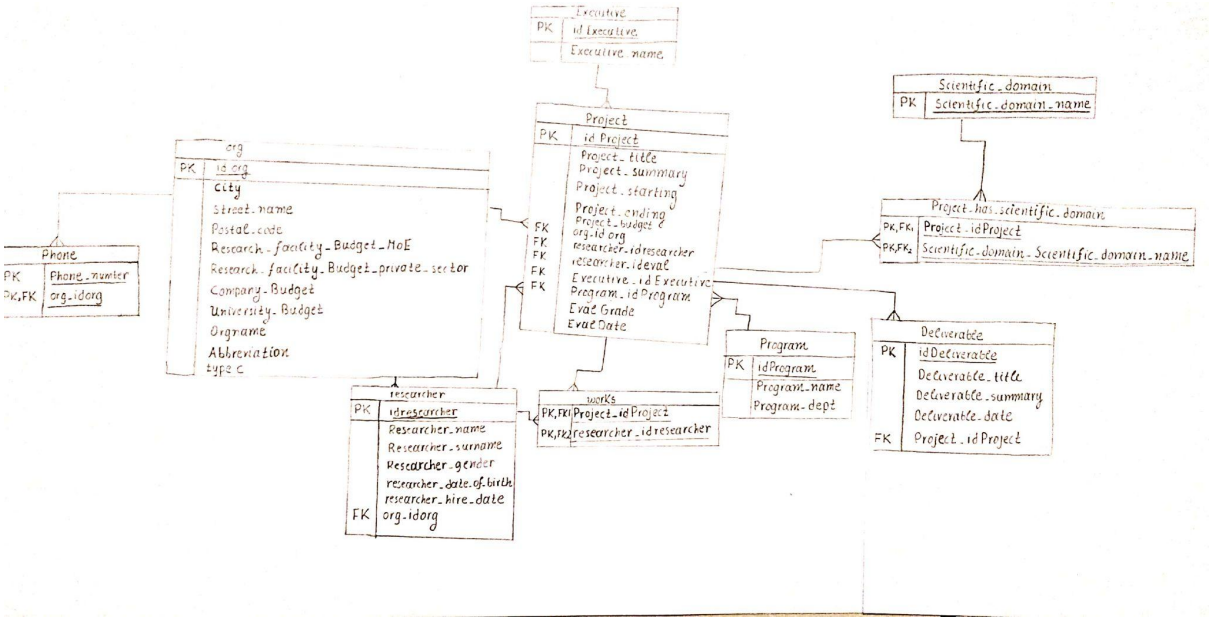
Νάστου Σαββίνα 03119146
Παπαδόπουλος Θεόδωρος 03119017
Πυλιώτης Αθανάσιος 03119050

Βάσεις Δεδομένων
Εξαμηνιαία Εργασία

2.1

Σχεσιακό Διάγραμμα



Οι πίνακες είναι:
Project για έργο/επιχορήγηση
researcher για τους ερευνητές που εργάζονται σε/διαχειρίζονται/αξιολογούν έργο και εργάζονται σε οργανισμό
org για τους οργανισμούς
Executive για τα στελέχη
Program για τα προγράμματα
Deliverable για τα παραδοτέα
Scientific_domain για τα επιστημονικά πεδία
Phone για τα τηλέφωνα του οργανισμού
και τους πίνακες
works και Project_has_scientific_domain για τις σχέσεις “ερευνητής εργάζεται σε έργο” και “έργο αφορά επιστημονικό πεδίο αντίστοιχα”

Για να υλοποιήσουμε το ISA, αντί να έχουμε διαφορετικές οντότητες για το ερευνητικό [κέντρο, το πανεπιστήμιο και την εταιρία, φτιάξαμε έναν πίνακα για τον οργανισμό με attributes τους προϋπολογισμούς και ορίσαμε ένα type ENUM για να διακρίνουμε το είδος του οργανισμού. Ορίσαμε πίνακες για τις σχέσεις works και Project_has_scientific_domain. Οι υπόλοιπες σχέσεις υλοποιούνται μέσω foreign keys (πχ. για τη σχέση “ερευνητής αξιολογεί έργο” έχουμε προσθέσει στον πίνακα του Project ένα foreign key researcher_ideval με το id του researcher που κάνει την αξιολόγηση το οποίο αντιστοιχεί στο primary key

“idresearcher” του πίνακα researcher. Επίσης φτιάξαμε έναν πίνακα για τα τηλέφωνα του οργανισμού με σχέση one-to-many, για να απεικονίσουμε τις αντίστοιχες { } του ER διαγράμματος, ότι δηλαδή ένας οργανισμός μπορεί να έχει πολλά τηλέφωνα.

Ευρετήρια - Indices

Στο περιβάλλον του MySQLWorkbench που δουλεύουμε, δημιουργούνται αυτόματα κάποια indices. Συγκεκριμένα, δημιουργούνται indices για τα primary keys. Αυτό είναι λογικό αν σκεφτούμε ότι τα primary keys χρησιμοποιούνται πολύ συχνά οπότε θέλουμε να έχουμε γρήγορη πρόσβαση σε αυτά. Επίσης δημιουργούνται indices για τα foreign keys επειδή θέλουμε γρήγορη προσπέλασή τους όταν γίνεται κάποια ενημέρωση ή διαγραφή στην οντότητα της οποίας τα κλειδιά είναι foreign keys σε άλλες οντότητες. Τα foreign keys χρησιμοποιούνται επίσης πολύ στα JOIN που εκτελούμε στα queries οπότε η γρήγορη πρόσβαση σε αυτά είναι απαραίτητη και για αυτόν τον λόγο. Προφανώς, αν κάποιο foreign key είναι και primary key, έχει ήδη δημιουργηθεί index για αυτό, επομένως δεν ξαναδημιουργείται. Υλοποιούμε επιπλέον ευρετήρια για την ημερομηνία έναρξης και λήξης, επειδή χρησιμοποιούνται στα queries 3.6

Constraints

Τα παραπάνω constraints τηρούνται ανεξάρτητα με την αλλαγή την οποία θα επιφέρουμε στην βάση(insert,update,delete)

Οργανισμός

- Ανάλογα με το είδος του οργανισμού(enumeraton) να είναι 0 ή διάφορα του 0 τα αντίστοιχα πεδία budget.

Ερευνητής

- Ημερομηνία γέννησης < Ημερομηνία πρόσληψης
- Ημερομηνία γέννησης < Τρέχουσα ημερομηνία
- Οι ερευνητές ενός έργου και ο επιστημονικός υπεύθυνος να μην είναι αξιολογητές
- Η ημερομηνία πρόσληψης ενός εργαζομένου που εργάζεται στο έργο < ημερομηνία λήξης του έργου
- Η ημερομηνία πρόσληψης ενός εργαζομένου που αξιολογεί έργο < ημερομηνία αξιολόγησης του έργου
- Ο ερευνητής δουλεύει μόνο σε έργα που ανήκουν στον οργανισμό που δουλεύει αυτός.
- Ο αξιολογητής δεν δουλεύει στον οργανισμό του οποίου το project αξιολογεί

Έργο

- Ημερομηνία έναρξης < Ημερομηνία λήξης
- Διάρκεια >=1 και <=4
- Ημερομηνία αξιολόγησης<Ημερομηνία έναρξης
- Βαθμός αξιολόγησης >=50 και <=100

Παραδοτέο

- Ημερομηνία παράδοσης < Ημερομηνία λήξης του έργου

Το κάθε τηλέφωνο είναι μοναδικό

Queries

Στα queries θεωρούμε ότι ο manager ανήκει στο δυναμικό που εργάζεται για το έργο. Αυτός είναι ο λόγος που χρησιμοποιούμε union με δύο select στα περισσότερα queries.

- **Query 3.1**

```
form = ProjectFilter()
if(request.method == "POST"):
    new = form.__dict__
    Project_starting = new['Project_starting'].data
    Project_ending =  new['Project_ending'].data
    Executive_idExecutive = new['Executive_idExecutive'].data
    if Executive_idExecutive != None:
        if(Project_ending != None):
            if(Project_starting != None):
                query = "SELECT * FROM project WHERE Project_starting >= '{}' AND Project_ending <= '{}' AND
                Executive_idExecutive = '{}';".format(Project_starting, Project_ending, Executive_idExecutive)
            else:
                query = "SELECT * FROM project WHERE Project_ending <= '{}' AND Executive_idExecutive = '{}';".format(
                Project_ending,Executive_idExecutive)
        else:
            if(Project_starting != None):
                query = "SELECT * FROM project WHERE Project_starting >= '{}' AND Executive_idExecutive = '{}';".format(
                Project_starting, Executive_idExecutive)
            else:
                query = "SELECT * FROM project WHERE Executive_idExecutive = '{}';".format(Executive_idExecutive)
    else:
        if(Project_ending != None):
            if(Project_starting != None):
                query = "SELECT * FROM project WHERE Project_starting >= '{}' AND Project_ending <= '{}';".format(
                Project_starting, Project_ending)
            else:
                query = "SELECT * FROM project WHERE Project_ending <= '{}';".format(Project_ending)
        else:
            if(Project_starting != 0):
                query = "SELECT * FROM project WHERE Project_starting >= '{}';".format(Project_starting)
            else:
                query = "SELECT * FROM project;"
```

- **Query 3.3**

```sql
drop view if exists proj_res;
CREATE VIEW proj_res AS
(SELECT
r.idresearcher,

CONCAT(r.researcher_name," ", r.researcher_surname) as Full_name,
p.idProject,
p.Project_title
FROM researcher r
INNER JOIN works w ON r.idresearcher = w.researcher_idresearcher
INNER JOIN Project p ON p.idProject = w.Project_idProject )
union
(SELECT DISTINCT
r.idresearcher,

CONCAT(r.researcher_name," ", r.researcher_surname) as Full_name,
p.idProject,
p.Project_title
FROM Project p
inner join researcher r on p.researcher_idresearcher=r.idresearcher
) order by idresearcher
```

```sql
CREATE VIEW res_org AS
SELECT
r.idresearcher,
CONCAT(r.researcher_name," ", r.researcher_surname) as Full_name,
o.idorg,
o.Orgname,
o.Abbreviation
FROM researcher r
INNER JOIN org o ON r.org_idorg = o.idorg
```

- **Query 3.3**

```sql
select  s.Scientific_domain_Scientific_domain_name,p.idProject,p.Project_starting,
p.researcher_idresearcher,p.Project_ending,r.idresearcher,r.Researcher_name,r.Researcher_surname from
 Project_has_Scientific_domain s
 inner join Project p on s.Project_idProject=p.idProject
 inner join works w on w.Project_idProject=p.idProject
 inner join researcher r on r.idresearcher=w.researcher_idresearcher
 where s.Scientific_domain_Scientific_domain_name='Law' and p.Project_starting<curdate() and p.Project_ending>curdate()
 union
 select s.Scientific_domain_Scientific_domain_name,p.idProject,p.Project_starting,p.researcher_idresearcher,
 p.Project_ending,r.idresearcher,r.Researcher_name,r.Researcher_surname from
 Project_has_Scientific_domain s inner join Project p on s.Project_idProject=p.idProject
 inner join researcher r on r.idresearcher=p.researcher_idresearcher
 where s.Scientific_domain_Scientific_domain_name='Law' and p.Project_starting<curdate() and p.Project_ending>curdate() order by idProject;
```

- **Query 3.4**

```
select distinct po.idorg,po.cnt
from ((select   o.idorg, EXTRACT(YEAR FROM p.Project_starting) as pyear,count(distinct idProject) as cnt from org o inner join Project p on o.idorg=p.org_idorg
group by idorg,pyear )  po
inner join
(select   o.idorg, EXTRACT(YEAR FROM p.Project_starting) as pyear,count(distinct idProject) as cnt
from org o inner join Project p on o.idorg=p.org_idorg group by idorg,pyear ) l on l.idorg=po.idorg)
where ((po.pyear=l.pyear+1 or po.pyear=l.pyear-1)and po.cnt=l.cnt and po.cnt>=10)  ;
```

- **Query 3.5**

```
select   t.Scientific_domain_Scientific_domain_name ,s.Scientific_domain_Scientific_domain_name,count(*)
from Project_has_Scientific_domain s inner join Project_has_Scientific_domain t on t.Project_idProject=s.Project_idProject
where t.Scientific_domain_Scientific_domain_name<s.Scientific_domain_Scientific_domain_name
group by t.Scientific_domain_Scientific_domain_name ,s.Scientific_domain_Scientific_domain_name order by count(*) DESC  LIMIT 3;
```

- **Query 3.6**

```
select  st.researcher_idresearcher,st.researcher_date_of_birth,count(st.researcher_idresearcher)
from( select w.researcher_idresearcher,r.researcher_date_of_birth,w.Project_idProject
from researcher r inner join works w on w.researcher_idresearcher=r.idresearcher
inner join Project p on p.idProject=w.Project_idProject
where p.Project_starting<curdate() and p.Project_ending>curdate() and DATEDIFF(curdate(),r.researcher_date_of_birth)<40*365
union
select p.researcher_idresearcher,r.researcher_date_of_birth,p.idProject
from Project p inner join researcher r on r.idresearcher=p.researcher_idresearcher
where p.Project_starting<curdate() and p.Project_ending>curdate() and  DATEDIFF(curdate(),r.researcher_date_of_birth)<40*365) st
group by st.researcher_idresearcher,st.researcher_date_of_birth order by count(st.researcher_idresearcher) DESC LIMIT 5;
```

- **Query 3.7**

```
 SELECT
sl.Executive_name,
sl.Orgname,
sum(sl.Project_budget) as sumb
FROM
(SELECT DISTINCT
e.Executive_name,
o.Orgname,
o.typec,
p.Project_budget,
e.idExecutive,
o.idorg
FROM Executive e
INNER JOIN Project p ON e.idExecutive = p.Executive_idExecutive
INNER JOIN org o ON o.idorg = p.org_idorg)sl
WHERE(sl.typec = 'Comp')
GROUP BY sl.idExecutive,sl.Orgname
ORDER BY sumb DESC
LIMIT 5;
```

- **Query 3.8**

```
select * from
 (SELECT
CONCAT(l.researcher_name," ", l.researcher_surname) as Full_name,
l.idresearcher,count(distinct l.idProject) as cnt
FROM
(
(SELECT r.idresearcher,r.researcher_name,r.researcher_surname,p.idProject from
works w
INNER JOIN researcher r ON r.idresearcher = w.researcher_idresearcher
INNER JOIN Project p ON p.idProject = w.Project_idProject
WHERE p.idProject not in (select d.Project_idProject from Deliverable d))
union (select r.idresearcher,r.researcher_name,r.researcher_surname,p.idProject from
 Project p
 inner join researcher r on r.idresearcher=p.researcher_idresearcher
 WHERE p.idProject not in (select d.Project_idProject from Deliverable d))) l

GROUP BY l.idresearcher,l.researcher_name,l.researcher_surname) m
where m.cnt>=5;
```

2.2

Τα ddl ,dml scripts παρατίθενται στο github. Οι triggers βρίσκονται στο dml αρχείο insert.sql καθώς αφορούν dml λειτουργίες.

2.3

Τεχνολογία ανάπτυξης εφαρμογής

Dependencies

MySQL για Windows

XAMPP για Windows

Είναι πιθανό το XAMPP να μην μπορεί να ρυθμίσει την python. Θα πρέπει να ακολουθήσετε αυτά τα βήματα:

Ανοίξτε τον πίνακα ελέγχου του XAMPP

Εκκινήστε τον Apache και τη MySQL (θα πρέπει να το κάνετε αυτό ούτως ή άλλως)

Από το Apache ανοίξτε το Config και πατήστε Apache (httpd.conf)

Πατήστε ctrl+f, αναζητήστε το AddHandler και προσθέστε .py στο τέλος

Η Python, με τις πρόσθετες βιβλιοθήκες:

Flask

Flask-MySQLdb

Flask-WTForms

Οδηγίες Εγκατάστασης

Αρχικά στον υπολογιστή μας έχουμε εγκατεστημένο DBMS (στην περίπτωσή μας είναι το MysqlWorkbench), έχοντας δημιουργήσει MySQL user με privileges. Επίσης χρειαζόμαστε έναν web server (στην περίπτωσή μας τον XAMP). Εκεί τρέχουμε τα scripts Schema.sql και Insert.sql

Ένα package με το όνομα "ELIDEK", περιέχει τον κώδικα και τα αρχεία της εφαρμογής, χωρισμένα σε φακέλους για κάθε κατηγορία (models, controllers, HTML templates - views, static files όπως css ή images).

Η __init__.py ρυθμίζει την εφαρμογή, συμπεριλαμβανομένων των απαραίτητων πληροφοριών και διαπιστευτηρίων για τη βάση δεδομένων.
Η forms.py περιέχει τις απαραίτητες φόρμες/κλάσεις που χρησιμοποιήθηκαν για την εφαρμογή αυτή
Η routes.py περιέχει επί του παρόντος όλα τα τελικά σημεία και τους αντίστοιχους controllers
Η run.py εκκινεί τον απλό, ενσωματωμένο διακομιστή και εκτελεί την εφαρμογή σε αυτόν
Όταν βρίσκεστε μέσα στη βάση δεδομένων, εκτελέστε μέσω της εντολής setx FLASK_APP "app.py", για να ορίσετε τη μεταβλητή περιβάλλοντος FLASK_APP στην app.py, και στη συνέχεια χρησιμοποιήστε το flask run.

Μέσα στο Windows PowerShell: Χρησιμοποιήστε pip3 install <package_name> για να εγκαταστήσετε κάθε μεμονωμένο πακέτο Python (βιβλιοθήκη) απευθείας για ολόκληρο το σύστημα. Συνιστάται να δημιουργήσετε ένα εικονικό περιβάλλον με τη μονάδα venv. Θα πρέπει να εκτελέσετε την εντολή py -3 -m venv <VE_name> για να δημιουργήσετε το εικονικό σας περιβάλλον και μέσα σε αυτό να κατεβάσετε τα απαραίτητα πακέτα. Χρησιμοποιήστε το <VE_name>\Scripts\activate για να το ενεργοποιήσετε και στη συνέχεια cd <VE_name> για να μπείτε στο εικονικό περιβάλλον. Τα απαραίτητα πακέτα για αυτή την εφαρμογή αναφέρονται στο requirements.txt και μπορούν να εγκατασταθούν όλα μαζί μέσω pip install -r requirements.txt. Το Virtual Environment θα πρέπει να προτιμάται στο αρχείο htdocs του xampp για Windows. Τα αρχεία της βάσης δεδομένων σας θα πρέπει να βρίσκονται όλα μέσα στο Virtual Environment, αφού το κατεβάσετε και το αποσυμπιέσετε από το GitHub.

2.4
Ο σύνδεσμος για το git repo της εφαρμογής μας είναι: https://github.com/Athan-P/Databases-Project

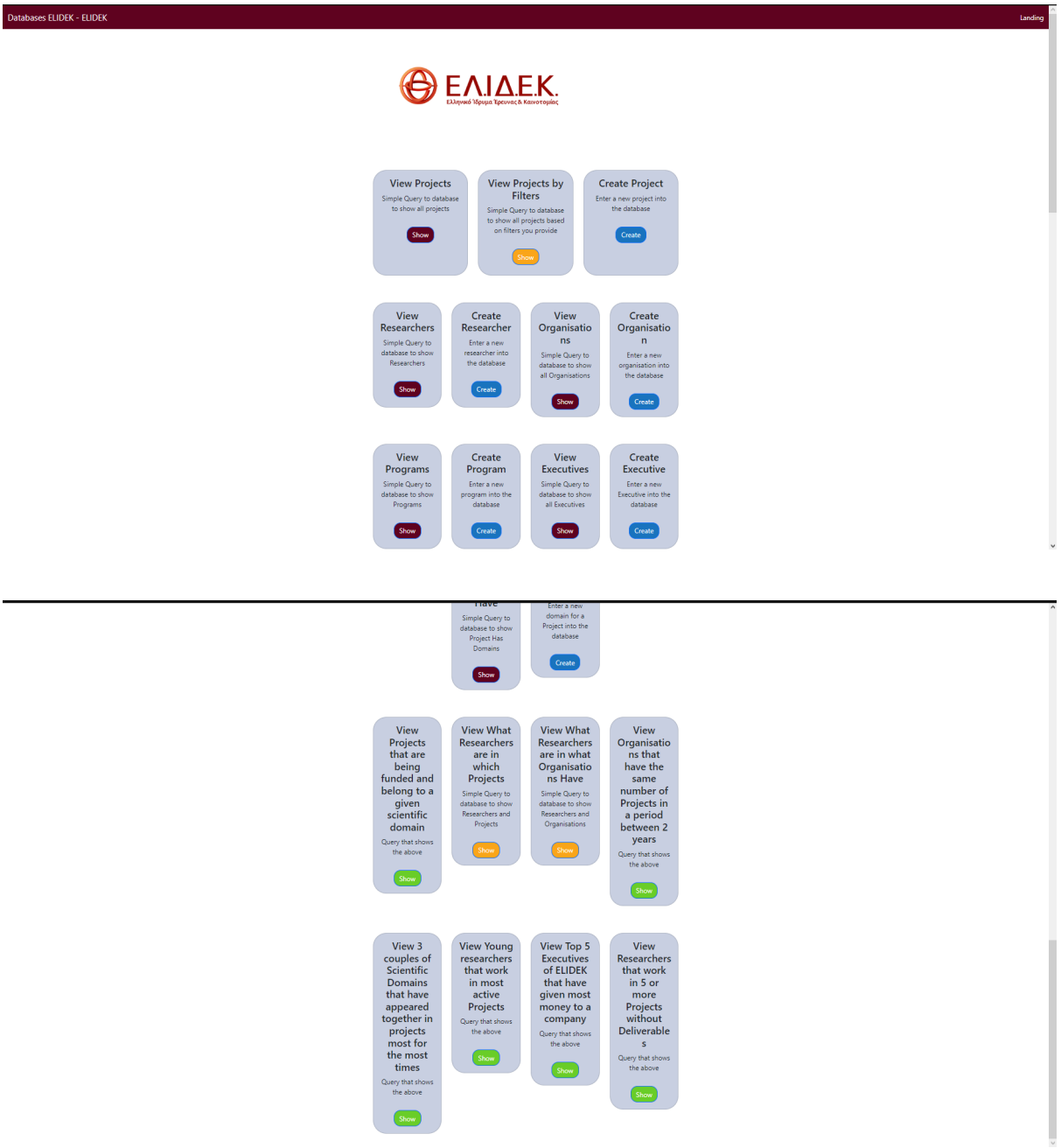
Στοιχεία χρήστη είναι by default:

```
app.config["MYSQL_USER"] = 'root'
app.config["MYSQL_PASSWORD"] = ''
app.config["MYSQL_DB"] = 'myDB'
app.config["MYSQL_HOST"] = 'localhost'
app.config["SECRET_KEY"] = 'key'
```

app.config["WTF_CSRF_SECRET_KEY"] = 'key'

Οδηγίες Πλοήγησης στην εφαρμογή

Στην Εφαρμογή μας το User Interface είναι όπως φαίνεται παρακάτω:



Όπως μπορείτε να δείτε, η λειτουργία λήψης δεδομένων όπως και η δημιουργία καινούργιων είναι στην αρχική σελίδα ELIDEK. Μέσα στη Λήψη γίνεται και το Update και το Delete. Στο τέλος της σελίδας, μετά τις λήψεις και τις δημιουργίες για όλους τους πίνακες, μπορούμε να δούμε όλα τα queries.