και το όνομα και επώνυμο του αντίστοιχου αξιολογητή.



A screenshot of a software dialog box titled "Evaluation info". The dialog box has a standard Windows-style title bar with a feather icon, a minus button, a maximize button, and a close button. The main content area is light gray and contains two text input fields stacked vertically, each with a label above it: "Enter employee name" and "Enter employee surname". Below these fields is a large, light gray button labeled "Enter".

Κώδικας manager

Συνάρτηση manager_screen(): (): Toplevel στο root .Δημιουργούνται 7 κουμπιά που εμφανίζονται στη οθόνη με την χρήση .pack() τα οποία καλούν την αντιστοιχεί συνάρτηση.

Συνάρτηση update_mpasswd():Γίνεται σύνδεση με την βάση και update password του manager που συνδέθηκε .

Συνάρτηση update_memail():Γίνεται σύνδεση με την βάση και update email του manager που συνδέθηκε.

Συνάρτηση update_mprofile():Toplevel στο manager screen δημιουργούνται δυο κουμπιά και δυο entries και καθένα από τα κουμπιά καλούν μια από τις δύο παραπάνω συναρτήσεις.

Συνάρτηση update_salary():Toplevel στο manager screen δημιουργείται ένα κουμπί και δυο entries. Το κουμπί καλεί την update_s().

Συνάρτηση update_s():Γίνεται σύνδεση με την βάση και update το salary της θέσης που δήλωσε ο manager που συνδέθηκε.

Συνάρτηση update_company():Toplevel στο manager screen δημιουργείται ένα κουμπί και πέντε entries για να συμπληρωσει ο χρήστης. Το κουμπί καλεί την company_update().

Συνάρτηση company_update():Γίνεται σύνδεση με την βάση και update το company της του συγκεκριμένου manager.

Συνάρτηση see_finalresults():Toplevel στο manager screen γίνεται σύνδεση με την βάση και εμφανίζονται όλες οι τελειοποιημένες αξιολογήσεις.

Συνάρτηση procedure_mesos():Γίνεται σύνδεση με την βάση και καλείται η procedure mesos για τον αξιολογητή που ζητήθηκε από το manager.

Συνάρτηση mesos_evaluator(): Toplevel στο manager screen δημιουργείται ένα κουμπί και ένα entry για να συμπληρώσει ο manager το username ενός αξιολογητή. Το κουμπί καλεί την procedure_mesos().

Συνάρτηση update_eprofile():Toplevel στο manager screen δημιουργείται ένα κουμπί και τέσσερα entries και το κουμπί καλεί την update_empl() για να γίνει το update.

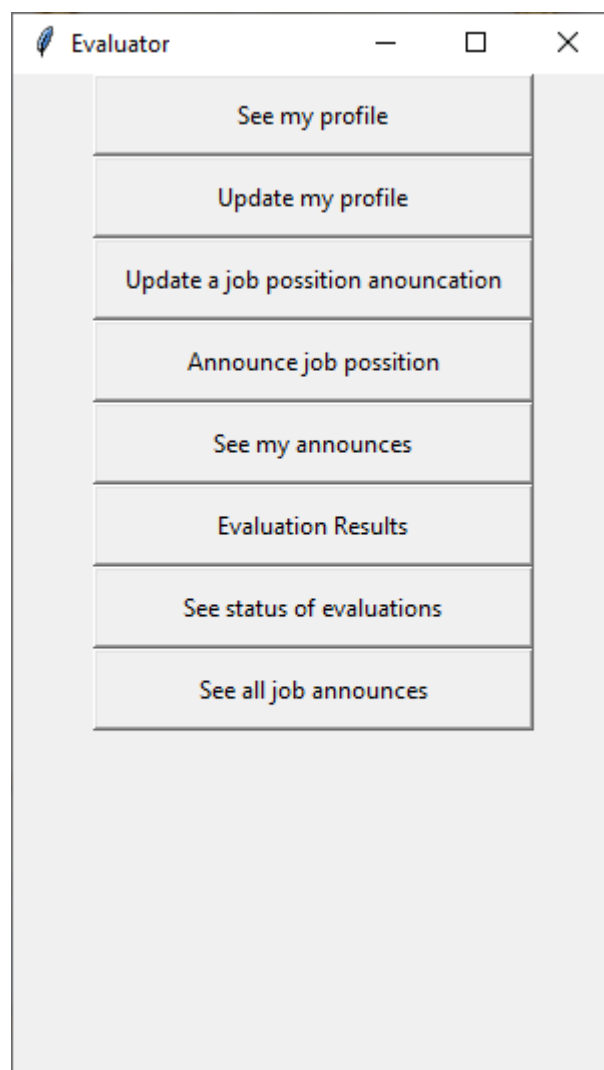
Συνάρτηση update_empl():Γίνεται σύνδεση με την βάση και update στα γνωρίσματα του employee που επιθυμεί να τροποποιήσει ο manager

Συνάρτηση procedure_one():Toplevel στο manager screen δημιουργείται ένα κουμπί και δυο entries για να συμπληρώσει ο χρήστης όνομα και επώνυμο υπάλληλου .Το κουμπί καλεί την call_prone().

Συνάρτηση call_prone():Γίνεται σύνδεση με την βάση και καλείται η procedure employee για τον υπάλληλο που επιθυμεί ο manager

EVALUATOR

Ο evaluator όταν συνδέεται του εμφανίζεται μια οθόνη γραφεί Evaluator και η οποία αποτελείται από 8 κουμπιά.



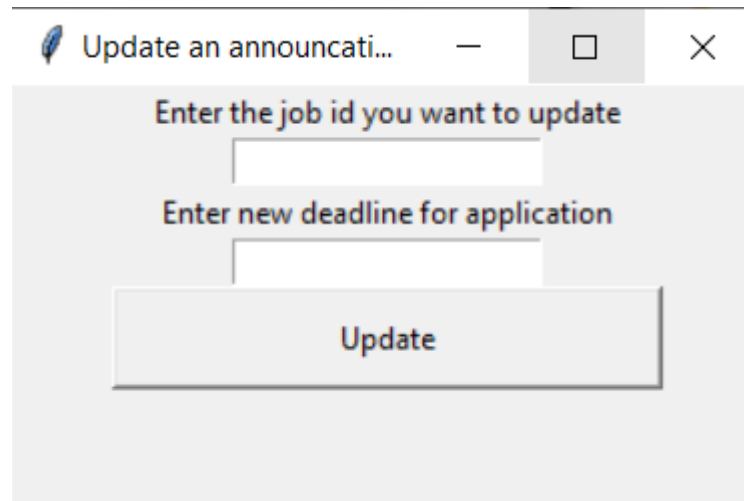
Το κουμπί see file εμφανίζει στη οθόνη τα στοιχεία του.



Πατώντας update my profile του δίνεται η δυνατότητα να αλλάξει τα στοιχεία του στο σύστημα

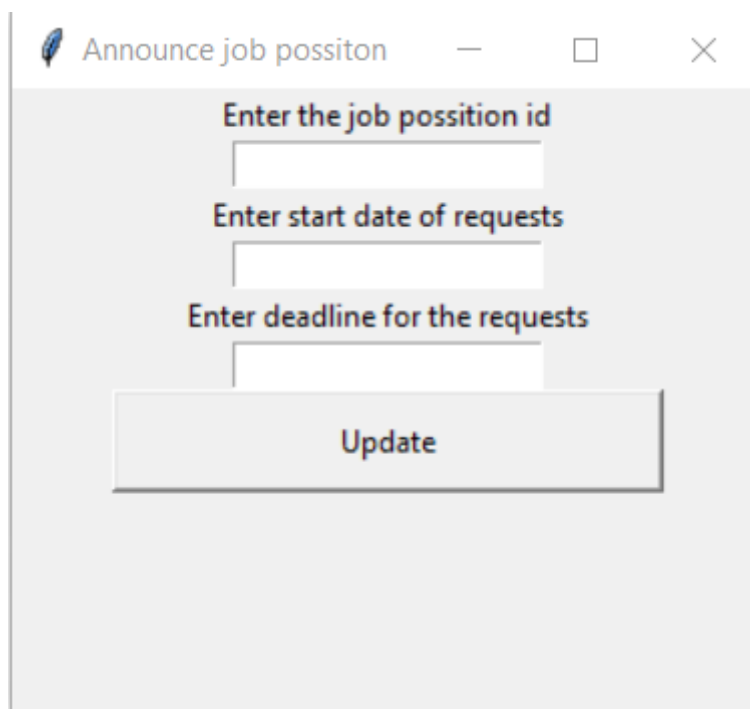
A screenshot of a window titled "Update my profile" with a feather icon. The window contains four input fields with labels: "Enter new name", "Enter new surname", "Enter new email", and "Enter new password". Below these fields is a large "Update" button. The window has standard minimize, maximize, and close buttons.

Αν επιθυμεί να τροποποιήσει μια ανακοίνωση για μια θέση εργασίας πατά το 3 κουμπι και όπως φαίνεται παρακάτω βάζοντας το κωδικό της θέσης εργασίας μπορεί να αλλάξει το deadline.



The screenshot shows a dialog box titled "Update an announcement...". It contains two input fields: "Enter the job id you want to update" and "Enter new deadline for application". Below these fields is a large button labeled "Update".

Αντίθετα αν θέλει να βγάλει μια προκήρυξη για μια θέση τότε επιλεγεί το κουμπί announce job position και συμπληρώνει την παρακάτω φόρμα.



The screenshot shows a dialog box titled "Announce job position". It contains three input fields: "Enter the job position id", "Enter start date of requests", and "Enter deadline for the requests". Below these fields is a large button labeled "Update".

Το κουμπί see my announnces εμφανίζει στη οθόνη του αξιολογητή όλες τις ανακοινώσεις που έχει βγάλει ο ίδιος.

My announces

AMartin 1 2000-02-13 2021-06-06

AMartin 4 None None

AMartin 5 None None

Evaluation Status

Enter the jobid you want to see status for

5

Enter

0

0

{{ORISTIKOPOIMENOI PINAKES}}

0

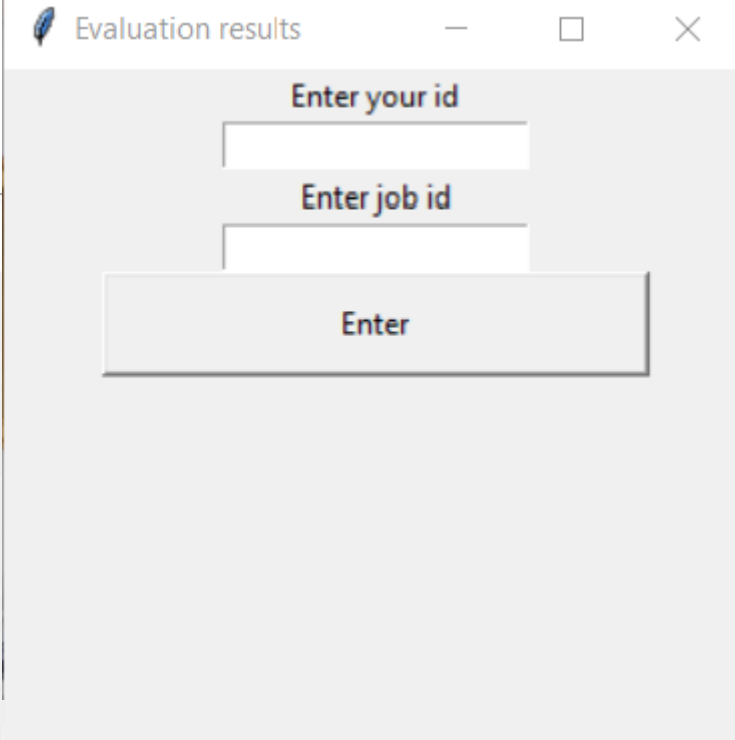
0

{{ORISTIKOPOIMENOI PINAKES}}

Στο σημείο αυτό μπορεί να δει την κατάσταση μιας ανακοίνωσης του δηλαδή αν όλες οι αιτήσεις έχουν αξιολογηθεί και ανάλογα εμφανίζεται αντίστοιχο μήνυμα.

Εδώ μπορεί να δει το βαθμό των αξιολογήσεων του.

Με το
job

A screenshot of a software window titled "Evaluation results". The window has a standard title bar with a minimize button, a maximize button (disabled), and a close button. The main content area is light gray and contains three stacked input fields. The first field is labeled "Enter your id", the second is labeled "Enter job id", and the third is a button labeled "Enter".

Evaluation results

Enter your id

Enter job id

Enter

κουμπί see all
announces

μπορεί να δει και ανακοινώσεις άλλων αξιολογικών

Κώδικας evaluator:

Συνάρτηση evaluator_screen(): Toplevel στο root .Δημιουργούνται 8 κουμπιά που εμφανίζονται στη οθόνη με την χρήση .pack() τα οποία καλούν την αντιστοιχεί συνάρτηση.

Συνάρτηση evaluator_profile(): Γίνεται σύνδεση με την βάση και select fetch all των στοιχείων του evaluator που συνδέθηκε.

Συνάρτηση eval_update(): Toplevel στο evaluator screen έχει 4 entries και 1 κουμπί το οποίο συνδέεται με την συνάρτηση update_eval()

Συνάρτηση update_eval(): Toplevel στο evaluator screen γίνεται σύνδεση με την βάση και update name ,surname,email,password του evaluator που συνδέθηκε .

Συνάρτηση update_antable(): Γίνεται σύνδεση με την βάση και update του closedate για την θέση εργασίας που επέλεξε ο evaluator που συνδέθηκε.

Συνάρτηση update_announces(): Toplevel στο evaluator screen έχει 2 entries και 1 κουμπί το οποίο συνδέεται με την συνάρτηση update_antable()

Συνάρτηση insert_announces(): Γίνεται σύνδεση με την βάση και insert στο announces.

Συνάρτηση announce_job():() Toplevel στο evaluator screen έχει 3entries και 1 κουμπί το οποίο συνδέεται με την συνάρτηση insert_announces()

Συνάρτηση see_my_announces():Γίνεται σύνδεση με την βάση και select fetch all από τον πίνακα announces του evaluator που συνδέθηκε.

Συνάρτηση call_pro_two():Γίνεται σύνδεση με την βάση και call procedure sumofevaluation.

Συνάρτηση pro_two():Toplevel στο evaluator screen δημιουργείται ένα κουμπί και δυο entries για να συμπληρώσει ο χρήστης.Το κουμπί καλεί την call_pro_two()

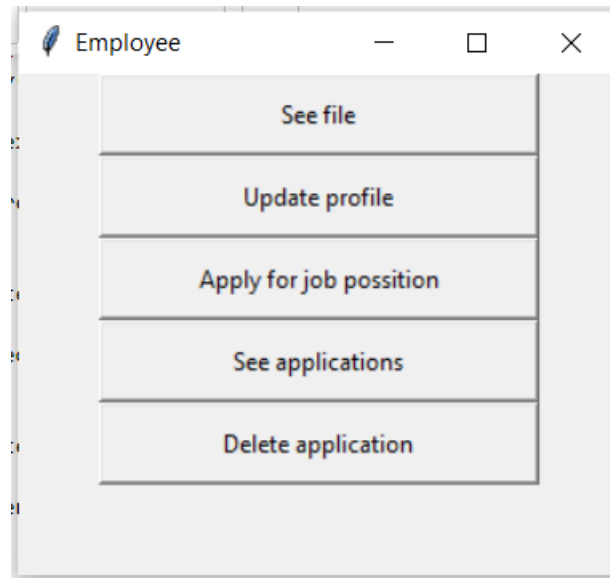
Συνάρτηση call_pr_three():Γίνεται σύνδεση με την βάση και call procedure evaluationStatus.

Συνάρτηση pro_three():Toplevel στο evaluator screen δημιουργείται ένα κουμπί και ένα entry για να συμπληρώσει ο χρήστης .Το κουμπί καλεί την call_pr_three()

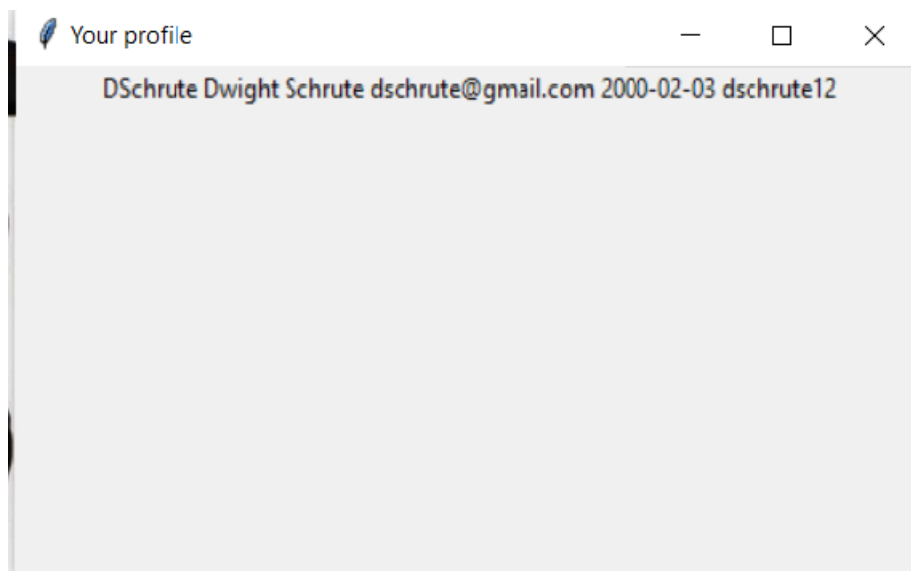
Συνάρτηση see_all_anounces():():Toplevel στο evaluator screen γίνεται σύνδεση με την βάση και select fetch all των στοιχείων που περιεχει ο πίνακας announces.

EMPLOYEE

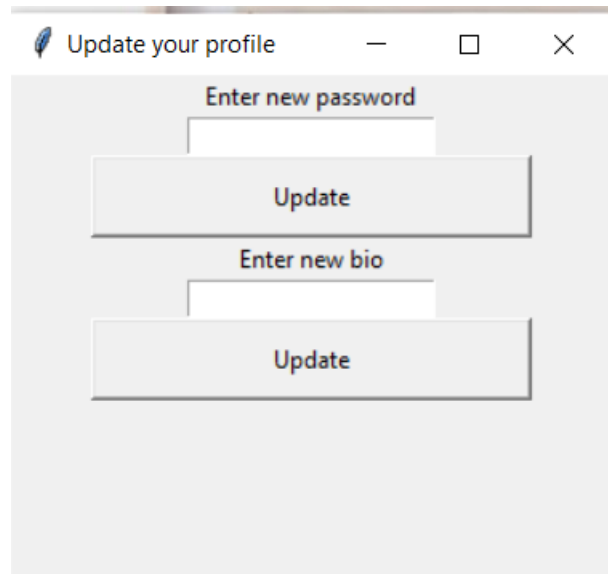
Ο υπάλληλος όταν συνδέεται του εμφανίζεται μια οθόνη γραφεί Employee και η οποία αποτελείται από 5 κουμπιά.



Το see file ανοίγει το παρακάτω παράθυρο στο οποίο φαίνονται τα στοιχεία του υπαλλήλου.

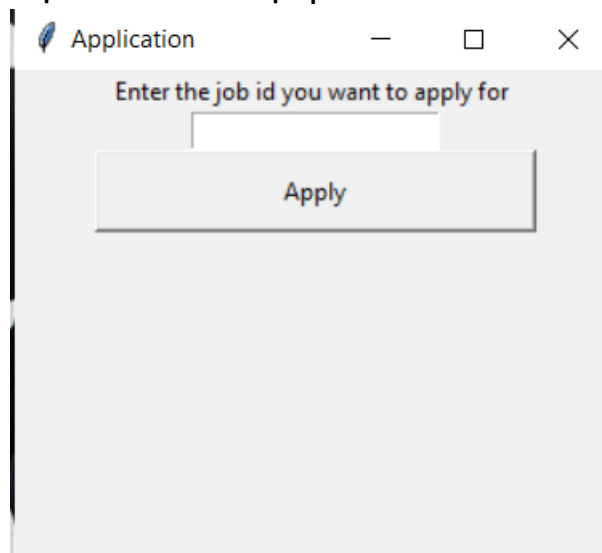


Το update your profile ανοίγει το παρακάτω παράθυρο στο οποίο υπάρχουν 2 entries για να βάλει ο υπάλληλος το καινούργιο του password και το καινούργιο του bio υπάρχουν 2 κουμπιά ώστε να μπορεί να τροποποιήσει μονό ένα αν επιθυμεί.



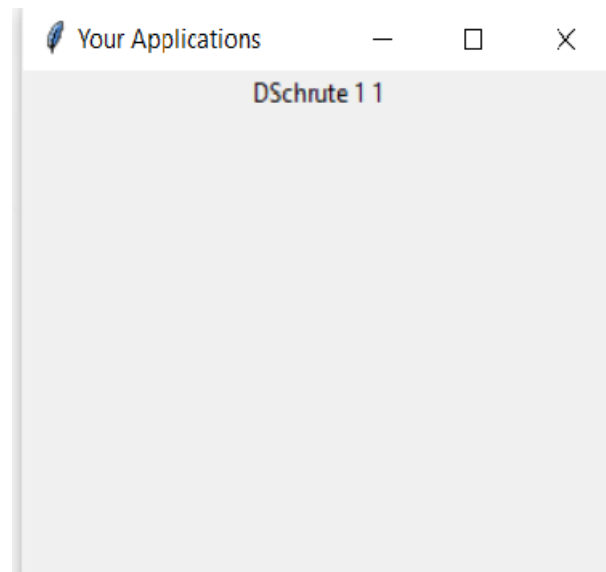
The screenshot shows a window titled "Update your profile" with standard window controls (minimize, maximize, close). Inside the window, there are two sections. The first section is labeled "Enter new password" and contains a text input field followed by an "Update" button. The second section is labeled "Enter new bio" and also contains a text input field followed by an "Update" button.

Το apply for job position ανοίγει το παρακάτω παράθυρο στο οποίο υπάρχουν 1 entry για να βάλει ο υπάλληλος το id της θέσης για την οποία θέλει να δηλώσει ενδιαφέρον.

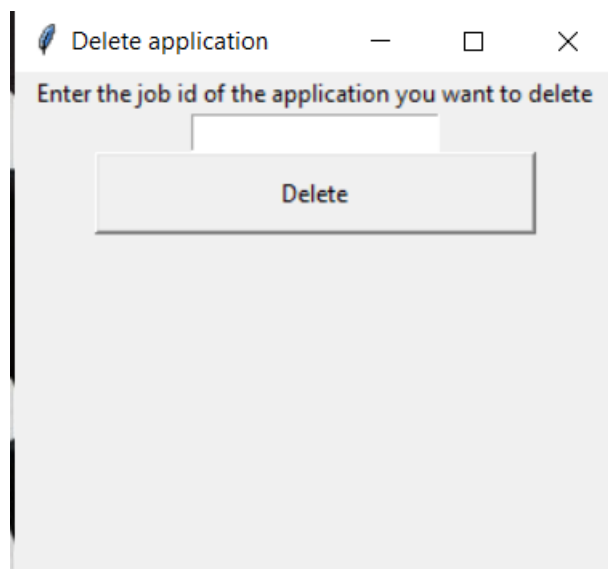


The screenshot shows a window titled "Application" with standard window controls (minimize, maximize, close). Inside the window, there is a single section labeled "Enter the job id you want to apply for" which contains a text input field followed by an "Apply" button.

Το see applications ανοίγει το παρακάτω παράθυρο στο οποίο φαίνονται τα στοιχεία της αίτησης.



Το delete application ανοίγει το παρακάτω παράθυρο στο οποίο υπάρχουν ένα entry για να βάλει ο υπάλληλος το id της θέσης για την οποία θέλει να διαγράψει τη αίτηση που έχει κάνει .



Κώδικας employee

Συνάρτηση employee_screen(): Toplevel στο root .Δημιουργούνται 5 κουμπιά που εμφανίζονται στη οθόνη με την χρήση .pack() τα οποία καλούν την αντιστοιχεί συνάρτηση.

Συνάρτηση see_file(): Toplevel στο employee screen γίνεται σύνδεση με την βάση και select fetch all των στοιχείων του employee που συνδέθηκε.

Συνάρτηση update_passw(): Toplevel στο employee screen γίνεται σύνδεση με την βάση και update password του employee που συνδέθηκε .

Συνάρτηση update_bio(): Toplevel στο employee screen γίνεται σύνδεση με την βάση και update cv του employee που συνδέθηκε.

Συνάρτηση update_profile(): Toplevel στο employee screen έχει 2 entries και 2 κουμπιά το καθένα συνδέεται με μια από τις 2 παραπάνω συναρτήσεις .

Συνάρτηση applyforjob(): Toplevel στο employee screen έχει ένα και ένα κουμπί δηλώνει την θέση για την οποία κάνει αίτηση και το κουμπί καλεί την συνάρτηση insert_requests():.

Συνάρτηση insert_requests(): Toplevel στο employee screen γίνεται σύνδεση με την βάση και insert στο πίνακα request τη νέα αίτηση που έκανε ο συγκεκριμένος employee.

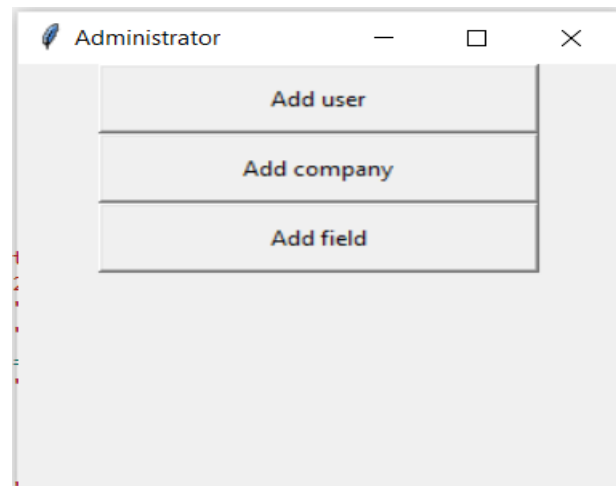
Συνάρτηση see_applications(): Toplevel στο employee screen γίνεται σύνδεση με την βάση και select fetch all των requests για θέση εργασίας του employee που συνδέθηκε.

Συνάρτηση delete_application(): Toplevel στο employee screen έχει ένα entries και ένα κουμπί το οποίο συνδέεται με τη συνάρτηση delete_a().

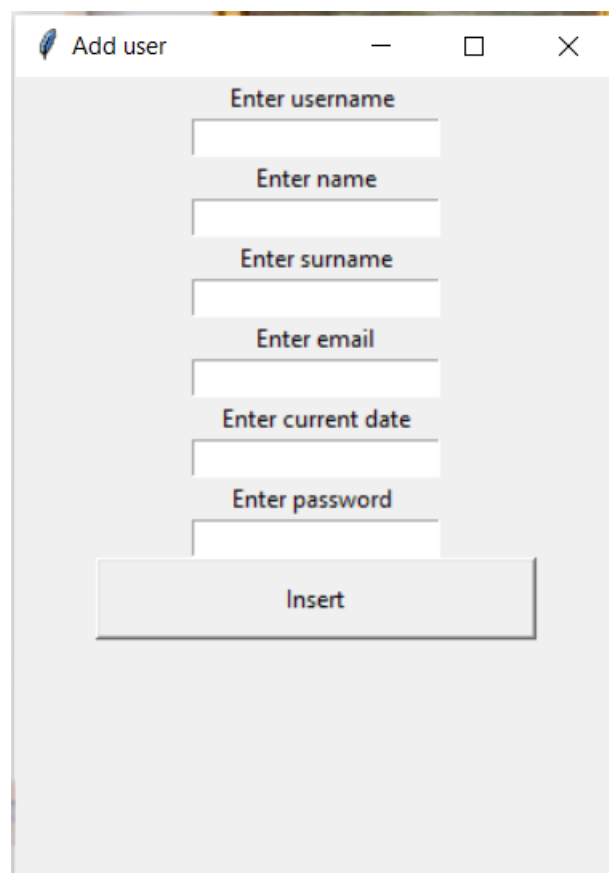
Συνάρτηση delete_a():Τοplevel στο employee screen γίνεται σύνδεση με την βάση και delete από τον πίνακα requests για τον θέση που επιθυμεί ο χρήστης

ADMINISTRATOR

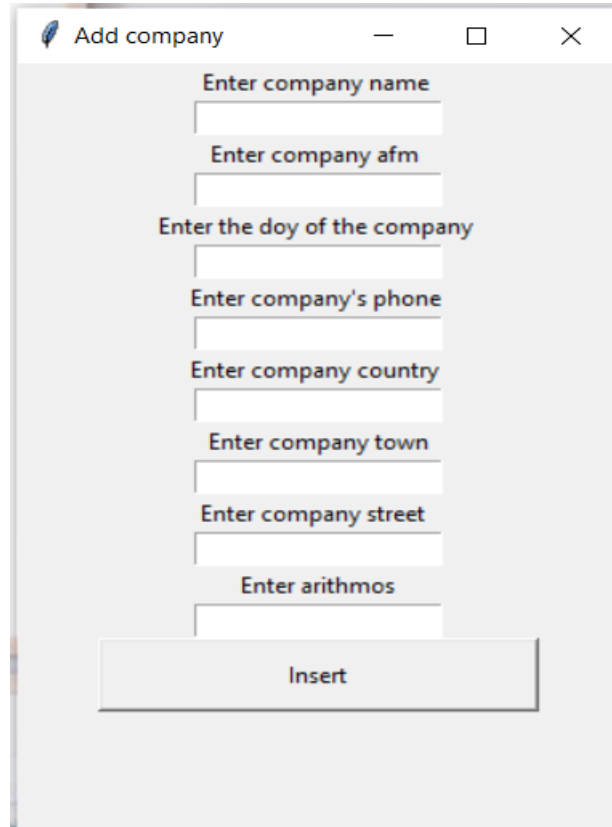
Ο διαχειριστής όταν συνδέεται του εμφανίζεται μια οθόνη γραφεί Administrator και η οποία αποτελείται από 3 κουμπιά.



Πρώτο κουμπί είναι το add user αν το πατήσει του εμφανίζεται μια φόρμα για να συμπληρώσει τα στοιχεία του χρήστη που θέλει να προσθέσει και πατώντας το κουμπί Insert ο νέος χρήστης προστίθενται στη βάση

A screenshot of a software window titled "Add user". The window has a standard Windows-style title bar with a minimize button, a maximize button, and a close button. The main content area is light gray and contains a form with six input fields stacked vertically. Each input field has a label above it: "Enter username", "Enter name", "Enter surname", "Enter email", "Enter current date", and "Enter password". Below the input fields is a large rectangular button labeled "Insert".

Δεύτερο κουμπί είναι το add company αν το πατήσει του εμφανίζεται μια φόρμα για να συμπληρώσει τα στοιχεία της νέας εταιρίας που θέλει να προσθέσει και πατώντας το κουμπί Insert η νέα εταιρία προστίθενται στη βάση.



A screenshot of a software window titled "Add company". The window contains a vertical stack of text input fields with the following labels: "Enter company name", "Enter company afm", "Enter the doy of the company", "Enter company's phone", "Enter company country", "Enter company town", "Enter company street", and "Enter arithmos". Below these fields is a large rectangular button labeled "Insert".

Το τρίτο κουμπί είναι το add field αν το πατήσει του εμφανίζεται μια φόρμα για να συμπληρώσει τα στοιχεία του νέου τομέα που θέλει να προσθέσει και πατώντας το κουμπί Insert ο νέος τομέας προστίθενται στη βάση.



A screenshot of a software window titled "Add field". The window contains a vertical stack of text input fields with the following labels: "Enter field title", "Enter field description", and "Enter jobid for the field". Below these fields is a large rectangular button labeled "Insert".

Κώδικας Administrator

Συνάρτηση administrator_screen(): Toplevel στο root

.Δημιουργούνται 3 κουμπιά που εμφανίζονται στη οθόνη με την χρήση .pack() τα οποία καλούν την αντιστοιχεί συνάρτηση.

Συνάρτηση add_field(): Toplevel στο administrator screen έχει 3 labels και 3 entries και ένα κουμπί που ενεργοποιεί την συνάρτηση field insert.

Συνάρτηση field_insert():Σύνδεση με τη βάση και μέσω mysql γίνεται insert στο πίνακα field το οποίο εκτελεί ο cursor.

Συνάρτηση add_company():Toplevel στο administrator screen έχει 8 labels και 8 entries και ένα κουμπί που ενεργοποιεί την συνάρτηση company insert.

Συνάρτηση company_insert():Σύνδεση με τη βάση και μέσω mysql γίνεται insert στο πίνακα company το οποίο εκτελεί ο cursor.

Συνάρτηση add_user():Toplevel στο administrator screen έχει 6 labels και 6 entries και ένα κουμπί που ενεργοποιεί την συνάρτηση field insert.

Συνάρτηση user_insert():Σύνδεση με τη βάση και μέσω mysql γίνεται insert στο πίνακα user το οποίο εκτελεί ο cursor

Το τελευταίο ερώτημα του δευτέρου μέρους δεν έχει υλοποιηθεί.

SQL

```
create database StaffEvaluation;
```

```
use StaffEvaluation;
```

```
DROP TABLE IF EXISTS user;
```

```
CREATE TABLE user(  
    username VARCHAR(20) NOT NULL DEFAULT 'unknown',  
    name VARCHAR(10) NOT NULL DEFAULT 'unknown',  
    surname VARCHAR(10) NOT NULL DEFAULT 'unknown',  
    email VARCHAR(20) NOT NULL DEFAULT 'unknown',  
    sign_in_date DATE NOT NULL,  
    password VARCHAR(10) NOT NULL,  
    PRIMARY KEY(username)  
);
```

```
CREATE TABLE log(  
    logid INT(10) NOT NULL AUTO_INCREMENT,  
    username VARCHAR(20) NOT NULL DEFAULT 'unknown',  
    actiondatetime DATETIME NOT NULL,  
    action ENUM ('insert','update','delete'),  
    success ENUM ('0','1'),  
    tablename VARCHAR(20) NOT NULL DEFAULT 'unknown',  
    PRIMARY KEY(logid),  
    CONSTRAINT LOGUS  
    FOREIGN KEY(username) REFERENCES user(username)  
    ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
DROP TABLE IF EXISTS company;
```

```
CREATE TABLE company(  
    
```

```
companyName VARCHAR(20) NOT NULL DEFAULT 'unknown',
afm INT(10) NOT NULL DEFAULT '0',
DOY VARCHAR(15) NOT NULL DEFAULT 'unknown',
phonenummer INT(10) NOT NULL DEFAULT '0',
country VARCHAR(15) NOT NULL DEFAULT 'unknown',
town VARCHAR(15) NOT NULL DEFAULT 'unknown',
street VARCHAR(15) NOT NULL DEFAULT 'unknown',
arithmos INT(10) NOT NULL DEFAULT '0',
m_username varchar(20) NOT NULL,
PRIMARY KEY(afm)
);
```

```
DROP TABLE IF EXISTS employe;
```

```
CREATE TABLE employe (
  em_username VARCHAR(20) NOT NULL REFERENCES user(username),
  certifications VARCHAR(25) NOT NULL DEFAULT 'unknown',
  am INT(10) NOT NULL DEFAULT '0',
  cv TEXT NOT NULL,
  companyAfm INT(10) NOT NULL DEFAULT '0',
  workyears INT(10) NOT NULL DEFAULT '0',
  achievement VARCHAR(25) NOT NULL DEFAULT 'not_found',
  languages VARCHAR(25) NOT NULL DEFAULT 'unknown',
  PRIMARY KEY (em_username),
  CONSTRAINT COMAFM
  FOREIGN KEY(companyAfm) REFERENCES company(afm)
  ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
DROP TABLE IF EXISTS administrator;
```

```
CREATE TABLE administrator(  
    ad_username VARCHAR(20) NOT NULL REFERENCES user(username),  
    PRIMARY KEY(ad_username)  
);
```

```
DROP TABLE IF EXISTS evaluator;
```

```
CREATE TABLE evaluator(  
    ev_username VARCHAR(20) NOT NULL REFERENCES user(username),  
    ev_code INT(10) NOT NULL DEFAULT '0',  
    company_afm INT(10) NOT NULL DEFAULT '0',  
    PRIMARY KEY(ev_username)  
);
```

```
DROP TABLE IF EXISTS manager;
```

```
CREATE TABLE manager (  
    m_username varchar(20) NOT NULL REFERENCES user(username),  
    workyears int(2) NOT NULL DEFAULT 0,  
    companyAfm bigint(11) NOT NULL DEFAULT 0,  
    PRIMARY KEY (m_username)  
);
```

```
DROP TABLE IF EXISTS updates;
```

```
CREATE TABLE updates(  
  m_username VARCHAR(20) NOT NULL,  
  em_username VARCHAR(20) NOT NULL,  
  PRIMARY KEY(m_username,em_username),  
  CONSTRAINT UPDM  
  FOREIGN KEY (m_username) REFERENCES manager(m_username)  
  ON DELETE CASCADE ON UPDATE CASCADE,  
  CONSTRAINT UPDEM  
  FOREIGN KEY (em_username) REFERENCES employe(em_username)  
  ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
DROP TABLE IF EXISTS works;
```

```
CREATE TABLE works(  
  username VARCHAR(20) NOT NULL,  
  afm INT(10) NOT NULL DEFAULT '0',  
  PRIMARY KEY(username,afm),  
  CONSTRAINT WORK  
  FOREIGN KEY(username) REFERENCES user(username)  
  ON DELETE CASCADE ON UPDATE CASCADE,  
  CONSTRAINT WORKAFM  
  FOREIGN KEY(afm) REFERENCES company(afm)  
  ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
DROP TABLE IF EXISTS degree;
```

```
CREATE TABLE degree(  
  dttitle VARCHAR(50) NOT NULL ,  
  idrima VARCHAR(40) NOT NULL,
```

```
etosktisis INT(4) NOT NULL,  
bathmos ENUM('LYKEIO','UNIV','MASTER','PHD'),  
username VARCHAR(20) NOT NULL,  
PRIMARY KEY(dttitle,idrima),  
CONSTRAINT DEG  
FOREIGN KEY(username) REFERENCES employe(em_username)  
ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
DROP TABLE IF EXISTS project;
```

```
CREATE TABLE project(  
aa INT(10) NOT NULL AUTO_INCREMENT,  
url VARCHAR(60) NOT NULL,  
perigrafi TEXT NOT NULL,  
username VARCHAR(20) NOT NULL,  
PRIMARY KEY(aa),  
CONSTRAINT PRJE  
FOREIGN KEY(username) REFERENCES employe(em_username)  
ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
DROP TABLE IF EXISTS job_position;
```

```
CREATE TABLE job_position(  
jobid INT(10) NOT NULL,  
jobedra VARCHAR(20) NOT NULL DEFAULT 'unknown',  
salary INT(10) NOT NULL DEFAULT '0',  
jobtitle VARCHAR(50) NOT NULL DEFAULT 'unknown',  
PRIMARY KEY(jobid)  
);
```

DROP TABLE IF EXISTS requests;

```
CREATE TABLE requests(  
    em_username VARCHAR(20) NOT NULL,  
    jobid INT(10) NOT NULL,  
    PRIMARY KEY(em_username,jobid),  
    CONSTRAINT RQS1  
    FOREIGN KEY (em_username) REFERENCES employe(em_username)  
    ON DELETE CASCADE ON UPDATE CASCADE,  
    CONSTRAINT RQS3  
    FOREIGN KEY (jobid) REFERENCES job_position(jobid)  
    ON DELETE CASCADE ON UPDATE CASCADE  
);
```

DROP TABLE IF EXISTS evaluation;

```
CREATE TABLE evaluation(  
    evalid INT(10) NOT NULL DEFAULT '0',  
    report ENUM ('0','1','2','3','4'),  
    capability ENUM ('0','1','2'),  
    em_username VARCHAR(20) NOT NULL,  
    jobid INT(10) NOT NULL,  
    ev_username VARCHAR(20) NOT NULL DEFAULT 'unknown',  
    ev_comments VARCHAR(100) NOT NULL DEFAULT 'unknown',  
    interview ENUM ('0','1','2','3','4'),  
    PRIMARY KEY(evalid),  
    CONSTRAINT JOBB  
    FOREIGN KEY(jobid) REFERENCES requests(jobid),  
    CONSTRAINT EMPL2
```

```
FOREIGN KEY(em_username) REFERENCES requests(em_username),  
CONSTRAINT SJDJAS  
FOREIGN KEY(ev_username) REFERENCES evaluator(ev_username)  
ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
DROP TABLE IF EXISTS givesreport;
```

```
CREATE TABLE givesreport(  
    evalid INT(10) NOT NULL DEFAULT '0',  
    mReport varchar(20) NOT NULL,  
    m_username varchar(20) NOT NULL,  
    PRIMARY KEY (evalid,m_username),  
    CONSTRAINT GR1  
    FOREIGN KEY (evalid) REFERENCES evaluation(evalid)  
    ON DELETE CASCADE ON UPDATE CASCADE,  
    CONSTRAINT GR2  
    FOREIGN KEY (m_username) REFERENCES manager(m_username)  
    ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
DROP TABLE IF EXISTS evaluationresult;
```

```
CREATE TABLE evaluationresult(  
    reid INT(10) NOT NULL AUTO_INCREMENT,  
    ad_username VARCHAR(20) NOT NULL DEFAULT 'unknown',  
    evalid INT(10) NOT NULL,  
    finalgrade INT(10) NOT NULL DEFAULT '0',  
    PRIMARY KEY(reid),  
    CONSTRAINT USRNAME  
    FOREIGN KEY(ad_username) REFERENCES administrator(ad_username)
```

```
ON DELETE CASCADE ON UPDATE CASCADE,  
CONSTRAINT EVID  
FOREIGN KEY(evalid) REFERENCES evaluation(evalid)  
ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
DROP TABLE IF EXISTS announces;
```

```
CREATE TABLE announces(  
    username VARCHAR(20) NOT NULL DEFAULT 'unknown',  
    jobid INT(10) NOT NULL DEFAULT '0',  
    opendate DATE NOT NULL,  
    closedate DATE NOT NULL,  
    PRIMARY KEY(username,jobid),  
    CONSTRAINT USRNM  
    FOREIGN KEY(username) REFERENCES evaluator(ev_username)  
    ON DELETE CASCADE ON UPDATE CASCADE,  
    CONSTRAINT JOBID  
    FOREIGN KEY(jobid) REFERENCES job_position(jobid)  
    ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
DROP TABLE IF EXISTS field;
```

```
CREATE TABLE field(  
    ftitle VARCHAR(50) NOT NULL DEFAULT 'unknown',  
    perigrafi VARCHAR(20) NOT NULL DEFAULT 'unknown',
```



```
jobid INT(10) NOT NULL DEFAULT '0',
    parent VARCHAR(50) NOT NULL DEFAULT 'unknown',
PRIMARY KEY(ftitle),
CONSTRAINT JBID1
FOREIGN KEY(jobid) REFERENCES job_position(jobid)
ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT PRNT
FOREIGN KEY(parent) REFERENCES field(ftitle)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
DROP TABLE IF EXISTS decides;
```

```
CREATE TABLE decides(
    m_username VARCHAR(50) NOT NULL DEFAULT 'unknown',
    jobid INT(10) NOT NULL DEFAULT '0',
    PRIMARY KEY(m_username,jobid),
    CONSTRAINT DEMA
FOREIGN KEY(m_username) REFERENCES manager(m_username)
ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT DEJOB
FOREIGN KEY(jobid) REFERENCES job_position(jobid)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

INSERT INTO user VALUES

('MScott','Michael','Scott','mscott@gmail.com','2000-02-02','mscott123'),
('DSchrute','Dwight','Schrute','dschrute@gmail.com','2000-02-03','dschrute12'),
('PBeesly','Pam','Beesly','pbeesly@gmail.com','2000-02-04','pbeesly123'),
('JHalpert','Jim','Halpert','jhalpert@gmail.com','2000-02-05','jhalpert12'),
('CBratton','Creed','Bratton','cbratton@gmail.com','2000-02-06','cbratton12'),
('AMartin','Angela','Martin','amartin@gmail.com','2000-02-07','amartin123'),
('TFlenderson','Toby','Flenderson','tflenders@gmail.com','2000-02-08','tflender12'),
('RHoward','Ryan','Howard','rhoward@gmail.com','2000-02-09','rhoward123'),
('JLevinson','Jan','Levinson','jlevinson@gmail.com','2000-02-10','jlevinson1'),
('SHudson','Stanley','Hudson','shudson@gmail.com','2000-02-11','shudson12'),
('ABernard','Andy','Bernard','abernard@gmail.com','2000-02-12','abernard12');

INSERT INTO company VALUES

('Dunder Mifflin',12345678,'DOY1',261012345,'Greece','Skydra','Korinthou',34,'MScott'),
('Corporate',23456789,'DOY2',261023456,'Greece','Edessa','Aiolou',23,'JLevinson'),
('Stamford',34567891,'DOY3',261034567,'Greece','Giannitsa','Maizwnos',4,'SHudson');

INSERT INTO employe VALUES

('DSchrute','Master the Market',2345678,'MyCV1',12345678,20,'Best Employee','Greek,French'),
('PBeesly','The Paper God',3456789,'MyCV2',23456789,20,'Best Receptionist','Greek,Spanish'),
('JHalpert','How to Sell Paper',4567891,'MyCV3',23456789,20,'Best Comedian','Greek,Italian'),
('CBratton','Just Be Idle',5678912,'MyCV4',34567891,20,'Lazy & Thief','Greek,German');

INSERT INTO evaluator VALUES

('AMartin',1,12345678),
('TFlenderson',2,23456789),
('RHoward',3,34567891);

INSERT INTO manager VALUES

('MScott',20,12345678),

```
('JLevinson',20,23456789),  
('SHudson',20,34567891);
```

INSERT INTO administrator VALUES

```
('ABernard');
```

INSERT INTO degree VALUES

```
('Financial Management ','Université du Montreal à Québec',2000,'MASTER','DSchrute'),  
('Supply Chain Management & Logistics ','Amsterdam University of Applied  
Sciences',1999,'MASTER','PBeesly'),  
('Business Analytics','Univeristy of Patras',2000,'UNIV','JHalpert'),  
('Communication Science','University of Thessalonikh',1999,'UNIV','CBratton');
```

INSERT INTO project VALUES

```
(1,'https://github.com/Project1','MyProject1','DSchrute'),  
(2,'https://github.com/Project2','MyProject2','PBeesly'),  
(3,'https://github.com/Project3','MyProject3','JHalpert'),  
(4,'https://github.com/Project4','MyProject4','CBratton');
```

INSERT INTO job_position VALUES

```
(1,'Skydra',10000,'Manager of Marketing'),  
(2,'Edessa',7000,'Web Designer'),  
(3,'Giannitsa',5000,'Logistics'),  
(4,'Giannitsa',6000,'Project Manager'),  
(5,'Skydra',5000,'Social Media Assistant'),  
(6,'Edessa',7000,'Public Relations'),  
(7,'Skydra',6500,'Chief Executive Officer'),  
(8,'Edessa',7500,'Human Resources');
```

INSERT INTO evaluation VALUES

```
(1,'4','2','DSchrute',1,'AMartin','Excellent! The position was made for him!','4'),  
(2,'3','2','PBeesly',3,'TFlenderson','Very Good! We should give him the position!','3'),  
(3,'2','1','JHalpert',4,'RHoward','Average! We should think carefully about giving him the  
position!','2'),  
(4,'1','1','CBratton',8,'AMartin','Horrendous! He must be kidding about his request for  
promotion!','1');
```

INSERT INTO announces VALUES

```
('AMartin',1,'2000-02-13','2000-02-20'),  
(TFlenderson',3,'2000-02-14','2000-02-21'),  
(RHoward',4,'2000-02-15','2000-02-22'),  
(AMartin',8,'2000-02-16','2000-02-23');
```

INSERT INTO field VALUES

```
('Marketing','Promoting Products',1,'Marketing'),  
(Computer Science','Developing Software',2,'Computer Science'),  
(Economics','Budget Management',3,'Economics'),  
(Management','Methods Application',4,'Management'),  
(Journalism','Products Marketing',5,'Journalism'),  
(Business Marketing','Improves Relations',6,'Business Marketing'),  
(Administration','Managing Operations',7,'Administration'),  
(Psychology','Managing People',8,'Psychology');
```

INSERT INTO decides VALUES

```
('MScott',1),  
(JLevinson',2),  
(SHudson',3),  
(SHudson',4),  
(MScott',5),  
(JLevinson',6),  
(MScott',7),
```

```
('JLevinson',8);
```

```
INSERT INTO updates VALUES
```

```
('MScott','DSchrute'),  
('JLevinson','PBeesly'),  
('JLevinson','JHalpert'),  
('SHudson','CBratton');
```

```
INSERT INTO works VALUES
```

```
('MScott',12345678),  
('DSchrute',12345678),  
('PBeesly',23456789),  
('JHalpert',23456789),  
('CBratton',34567891),  
('AMartin',12345678),  
('TFlenderson',23456789),  
('RHoward',34567891),  
('JLevinson',23456789),  
('SHudson',34567891);
```

```
INSERT INTO requests VALUES
```

```
('DSchrute',1),  
('PBeesly',3),  
('JHalpert',4),  
('CBratton',8);
```

```
INSERT INTO givesreport VALUES
```

```
(1,'4','MScott'),  
(2,'3','SHudson'),  
(3,'2','SHudson'),  
(4,'1','JLevinson');
```

DELIMITER \$

CREATE PROCEDURE employee(IN em_name VARCHAR(20), IN em_surname VARCHAR(20))

BEGIN

SELECT requests.em_username AS Username, requests.jobid AS Job_ID FROM requests

INNER JOIN employe ON requests.em_username=employe.em_username

INNER JOIN user ON employe.em_username=user.username

```
WHERE em_name=user.name AND em_surname=user.surname;
```

```
SELECT evaluation.report AS Report, evaluation.capability AS Capability,  
evaluation.interview AS Interview,  
evaluation.ev_comments AS comments FROM evaluation  
INNER JOIN requests ON evaluation.em_username=requests.em_username  
INNER JOIN employe ON requests.em_username=employe.em_username  
INNER JOIN user ON employe.em_username=user.username  
WHERE em_name=user.name AND em_surname=user.surname;
```

```
SELECT user.name, user.surname FROM user  
INNER JOIN evaluator ON user.username=evaluator.ev_username  
INNER JOIN evaluation ON evaluator.ev_username=evaluation.ev_username  
WHERE evaluation.ev_username IN  
(SELECT evaluation.ev_username FROM evaluation  
INNER JOIN requests ON evaluation.em_username=requests.em_username  
INNER JOIN employe ON requests.em_username=employe.em_username  
INNER JOIN user ON employe.em_username=user.username  
WHERE em_name=user.name AND em_surname=user.surname);
```

```
END $
```

```
DELIMITER ;
```

```
call employee('Dwight','Schrute');
```

```
-----
```

```
DROP PROCEDURE IF EXISTS evaluationStatus;
```

```
DELIMITER $
```

```
CREATE PROCEDURE evaluationStatus(IN jjid INT)
```

BEGIN

DECLARE I INT;

DECLARE k INT;

SELECT count(em_username) INTO I from requests WHERE jobid=jid;

SELECT count(finalgrade) INTO k FROM evaluationresult

INNER JOIN evaluation ON evaluation.evalid=evaluationresult.evalid

INNER JOIN evaluator ON evaluation.ev_username=evaluator.ev_username

INNER JOIN announces ON evaluator.ev_username=announces.username

INNER JOIN job_position ON announces.jobid=job_position.jobid

WHERE jid=job_position.jobid;

IF (I=k) THEN

SELECT 'ORISTIKOPOIMENOI PINAKES';

SELECT em_username AS upallilos,finalgrade AS bathmologia FROM evaluationresult

INNER JOIN evaluation ON evaluation.evalid=evaluationresult.evalid

INNER JOIN evaluator ON evaluation.ev_username=evaluator.ev_username

INNER JOIN announces ON evaluator.ev_username=announces.username

INNER JOIN job_position ON announces.jobid=job_position.jobid

WHERE jid=job_position.jobid

ORDER BY finalgrade DESC;

ELSEIF (I>k) THEN


```

SELECT 'EKREMOYN AJIOLOGISEIS';

SELECT em_username AS upallilos,finalgrade AS bathmologia FROM evaluationresult
INNER JOIN evaluation ON evaluation.evalid=evaluationresult.evalid
INNER JOIN evaluator ON evaluation.ev_username=evaluator.ev_username
INNER JOIN announces ON evaluator.ev_username=announces.username
INNER JOIN job_position ON announces.jobid=job_position.jobid
WHERE jid=job_position.jobid

ORDER BY finalgrade DESC;

```

```

ELSEIF (l=0) THEN

```

```

    SELECT 'NO REQUEST';

```

```

END IF ;

```

```

END$

```

```

DELIMITER ;

```

```

-----

```

```

DELIMITER $

```

```

CREATE PROCEDURE sumofevaluation(IN evaluatorID INT,IN jobID INT)

```

```

BEGIN

```

```

DECLARE total INT(10);

```

```

DECLARE id INT(10);

```

```

DECLARE inter ENUM ('0','1','2','3','4');

```

```

DECLARE rep ENUM ('0','1','2','3','4');

```

```

DECLARE cap ENUM ('0','1','2');

```

```

SELECT evaluation.evalid, interview,report,capability, (interview+report+capability - 3) AS
total

```

```
into id, inter, rep , cap, total  
FROM evaluation  
INNER JOIN evaluator ON evaluation.ev_username=evaluator.ev_username  
INNER JOIN announces ON evaluator.ev_username=announces.username  
INNER JOIN job_position ON announces.jobid=job_position.jobid  
WHERE evaluatorID=evaluator.ev_code AND jobID=job_position.jobid  
LIMIT 1;
```

```
IF (inter>0 AND rep>0 AND cap>0) THEN
```

```
    INSERT INTO evaluationresult (ad_username, evalid, finalgrade)  
    VALUES ("ABernard",id, total);
```

```
    SELECT finalgrade  
    FROM evaluationresult  
    INNER JOIN evaluation ON evaluation.evalid=evaluationresult.evalid  
    INNER JOIN evaluator ON evaluation.ev_username=evaluator.ev_username  
    INNER JOIN announces ON evaluator.ev_username=announces.username  
    INNER JOIN job_position ON announces.jobid=job_position.jobid  
    WHERE evaluatorID=evaluator.ev_code AND jobID=job_position.jobid;
```

```
ELSE
```

```
    SELECT 'Not all phases completed';
```

```
END IF;
```

```
END$
```

```
DELIMITER ;
```

```
call sumofevaluation(1,1);
```

```
drop procedure sumofevaluation;
```

```
DROP PROCEDURE IF EXISTS mesos;
```

```
DELIMITER $
```

```
CREATE PROCEDURE mesos(IN evaluator_username VARCHAR(20))
```

```
BEGIN
```

```
SELECT AVG(finalgrade)
```

```
FROM evaluationresult
```

```
WHERE evalid in
```

```
(SELECT evalid FROM evaluation WHERE ev_username=evaluator_username);
```

```
END $
```

```
DELIMITER ;
```

```
DROP TRIGGER IF EXISTS changeprofile;
```

```
DELIMITER $
```

```
CREATE TRIGGER changeprofile BEFORE UPDATE ON user
```

```
FOR EACH ROW
```

```
BEGIN
```

```
IF(OLD.username NOT LIKE 'ABernard') THEN
```

```
        SET NEW.username=OLD.username;

        SET NEW.name=OLD.name;

        SET NEW.surname=OLD.surname;

        SET NEW.email=OLD.email;

        SET NEW.sign_in_date=OLD.sign_in_date;

    END IF;

END$

DELIMITER ;

UPDATE user SET name='papakwstas' WHERE username='ABernard';

SELECT * FROM user WHERE username ='ABernard';
```

```
DROP TRIGGER IF EXISTS companyUpdate;
```

```
DELIMITER $
```

```
CREATE TRIGGER companyUpdate BEFORE UPDATE ON company
FOR EACH ROW
BEGIN
```

```
    SET NEW.afm=OLD.afm;
```

```
SET NEW.companyName=OLD.companyName;
```

```
SET NEW.DOY=OLD.DOY;
```

```
END$
```

```
DELIMITER ;
```

```
ELEGXOS
```

```
UPDATE company SET afm='1111111' WHERE companyName='Dunder Mifflin';
```

```
SELECT afm FROM company WHERE companyName='Dunder Mifflin';
```

```
//Insert
```

```
CREATE TRIGGER job_insert
```

```
AFTER INSERT ON job_position
```

```
FOR EACH ROW
```

```
BEGIN
```

```
DECLARE currDate DATETIME;
```

```
SET currDate=CURRENT_TIMESTAMP();
```

```
INSERT INTO log VALUES(DEFAULT,'ABernard',currDate,'insert','1','job_position');
```

```
END$
```

```
CREATE TRIGGER employe_insert
```

```
AFTER INSERT ON employe
```

```
FOR EACH ROW
```

```
BEGIN
```

```
DECLARE currDate DATETIME;
```

```
SET currDate=CURRENT_TIMESTAMP();
```

```
INSERT INTO log VALUES(DEFAULT,'ABernard',currDate,'insert','1','employe');
```

```
END$
```

```
CREATE TRIGGER requests_insert
BEFORE INSERT ON requests
FOR EACH ROW
BEGIN
DECLARE currDate DATETIME;
SET currDate=CURRENT_TIMESTAMP();
INSERT INTO log VALUES(DEFAULT,'ABernard',currDate,'insert','1','requests');
END$
```

//Update

```
CREATE TRIGGER job_update
AFTER UPDATE ON job_position
FOR EACH ROW
BEGIN
DECLARE currDate DATETIME;
SET currDate=CURRENT_TIMESTAMP();
INSERT INTO log VALUES(DEFAULT,'ABernard',currDate,'update','1','job_position');
END$
```

```
CREATE TRIGGER employe_update
AFTER UPDATE ON employe
FOR EACH ROW
BEGIN
DECLARE currDate DATETIME;
SET currDate=CURRENT_TIMESTAMP();
INSERT INTO log VALUES(DEFAULT,'ABernard',currDate,'update','1','employe');
END$
```

```
CREATE TRIGGER requests_update
AFTER UPDATE ON requests
FOR EACH ROW
BEGIN
DECLARE currDate DATETIME;
SET currDate=CURRENT_TIMESTAMP();
INSERT INTO log VALUES(DEFAULT,'ABernard',currDate,'update','1','requests');
END$
```

//Delete

```
CREATE TRIGGER job_delete
AFTER DELETE ON job_position
FOR EACH ROW
BEGIN
DECLARE currDate DATETIME;
SET currDate=CURRENT_TIMESTAMP();
INSERT INTO log VALUES(DEFAULT,'ABernard',currDate,'delete','1','job_position');
END$
```

```
CREATE TRIGGER employe_delete
AFTER DELETE ON employe
FOR EACH ROW
BEGIN
DECLARE currDate DATETIME;
SET currDate=CURRENT_TIMESTAMP();
INSERT INTO log VALUES(DEFAULT,'ABernard',currDate,'delete','1','employe');
END$
```

```
CREATE TRIGGER requests_delete
AFTER DELETE ON requests
FOR EACH ROW
BEGIN
DECLARE currDate DATETIME;
SET currDate=CURRENT_TIMESTAMP();
INSERT INTO log VALUES(DEFAULT,'ABernard',currDate,'delete','1','requests');
END$
```
