ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΠΑΡΑΔΟΤΕΟ 2

ΟΜΑΔΑ:112

Git repo URL : https://github.com/SpyrosGiannopoulos/DB_PROJECT-PUBLIC.git

ΜΕΛΗ: ΣΠΥΡΙΔΩΝ ΓΙΑΝΝΟΠΟΥΛΟΣ 03119158

ΔΗΜΟΣΘΕΝΗΣ ΘΑΝΟΠΟΥΛΟΣ 03119072

Το σχεσιακό διάγραμμα προκύπτει με βάση τους εξής κανόνες (από το προτεινόμενο ER model έχοντας προσθέσει ως PKs σε κάθε οντότητα τα ids):

- Σε μια one to many σχέση με total participation από τη μεριά του many δε δημιουργούμε πίνακα για την σχέση που τα ενώνει και προσθέτουμε στην οντότητα της many πλευράς ως attribute το primary key (PK) της οντότητας της one side ως foreign key(FK). Αυτό γίνεται στις περιπτώσεις executives-projects, programs-projects, researchers(scientific advisor)-projects, organizations-projects, researchers-organizations.

- Σε μια weak entity σχέση με total participation από τη μεριά του many δε δημιουργούμε πίνακα για την σχέση που τα ενώνει και προσθέτουμε στην ασθενή οντότητα το PK της ισχυρής ως PK,FK (deliverables-projects).

- Σε μια many to many σχέση με total participation από τη μια μεριά δημιουργούμε ξεχωριστό πίνακα για την σχέση που τα ενώνει με στοιχεία το PK_1 του πίνακα αυτού, καθώς και PKs, FKs, τα primary keys των οντοτήτων που ενώνει. Επίσης στη μεριά που έχει το total participation τοποθετούμε ως FK το PK_1 εξασφαλίζοντας έτσι την πλήρη συμμετοχή (describes-projects, scientific fields-projects).

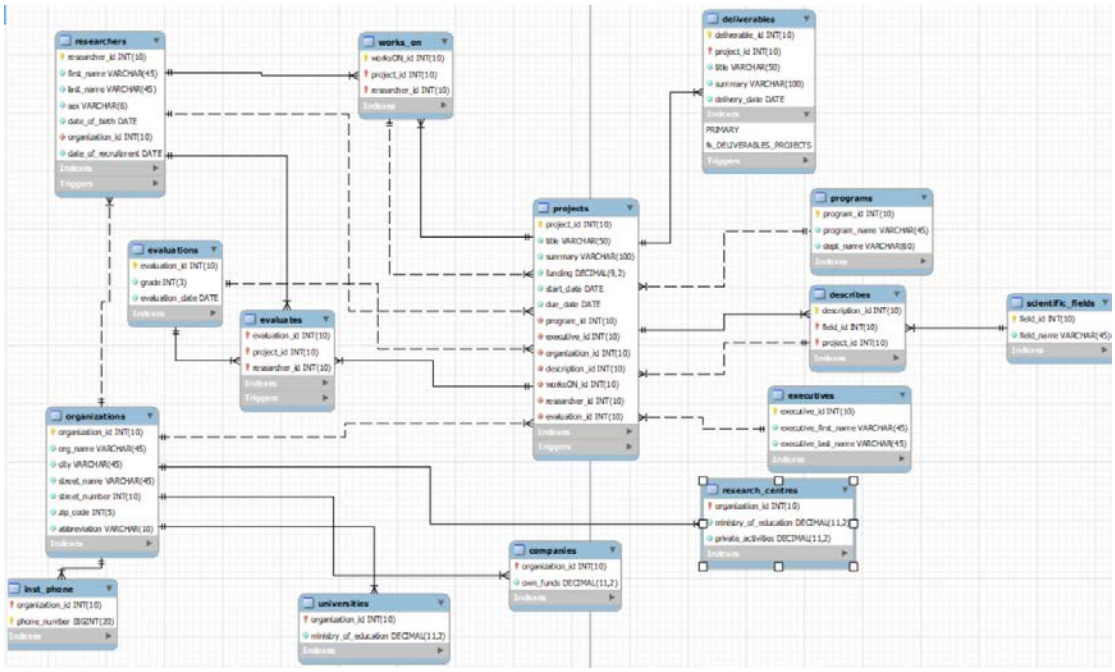
- Σε μια many to one σχέση με total participation και από τις δύο μεριές δημιουργούμε ξεχωριστό πίνακα για την σχέση που τα ενώνει με στοιχεία PKs, FKs τα primary keys των οντοτήτων που ενώνει. Επίσης στη many πλευρά προσθέτουμε ως FK το PK της one πλευράς εξασφαλίζοντας έτσι την πλήρη συμμετοχή (evaluations-projects).

Σημείωση: Στην περίπτωση του evaluations-projects που δημιουργείται ο πίνακας evaluates, επειδή υπάρχει και μια one to many σχέση με total participation από τη μεριά του many, αποφασίζεται στη μεριά του evaluates να προστεθεί ως PK,FK το researcher_id.

- Η ISA σχέση, που ενώνει τον πίνακα organizations με τις κατηγορίες του μπορεί να ανήκει, υλοποιείται με τη δημιουργία οντότητας για την κάθε επιμέρους κατηγορίας με PK,FK το organization_id και τα επιμέρους attributes της καθεμιάς.

- Το multivalued attribute του organization, το phone number, χρειάζεται δημιουργία νέας οντότητας inst_phone με PK,FK το organization_id και PK το phone number, ώστε έτσι ο κάθε οργανισμός να μπορεί να έχεις πολλά τηλέφωνα.

Το relational diagram που προκύπτει:



Δημιουργία πίνακα projects με PK το μοναδικό χαρακτηριστικό του project_id (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή και auto increment γιατί το id είναι μοναδικό κάθε φορά και επιλέγουμε να αυξάνεται αυτόματα σε κάθε insert), attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν και FKs που προκύπτουν από τους κανόνες που αναφέρθηκαν παραπάνω.

```
#Δημιουργία πίνακα projects (χωρίς τα foreign keys)

CREATE TABLE PROJECTS
(
    project_id int UNSIGNED NOT NULL AUTO_INCREMENT,
    title varchar(50)  NOT NULL,
    summary varchar(100) NOT NULL,
    funding DECIMAL(9,2) NOT NULL,
    start_date DATE NOT NULL,
    due_date DATE NOT NULL,
    PRIMARY KEY  (project_id)
);
```

Προσθήκη των foreign keys στο projects

```
#Προσθήκη program_id ως foreign key στο projects

ALTER TABLE PROJECTS
ADD program_id int UNSIGNED NOT NULL;

ALTER TABLE PROJECTS
ADD CONSTRAINT fk_PROJECTS_PROGRAMS FOREIGN KEY (program_id)
REFERENCES PROGRAMS (program_id) ON DELETE RESTRICT ON UPDATE CASCADE;

#Προσθήκη executive_id ως foreign key στο projects

ALTER TABLE PROJECTS
ADD executive_id int unsigned not null,
ADD CONSTRAINT fk_PROJECTS_EXECUTIVES FOREIGN KEY (executive_id)
REFERENCES EXECUTIVES(executive_id) ON DELETE RESTRICT ON UPDATE CASCADE;

#Προσθήκη organization_id ως foreign key στο projects

ALTER TABLE PROJECTS
ADD organization_id int unsigned not null,
ADD CONSTRAINT fk_PROJECTS_ORGANIZATIONS FOREIGN KEY (organization_id)
REFERENCES ORGANIZATIONS(organization_id) ON DELETE RESTRICT ON UPDATE CASCADE;


#Προσθήκη description_id ως foreign key στο projects

ALTER TABLE PROJECTS
ADD description_id int unsigned not null,
ADD CONSTRAINT fk_PROJECTS_DESCRIBES FOREIGN KEY (description_id)
REFERENCES DESCRIBES(description_id) ON DELETE RESTRICT ON UPDATE CASCADE;


#Προσθήκη worksON_id ως foreign key στο projects

ALTER TABLE PROJECTS
ADD worksON_id int unsigned not null,
ADD CONSTRAINT fk_PROJECTS_WORKS_ON FOREIGN KEY (worksON_id)
REFERENCES WORKS_ON(worksON_id) ON DELETE RESTRICT ON UPDATE CASCADE;
```

```
#Προσθήκη researcher_id ως foreign key στο projects
#Προσθήκη evaluation_id ως foreign key στο projects

ALTER TABLE PROJECTS
ADD researcher_id int unsigned not null,
ADD CONSTRAINT fk_PROJECTS_RESEARCHERS FOREIGN KEY (researcher_id)
REFERENCES RESEARCHERS(researcher_id) ON DELETE RESTRICT ON UPDATE CASCADE,
ADD evaluation_id int unsigned not null,
ADD CONSTRAINT fk_PROJECTS_EVALUATIONS FOREIGN KEY (evaluation_id)
REFERENCES EVALUATIONS(evaluation_id) ON DELETE RESTRICT ON UPDATE CASCADE;
```

Δημιουργία πίνακα researchers με PK το μοναδικό χαρακτηριστικό του researcher_id (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή και auto increment γιατί το id είναι μοναδικό κάθε φορά και επιλέγουμε να αυξάνεται αυτόματα σε κάθε insert), attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν και FKs που προκύπτουν από τους κανόνες αναφέρθηκαν παραπάνω. Επίσης προστίθεται και ως attribute το date_of_recruitment που βρίσκεται πάνω στη σχέση “υπαλληλική σχέση” του ER, η οποία δεν αναπαρίσταται στο relational diagram.

```
#Δημιουργία πίνακα researchers (χωρίς τα foreign keys και 1 attribute)

CREATE TABLE RESEARCHERS
(
    researcher_id int UNSIGNED NOT NULL AUTO_INCREMENT,
    first_name varchar(45) NOT NULL,
    last_name varchar(45) NOT NULL,
    sex varchar(6) NOT NULL,
    date_of_birth DATE NOT NULL,
    PRIMARY KEY (researcher_id)
);

#Προθήκη 2 attributes στο researchers, όπου το organization_id είναι foreign key

ALTER TABLE RESEARCHERS
ADD organization_id int unsigned not null,
ADD date_of_recruitment date not null,
ADD CONSTRAINT fk_RESEARCHERS_ORGANIZATIONS FOREIGN KEY (organization_id)
REFERENCES ORGANIZATIONS(organization_id) ON DELETE RESTRICT ON UPDATE CASCADE;
```

Δημιουργία πίνακα organizations με PK το μοναδικό χαρακτηριστικό του organization_id (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή και auto increment γιατί το id είναι μοναδικό κάθε φορά και επιλέγουμε να αυξάνεται αυτόματα σε κάθε insert), attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν.

```
#Δημιουργία πίνακα organizations

CREATE TABLE ORGANIZATIONS
(
    organization_id int UNSIGNED NOT NULL AUTO_INCREMENT,
    org_name varchar(45) NOT NULL,
    city varchar(45) NOT NULL,
    street_name varchar(45) NOT NULL,
    street_number int unsigned NOT NULL,
    zip_code int(5) NOT NULL,
    abbreviation varchar(10) NOT NULL,
    PRIMARY KEY (organization_id)
);
```

Δημιουργία πίνακα inst_phone με PK το μοναδικό συνδυασμό χαρακτηριστικών του organization_id (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή) και phone_number (not null), attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν.

```
#Δημιουργία πίνακα inst_phone

CREATE TABLE INST_PHONE
(
    organization_id int unsigned not null,
    phone_number BIGINT unsigned not null,
    PRIMARY KEY(organization_id,phone_number)
);

#Προσθήκη organization_id ως foreign key στο inst_phone

ALTER TABLE INST_PHONE
ADD CONSTRAINT fk_INST_PHONE_ORGANIZATIONS FOREIGN KEY (organization_id)
REFERENCES ORGANIZATIONS(organization_id) ON DELETE RESTRICT ON UPDATE CASCADE;
```

Δημιουργία πίνακα universities με PK το μοναδικό χαρακτηριστικό του organization_id (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή), attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν, καθώς και FK το organization_id που είναι μέλος της ISA σχέσης.

```
#Δημιουργία πίνακα universities

CREATE TABLE UNIVERSITIES
(
    organization_id int unsigned not null,
    ministry_of_education DECIMAL(11,2) NOT NULL,
    PRIMARY KEY(organization_id)
);

#Προσθήκη organization_id ως foreign key στο universities

ALTER TABLE UNIVERSITIES
ADD CONSTRAINT fk_UNIVERSITIES_ORGANIZATIONS FOREIGN KEY (organization_id)
REFERENCES ORGANIZATIONS(organization_id) ON DELETE RESTRICT ON UPDATE CASCADE;
```

Δημιουργία πίνακα companies με PK το μοναδικό χαρακτηριστικό του organization_id (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή), attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν, καθώς και FK το organization_id που είναι μέλος της ISA σχέσης.

```
#Δημιουργία πίνακα companies

CREATE TABLE COMPANIES
(
    organization_id int unsigned not null,
    own_funds DECIMAL(11,2) NOT NULL,
    PRIMARY KEY(organization_id)
);

#Προσθήκη organization_id ως foreign key στο companies

ALTER TABLE COMPANIES
ADD CONSTRAINT fk_COMPANIES_ORGANIZATIONS FOREIGN KEY (organization_id)
REFERENCES ORGANIZATIONS(organization_id) ON DELETE RESTRICT ON UPDATE CASCADE;
```

Δημιουργία πίνακα research centres με PK το μοναδικό χαρακτηριστικό του organization_id (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή), attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν, καθώς και FK το organization_id που είναι μέλος της ISA σχέσης.

```
#Δημιουργία πίνακα research_centres

CREATE TABLE RESEARCH_CENTRES
(
    organization_id int unsigned not null,
    ministry_of_education DECIMAL(11,2) NOT NULL,
    private_activities DECIMAL(11,2) NOT NULL,
    PRIMARY KEY(organization_id)
);

#Προσθήκη organization_id ως foreign key στο research_centres

ALTER TABLE RESEARCH_CENTRES
ADD CONSTRAINT fk_RESEARCH_CENTRES_ORGANIZATIONS FOREIGN KEY (organization_id)
REFERENCES ORGANIZATIONS(organization_id) ON DELETE RESTRICT ON UPDATE CASCADE;
```

Δημιουργία πίνακα programs με PK το μοναδικό χαρακτηριστικό του program_id (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή και auto increment γιατί το id είναι μοναδικό κάθε φορά και επιλέγουμε να αυξάνεται αυτόματα σε κάθε insert), attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν.

```
#Δημιουργία πίνακα programs

CREATE TABLE PROGRAMS
(
    program_id int UNSIGNED NOT NULL AUTO_INCREMENT,
    program_name varchar(45) NOT NULL,
    dept_name varchar(60) NOT NULL,
    PRIMARY KEY (program_id)
);
```

Δημιουργία πίνακα organizations με PK το μοναδικό χαρακτηριστικό του organization_id (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή και auto increment γιατί το id είναι μοναδικό κάθε φορά και επιλέγουμε να αυξάνεται αυτόματα σε κάθε insert), attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν.

```
#Δημιουργία πίνακα scientific_fields

CREATE TABLE SCIENTIFIC_FIELDS
(
    field_id int unsigned not null auto_increment,
    field_name varchar(45) NOT NULL,
    PRIMARY KEY(field_id)
);
```

Δημιουργία πίνακα executives με PK το μοναδικό χαρακτηριστικό του executive_id (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή και auto increment γιατί το id είναι μοναδικό κάθε φορά και επιλέγουμε να αυξάνεται αυτόματα σε κάθε insert), attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν.

```
#Δημιουργία πίνακα executives

CREATE TABLE EXECUTIVES
(
    executive_id int UNSIGNED NOT NULL AUTO_INCREMENT,
    executive_first_name varchar(45) NOT NULL,
    executive_last_name varchar(45) NOT NULL,
    PRIMARY KEY (executive_id)
);
```

Δημιουργία πίνακα deliverables με PK το μοναδικό συνδυασμό του project_id και deliverable (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή) και attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν. Το project_id είναι και foreign key.

```
#Δημιουργία πίνακα deliverables (χωρίς τα foreign keys)

CREATE TABLE DELIVERABLES
(
    deliverable_id int unsigned not null,
    project_id int unsigned not null,
    title varchar(50) not null,
    summary varchar(100) not null,
    delivery_date date not null,
    PRIMARY KEY(deliverable_id,project_id)
);

#Προσθήκη project_id ως foreign key στο deliverables

ALTER TABLE DELIVERABLES
ADD CONSTRAINT fk_DELIVERABLES_PROJECTS FOREIGN KEY (project_id)
REFERENCES PROJECTS(project_id) ON DELETE RESTRICT ON UPDATE CASCADE;
```

Δημιουργία πίνακα evaluations με PK το μοναδικό χαρακτηριστικό του evaluation_id (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή και auto-increment) και attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν.

```
#Δημιουργία πίνακα evaluations

CREATE TABLE EVALUATIONS
(
    evaluation_id int UNSIGNED NOT NULL AUTO_INCREMENT,
    grade int(3) unsigned NOT NULL,
    evaluation_date DATE NOT NULL,
    PRIMARY KEY (evaluation_id)
);
```

Δημιουργία πίνακα evaluates με PK το μοναδικό συνδυασμό του evaluation_id,project_id,researcher_id, τα οποία είναι και FKs (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή) και attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν.

```
#Δημιουργία πίνακα evaluates

CREATE TABLE EVALUATES
(
    evaluation_id int unsigned not null auto_increment,
    project_id int unsigned not null,
    researcher_id int unsigned not null,
    PRIMARY KEY(evaluation_id,project_id,researcher_id)
);

#Προσθήκη evaluation_id ως foreign key στο evaluates
#Προσθήκη project_id ως foreign key στο evaluates
#Προσθήκη researcher_id ως foreign key στο evaluates

ALTER TABLE EVALUATES
ADD CONSTRAINT fk_EVALUATES_EVALUATIONS FOREIGN KEY(evaluation_id)
REFERENCES EVALUATIONS(evaluation_id) ON DELETE RESTRICT ON UPDATE CASCADE,
ADD CONSTRAINT fk_EVALUATES_PROJECTS FOREIGN KEY(project_id)
REFERENCES PROJECTS(project_id) ON DELETE RESTRICT ON UPDATE CASCADE,
ADD CONSTRAINT fk_EVALUATES_RESEARCHERS FOREIGN KEY(researcher_id)
REFERENCES RESEARCHERS(researcher_id) ON DELETE RESTRICT ON UPDATE CASCADE
```

Δημιουργία πίνακα describes με PK το μοναδικό συνδυασμό των description_id, field_id, project_id (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή και το description_id auto-increment γιατί θέλουμε αυτό να υπάρχει μια φορά το description_id) και attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν. Επίσης τα field_id και project_id είναι FKs.

```
#Δημιουργία πίνακα describes (χωρίς foreign keys)

CREATE TABLE DESCRIBES
(
    description_id int unsigned not null auto_increment,
    field_id int unsigned not null,
    project_id int unsigned not null,
    PRIMARY KEY(description_id,field_id,project_id)
);


#Προσθήκη project_id ως foreign key στο describes

ALTER TABLE DESCRIBES
ADD CONSTRAINT fk_DESCRIBES_PROJECTS FOREIGN KEY (project_id)
REFERENCES PROJECTS(project_id) ON DELETE RESTRICT ON UPDATE CASCADE;

#Προσθήκη field_id ως foreign key στο describes

ALTER TABLE DESCRIBES
ADD CONSTRAINT fk_DESCRIBES_SCIENTIFIC_FIELDS FOREIGN KEY (field_id)
REFERENCES scientific_fields(field_id) ON DELETE RESTRICT ON UPDATE CASCADE;
```

Δημιουργία πίνακα works_on με PK το μοναδικό συνδυασμό των worksON_id, project_id, researcher_id (not null διότι θέλουμε να υπάρχει οπωσδήποτε τιμή και το worksON_id auto-increment γιατί θέλουμε αυτό να υπάρχει μια φορά το worksON_id) και attributes τα επιμέρους στοιχεία που το χαρακτηρίζουν. Επίσης τα project_id και researcher_id.

Δημιουργία constraints και triggers

Δημιουργία constraints τα οποία ελέγχουν αν το funding είναι στα επιθυμητά όρια αν η ημερομηνία έναρξης ενός έργου είναι μικρότερη από της λήξης κι αν η διάρκεια του έργου είναι μεταξύ 1 και 4 έτη, (όπως ζητείται στην εκφώνηση). Επίσης ελέγχουμε αν το έργο έχει πάρει έγκριση (επιλογή προσθήκης στη βάση μόνο έργων που έχουν εγκριθεί), και ο ερευνητής (με sex M ή F) να είναι ενήλικος κατά την ημέρα πρόσληψης.

```
#Έλεγχος στο projects

ALTER TABLE PROJECTS
ADD CONSTRAINT check_funding_min CHECK (100000.00<=funding),
ADD CONSTRAINT check_funding_max CHECK(funding<=1000000.00),
ADD CONSTRAINT valid_date CHECK (start_date<due_date),
ADD CONSTRAINT valid_duration CHECK (due_date>DATE_SUB(start_date,INTERVAL -1 YEAR)),
ADD CONSTRAINT valid_runtime CHECK (due_date<DATE_SUB(start_date,INTERVAL -4 YEAR));

#Δεν επιτρέπεται η εισαγωγή έργων που δεν έχουν πάρει έγκριση δηλαδή με βαθμό<50

ALTER TABLE EVALUATIONS
ADD CONSTRAINT check_grade CHECK (grade>=50);

#Έλεγχος η τιμή για το φύλο του ερευνητή να εισάγεται ως (M ή F)
#Ο ερευνητής να είναι 18 ετών την ημέρα πρόσληψής του στον οργανισμό

ALTER TABLE RESEARCHERS
ADD CONSTRAINT check_sex CHECK ((sex='M') XOR (sex='F')),
ADD CONSTRAINT check_date_of_birth CHECK (DATE_SUB(date_of_birth,INTERVAL -18 YEAR)<date_of_recruitment);
```

Δημιουργία triggers που ελέγχουν η ημέρα πρόσληψης ενός ερευνητή να είναι μικρότερη ίση της τρέχουσας ημερομηνίας (πριν την εισαγωγή στο researchers) και η ημερομηνία αξιολόγησης της ημερομηνίας έναρξης (πριν την εισαγωγή στο projects).

```
#Έλεγχος η ημέρα πρόσληψης ενός ερευνητή να μην είναι μεγαλύτερη της τρέχουσας ημερομηνίας
DELIMITER $$
CREATE TRIGGER check_date_of_recruitment
BEFORE INSERT ON RESEARCHERS
FOR EACH ROW
BEGIN
IF
(NEW.date_of_recruitment >CURRENT_DATE()) THEN
SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning: Recruitment date cant be greater than current date!';
END IF;
END$$
    DELIMITER ;
  #Η ημερομηνία της αξιολόγησης πρέπει να είναι μικρότερη της ημερομηνίας έναρξης του έργου
DELIMITER $$
CREATE TRIGGER check_evaluation_date
BEFORE INSERT ON PROJECTS
FOR EACH ROW
BEGIN
SET @eval_date=(SELECT evaluation_date FROM EVALUATIONS WHERE evaluation_id=NEW.evaluation_id);
IF (@eval_date > NEW.start_date) THEN
SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning: Evaluation date should be prior start date of project!';
END IF;
END$$
    DELIMITER ;
```

Δημιουργία trigger που ελέγχει, πριν την εισαγωγή στο deliverables, αν η ημερομηνία παράδοσης των παραδοτέων είναι έγκυρη.

```
#Η ημερομηνία παράδοσης του κάθε παραδοτέου πρέπει να είναι μεταξύ ημερομηνίας έναρξης του έργου και ημερομηνία λήξης

DELIMITER $$

CREATE TRIGGER check_delivery_date
BEFORE INSERT ON DELIVERABLES
FOR EACH ROW
BEGIN
SET @start_date = (SELECT start_date FROM PROJECTS WHERE project_id=NEW.project_id);
SET @due_date = (SELECT due_date FROM PROJECTS WHERE project_id=NEW.project_id);
IF(@start_date>NEW.delivery_date) OR (@due_date<NEW.delivery_date) THEN
SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning: Delivery date should be between start date of project and due date ';
END IF;
END$$
    DELIMITER ;
```

Έλεγχος οι προϋπολογισμοί των εταιρειών να είναι θετικοί

```
#Οι προϋπολογισμοί να είναι θετικοί κάθε οργανισμού

ALTER TABLE RESEARCH_CENTRES
ADD CONSTRAINT check_1 CHECK(ministry_of_education>0),
ADD CONSTRAINT check_2 CHECK(private_activities>0);

ALTER TABLE UNIVERSITIES
ADD CONSTRAINT check_3 CHECK(ministry_of_education>0);

ALTER TABLE COMPANIES
ADD CONSTRAINT check_4 CHECK(own_funds>0);
```

Δημιουργία trigger ελέγχου, πριν την εισαγωγή στο projects, ώστε ο αξιολογήτης ενός έργου να μην ανήκει στον οργανισμό που το διαχειρίζεται, γιατί κάτι τέτοιο δεν είναι ηθικά σωστό.

```
#Ο αξιολογητής δεν μπορεί να αξιολογεί πρότζεκτ του οργανισμού του

DELIMITER $$
CREATE TRIGGER check_for_evaluator_and_organization
BEFORE INSERT ON PROJECTS
FOR EACH ROW
BEGIN
SET @org_of_evaluator = (SELECT R.organization_id FROM RESEARCHERS R JOIN EVALUATES E WHERE (E.project_id=new.project_id and R.researcher_id = E.researcher_id));
IF(NEW.organization_id = @org_of_evaluator) THEN
SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning : Researcher cant evaluate project of his/hers organization';
END IF;
END $$

DELIMITER ;;
```

Δημιουργία trigger ελέγχου, πριν την εισαγωγή στο projects, ώστε ο επιστημονικός υπεύθυνος να ανήκει στον οργανισμό που διαχειρίζεται το πρότζεκτ.

```
#Ο επιστημονικός υπεύθυνος πρέπει να δουλεύει σε πρότζεκτ του οργανισμού του

DELIMITER $$
CREATE TRIGGER check_advisor_organization
BEFORE INSERT ON PROJECTS
FOR EACH ROW
BEGIN
SET @org_id = (SELECT organization_id FROM RESEARCHERS WHERE researcher_id = NEW.researcher_id);
IF(NEW.organization_id<>@org_id) THEN
SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning : Scientific advisor has to work at project of their organization !';
END IF;
END$$
DELIMITER ;;
```

Δημιουργία triggers που ελέγχουν, ο επιστημονικός υπεύθυνος να μην τοποθετείται και στο works_on (για το **ίδιο project)** και ένας ερευνητής να δουλεύει μόνο σε έργα του οργανισμού του(**έλεγχος στο project**).

```
# Ο επιστημονικός υπεύθυνος να μην τοποθετείται και στον πίνακα works_on
DELIMITER $$
CREATE TRIGGER check_for_advisor_and_worker
BEFORE INSERT ON PROJECTS
FOR EACH ROW
BEGIN
IF(EXISTS(SELECT * FROM WORKS_ON WHERE (NEW.researcher_id=researcher_id and project_id = new.project_id))) then
SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning : Scientific advisor cant also work on the project !';
END IF;
END $$
DELIMITER ;

#Ένας ερευνητής μπορεί να δουλεύει μόνο σε πρότζεκτ του οργανισμού του
DELIMITER $$
CREATE TRIGGER check_worker_organization
BEFORE INSERT ON PROJECTS
FOR EACH ROW
BEGIN
IF(EXISTS(SELECT * FROM WORKS_ON W JOIN RESEARCHERS R WHERE
 (R.researcher_id=W.researcher_id and W.project_id = new.project_id and R.organization_id <> new.organization_id))) THEN
SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning : Worker has to work on project of his organization !';
END IF;
END $$
DELIMITER ;;
```

Έλεγχος, πριν το update στο projects, ώστε τα νέα start και due dates να μην είναι μετά τις ημερομηνίες των παραδοτέων και έλεγχος στο works_on ώστε ο ερευνητής να δουλεύει μόνο σε έργα του οργανισμού του.

```
#Το νεό start date δε θα πρέπει να είναι μεγαλύτερο του delivery date και το νέο due date να μην είανι μεγαλύτερο
DELIMITER $$
CREATE TRIGGER check_alter_dates_projects
BEFORE UPDATE ON PROJECTS
FOR EACH ROW
BEGIN
IF(EXISTS(SELECT * FROM DELIVERABLES WHERE (project_id = new.project_id AND (new.start_date>delivery_date or new.due_date<delivery_date)))) THEN
SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning : Cant update start date or due date due to deliverable date';
END IF;
END $$
DELIMITER ;;
#Έλεγχος στο works_on ώστε ένας ερευνητής να δουλεύει μόνο σε project του οργανισμού του
DELIMITER $$
CREATE TRIGGER check_insert_new_working_relationship
BEFORE INSERT ON WORKS_ON
FOR EACH ROW
BEGIN
SET @org_id = (SELECT organization_id FROM RESEARCHERS WHERE new.researcher_id = researcher_id);
IF(EXISTS(SELECT * FROM PROJECTS WHERE (project_id = new.project_id AND organization_id <> @org_id))) THEN
SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning : Researcher cant work on a project not of his organization';
END IF;
END $$
DELIMITER ;;
```

Έλεγχος προϋποθέσεων διαγραφής ερευνητή. Αυτός μπορεί να διαγραφεί μόνο αν δεν είναι κάπου αξιολογητής ή αν δεν είναι επιστημονικός υπεύθυνος ή αν δεν είναι ο μοναδικός απλός ερευνητής.

```
#Έλεγχος προϋποθέσεων ώστε να διαγραφεί ένας ερευνητής

DELIMITER $$
CREATE TRIGGER check_delete_researcher
BEFORE DELETE ON RESEARCHERS
FOR EACH ROW
BEGIN
SET @org_id = (SELECT organization_id FROM RESEARCHERS WHERE researcher_id = OLD.researcher_id);
IF(EXISTS(SELECT * FROM EVALUATES WHERE researcher_id = OLD.researcher_id) OR EXISTS
(SELECT * FROM PROJECTS P WHERE(P.organization_id=@org_id AND P.researcher_id=OLD.researcher_id))
OR EXISTS (SELECT count(A.r_id) as C FROM
(SELECT W.project_id as p_id, W.researcher_id as r_id from WORKS_ON W ) AS A
INNER JOIN
(SELECT project_id as p_id FROM WORKS_ON WHERE researcher_id = OLD.researcher_id) AS B
ON A.p_id = B.p_id GROUP BY A.p_id HAVING C = 1))
THEN
SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning : Researcher is evaluator or scientific advisor or a single worker';
END IF;
END $$
```

Δημιουργούμε τα παρακάτω indexes, προκειμένου να βελτιωθεί η ταχύτητα της βάσης, με κριτήριο τη συχνότητα που κάποια attributes εμφανίζονται στις λειτουργίες της.

```
USE DB_PROJECT;

CREATE INDEX idx_start_date ON PROJECTS(start_date);
CREATE INDEX idx_due_date ON PROJECTS(due_date);
CREATE INDEX idx_title ON PROJECTS(title);
CREATE INDEX idx_first_name ON RESEARCHERS(first_name);
CREATE INDEX idx_last_name ON RESEARCHERS(last_name);

```

Οι λειτουργίες drop database και insert into database επισυνάπτονται στο git repo.

Επίσης έχουν δημιουργηθεί stored procedures τα οποία κι αυτά επισυνάπτονται στο git repo.

QUERIES

Για όλα τα queries έχουμε βάλει set@var=user_input, ώστε να μπορούμε αυτά να τα δοκιμάζουμε (και να τρέχουν) στη mysql. Αυτό αλλάζει στην python, όπου το set@var αντικαθίσταται με %s.

QUERY 3.1.

Εύρεση όλων των προγραμμάτων που υπάρχουν στη βάση

```
USE DB_PROJECT;


/*εύρεση όλων των προγραμμάτων */
select program_id, program_name from programs order by program_id;
```

Στα παρακάτω ερωτήματα χρειάζεται να βρούμε όλους τους ερευνητές που δουλεύουν σε ένα έργο με βάση κάποιο από τα δοθέντα κριτήρια. Η εύρεση όλων των ερευνητών για ένα έργο γίνεται:

```
/*εύρεση όλων των ερευνητών που δουλεύον στο έργο που έχει επιλέξει ο χρήστης*/
SET @selected_project = 10;
SELECT P.researcher_id, R.first_name,R.last_name FROM PROJECTS P INNER JOIN RESEARCHERS R
WHERE (@selected_project=P.project_id AND R.researcher_id=P.researcher_id)
UNION
SELECT R.researcher_id, R.first_name, R.last_name FROM RESEARCHERS R INNER JOIN WORKS_ON W
ON (@selected_project=W.project_id AND W.researcher_id=R.researcher_id);
```

Εδώ επιλέγουμε να βρούμε όλα τα ενεργά έργα που υπάρχουν την ημερομηνία που δίνει ο χρήστης(Κριτήριο Ημερομηνία) .

```
/*εύρεση ενεργών έργων την ημερομηνιά που έχει επιλέξει ο χρήστης*/
SET @user_date = '2022-06-03';
SELECT project_id,title FROM PROJECTS WHERE (start_date < @user_date AND due_date > @user_date)
ORDER BY project_id;
```

Εδώ επιλέγουμε να βρούμε όλα τα έργα με βάση τη διάρκεια που έχει επιλέξει ο χρήστης (Κριτήριο Διάρκεια).

```
/*εύρεση όλων των έργων με διάρκεια που έχει επιλέξει ο χρήστης*/
SET @project_duration= DATE_SUB(CURRENT_DATE(),INTERVAL -2 YEAR) - CURRENT_DATE() ;
SELECT project_id, title FROM PROJECTS WHERE (due_date-start_date)<@project_duration
order by project_id;
```

Εδώ επιλέγουμε να βρούμε όλα τα έργα που διαχειρίζεται ένας στέλεχος(Κριτήριο Διάρκεια).

```
/*εύρεση έργων που διαχειρίζεται ένας στέλεχος*/
SET @exec_proj = 2;
SELECT project_id, title FROM PROJECTS WHERE executive_id=@exec_proj
order by project_id;
```

QUERY 3.2

Δημιουργία δύο όψεων της βάσης

1η όψη: Έργα που δουλεύει ο καθένας ερευνητής

```
/* Δημιουργία όψης που εμφανίζει τα έργα που δουλεύει ένας ερευντής*/

DROP VIEW if exists proj_research;
CREATE VIEW proj_research AS
 select p.project_id, r.researcher_id, p.title, r.first_name, r.last_name
 from projects p INNER JOIN researchers r on p.researcher_id=r.researcher_id
 union
 select p.project_id, r.researcher_id, p.title, r.first_name, r.last_name
 from projects p inner join works_on w on p.project_id=w.project_id
 inner join researchers r on w.researcher_id=r.researcher_id
 order by researcher_id;

 select first_name, last_name, title from proj_research;
```

2η όψη: Έργα που έχει αναλάβει ο καθένας οργανισμός

```
USE DB_PROJECT;

#Δημιουργία όψης της βάσης που δείχνει τα έργα που έχει αναλάβει κάθε οργανισμός

DROP VIEW IF EXISTS proj_org;
CREATE VIEW proj_org AS
 select p.project_id, o.organization_id, p.title, o.org_name
 from projects p inner join organizations o on p.organization_id=o.organization_id
 order by o.organization_id;

select  org_name, title from proj_org
```

QUERY 3

Ο χρήστης επιλέγει ένα επιστημονικό πεδίο που έχει ιδιαίτερο ενδιαφέρον και του επιστρέφονται τα έργα που χρηματοδοτούνται σ' αυτό το πεδίο και τους ερευνητές που δουλεύουν σ' αυτό (έχουμε φτιάξει τη βάση ώστε να υπάρχουν μόνο εγκεκριμένα έργα).

```
#έργα που χρηματοδοτούνται σε ένα επιστημονικό πεδίο τον τελευταίο χρόνο (έχουμε μόνο εγκεκριμένα έργα)

SET @selected_field = 2;

SELECT P.project_id, P.title FROM PROJECTS P INNER JOIN DESCRIBES D
ON (P.project_id=D.project_id AND D.field_id=@selected_field)
WHERE (P.start_date<CURRENT_DATE() AND P.due_date>CURRENT_DATE())
ORDER BY p.project_id;

#ερευνητές που δουλεύουν σ'αυτό τον τελευταίο χρόνο (έχουμε μόνο εγκεκριμένα)

SELECT P.project_id, R.researcher_id, R.first_name, R.last_name FROM PROJECTS P INNER JOIN DESCRIBES D
ON (P.project_id=D.project_id AND D.field_id=@selected_field) INNER JOIN RESEARCHERS R
ON P.researcher_id = R.researcher_id WHERE (P.start_date<CURRENT_DATE() AND P.due_date>CURRENT_DATE())
UNION
SELECT P.project_id, R.researcher_id, R.first_name, R.last_name FROM PROJECTS P INNER JOIN DESCRIBES D
ON (P.project_id=D.project_id AND D.field_id=@selected_field)
INNER JOIN WORKS_ON W ON P.project_id=W.project_id INNER JOIN RESEARCHERS R
ON W.researcher_id=R.researcher_id WHERE (P.start_date<CURRENT_DATE() AND P.due_date>CURRENT_DATE())
Order by project_id;
```

QUERY 4

Επιστρέφονται όλοι οι ερευνητές που έχουν τον ίδιο αριθμό έργων σε διάστημα δύο συνεχόμενων ετών με τουλάχιστον 10 το χρόνο. Στη βάση μας και με κατάλληλη φόρμα από το ui, ο αριθμός 10 έχει παραμετρικοποιηθεί και μπορεί να οριστεί από το χρήστη, λόγω μη επάρκειας δεδομένων (στο query είναι ακριβώς όπως ζητείται με το 10).

```
select l.con1 as organization_name, l.cy2 as first_year, l.cy1 as second_year, l.c1 as no_of_projects from
(SELECT  cust.o_name as con1, cust.y2 as cy2, cust.y1 as cy1, count(distinct cust.p1_id) AS c1
from (select p1.project_id as p1_id ,p2.project_id as p2_id, p1.organization_id as o1_id, o.org_name as o_name,  YEAR(p2.start_date) as y2,YEAR(p1.start_date) as y1
 from projects p1 inner join projects p2 on p1.organization_id=p2.organization_id
and p1.project_id<>p2.project_id and (YEAR(p1.start_date)-YEAR(p2.start_date))=1 inner join organizations o on o.organization_id=p1.organization_id) cust
GROUP BY cust.o1_id, cust.y2,cust.y1) as l
inner join
(SELECT  cust.o_name as con2, cust.y2 as cy21, cust.y1 as cy11, count(distinct cust.p2_id) AS c2
from (select p1.project_id as p1_id ,p2.project_id as p2_id, p1.organization_id as o1_id, o.org_name as o_name,  YEAR(p2.start_date) as y2,YEAR(p1.start_date) as y1
 from projects p1 inner join projects p2 on p1.organization_id=p2.organization_id
and p1.project_id<>p2.project_id and (YEAR(p1.start_date)-YEAR(p2.start_date))=1 inner join organizations o on o.organization_id=p1.organization_id) cust
 GROUP BY cust.o1_id, cust.y2, cust.y1) as m
 on l.c1=m.c2 and l.con1=m.con2 and l.cy2=m.cy21 and l.cy1=m.cy11 and l.c1>=10 order by l.c1 DESC
```

QUERY 5

Επιλέγονται τα 3 κορυφαία σε συχνότητα εμφάνισης ζεύγη επιστημονικών πεδίων

```
#Τα 3 κορυφαία ζεύγη επιστημονικών πεδίων που εμφανίστηκαν σε έργα

select S.field_name, F.field_name, cust.counter from (
SELECT A.field_id as newField_A, B.field_id as newField_B ,B.project_id, count(*) as counter FROM DESCRIBES A INNER JOIN DESCRIBES B
ON (A.project_id = B.project_id AND  A.field_id<B.field_id AND A.description_id <> B.description_id)
GROUP BY A.field_id, B.field_id ORDER BY counter DESC limit 3) cust
inner join scientific_fields S
on S.field_id= cust.newField_A
inner join scientific_fields F on F.field_id=cust.newField_B
order by cust.counter DESC;
```

QUERY 6

Επιστρέφονται όλοι οι κάτω των 40 ταξινομημένοι σε φθίνουσα σειρά ως προς τον αριθμό των έργων

```
# Όλοι οι ερευνητές κάτω των 40 που δουλεύουν στα περισσότερα έργα

select cust.r_id, cust.r_fn, cust.r_ln, cust.r_pr, sum(cust.project_count) as s_pr from
(SELECT R.researcher_id as r_id, R.first_name as r_fn, R.last_name as r_ln, W.project_id as r_pr, count(*) as project_count FROM RESEARCHERS R
INNER JOIN WORKS_ON W ON (R.researcher_id = W.researcher_id AND
DATE_SUB(R.date_of_birth,INTERVAL -40 YEAR)> CURRENT_DATE())
INNER JOIN PROJECTS P ON P.project_id=W.project_id and P.start_date<CURRENT_DATE() AND
P.due_date>CURRENT_DATE()
GROUP BY R.researcher_id
UNION
SELECT R.researcher_id as r_id, R.first_name as r_fn, R.last_name as r_ln, P.project_id as r_pr, count(*) project_count FROM RESEARCHERS R
INNER JOIN PROJECTS P ON R.researcher_id = P.researcher_id AND
DATE_SUB(R.date_of_birth,INTERVAL -40 YEAR)> CURRENT_DATE()
AND P.start_date<CURRENT_DATE() AND P.due_date>CURRENT_DATE()
GROUP BY R.researcher_id ) as cust
group by (cust.r_id) order BY s_pr DESC ;
```

QUERY 7

Βρίσκονται τα στελέχη τα οποία έχουν δώσει αθροιστικά το μεγαλύτερο ποσό σε εταιρείες.

```
#Top 5 στελέχη που έχουν δώσει τη μεγαλύτερη χρηματοδότηση σε εταιρεία (company)

SELECT E.executive_id, E.executive_first_name, E.executive_last_name, O.org_name, SUM(P.funding) Total_funds
FROM EXECUTIVES E INNER JOIN PROJECTS P ON E.executive_id = P.executive_id INNER JOIN ORGANIZATIONS O
ON P.organization_id = O.organization_id INNER JOIN COMPANIES C ON C.organization_id=O.organization_id
GROUP BY E.executive_id ORDER BY Total_funds DESC
limit 5;
```

QUERY 8

Στη βάση μας, ότι “κανένα παραδοτέο” έχουν τα έργα που δεν έχουν πάρει έγκριση. Όπως προαναφέρθηκε εμείς δεν εισάγουμε μη εγκεκριμένα έργα. Άρα επειδή δεν είχε νόημα να μην επιστρέφουμε κάποιο αποτέλεσμα θεωρούμε ότι πάντα υπάρχει ένα παραδοτέο(πρέπει να εισάγεται στο UI) όπου η ημερομηνία παράδοσής του είναι ίδια με την ημέρα λήξης του έργου.

```
#Οι ερευνητές που εργάζονται σε 5 ή περισσότερα έργα που δεν έχουν παραδοτέα(όπου αυτο μεταφράζεται σε ένα παραδοτέο)

SELECT cust.researcher_id, cust.first_name, cust.last_name, count(cust.project_id) as counter from(
SELECT R.researcher_id, P.project_id, R.first_name, R.last_name,
count(*) AS deliverable_counter
FROM RESEARCHERS R INNER JOIN WORKS_ON W ON
R.researcher_id = W.researcher_id INNER JOIN PROJECTS P ON W.project_id = P.project_id
INNER JOIN DELIVERABLES D ON D.project_id=P.project_id
GROUP BY R.researcher_id, P.project_id
HAVING deliverable_counter = 1
UNION SELECT R.researcher_id, P.project_id, R.first_name, R.last_name,
count(*) as deliverable_counter
FROM RESEARCHERS R INNER JOIN PROJECTS P
ON P.researcher_id=R.researcher_id INNER JOIN DELIVERABLES D
ON D.project_id=P.project_id
GROUP BY R.researcher_id, P.project_id
HAVING deliverable_counter=1 ) cust
GROUP BY cust.researcher_id HAVING counter>=5 order by counter DESC, researcher_id;
```

ΟΔΗΓΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ

Για την λειτουργία της εφαρμογής μας θα πρέπει στον ίδιο φάκελο να βρίσκεται το πρόγραμμα της python MyApp.py και ένας φάκελος με το όνομα template μέσα στον οποίο θα πρέπει να βρίσκονται όλα τα html files. Στο κομμάτι της python θα πρέπει να περιέχονται οι εξής βιβλιοθήκες στα αντίστοιχα version τους.

Package	Version
certifi	2022.5.18.1
charset-normalizer	2.0.12
click	8.1.3
colorama	0.4.4
connection	2021.7.20
Flask	2.1.2
Flask-MySQLdb	1.0.1
idna	3.3
importlib-metadata	4.11.4
itsdangerous	2.1.2
Jinja2	3.1.2
MarkupSafe	2.1.1
mysqlclient	2.1.0
pip	22.1.2
requests	2.27.1
setuptools	62.3.2
typing_extensions	4.2.0
urllib3	1.26.9
Werkzeug	2.1.2
zipp	3.8.0

Για την εγκατάσταση συγκεκριμένου version του package χρησιμοποιούμε την εντολή pip install ‘library_name == version’ πχ pip install ‘Flask ==2.1.2 ’

Για το στήσιμο της βάσης συνιστούμε την εγκατάσταση του xampp για να κάνει launch τον sql server μας, πατώντας start δίπλα από την επιλογή MySQL. Επίσης απαιτείται η εγκατάσταση του MySQL Workbench μέσω του οποίου θα συνδεθούμε στον server με username = root και κενό password. Μετά την σύνδεση μας στο workbench ο χρήστης θα πρέπει να τρέξει διαδοχικά τα εξής sql scripts :

DB_CREATION.sql

DB_TABLE_CREATION_NEW.sql

DB_CONSTRAINTS.sql

DB_DATA_CREATION.sql

DB_INDEXES.sql

Επίσης θα χρειαστεί να πατήσουμε στο NAVIGATOR TAB/SCHEMAS στα stored procedures-> add stored procedure και να προσθέσουμε τα εξής procedures με ακριβώς τα εξής ονόματα :

31_duration

32_proj_org

32_proj_research

34_question

38_question

Τα οποία βρίσκονται στα ομώνυμα .txt αρχεία.

Για να τρέξουμε την εφαρμογή όντας μέσα στον φάκελο που έχει τον template φάκελο και το python file τρέχουμε τις εντολές pip install flask, pip install flask_mysqldb. Τέλος τρέχουμε την εντολή python –m flask run και πηγαίνουμε σε οποιονδήποτε browser στο url http://localhost:3000.