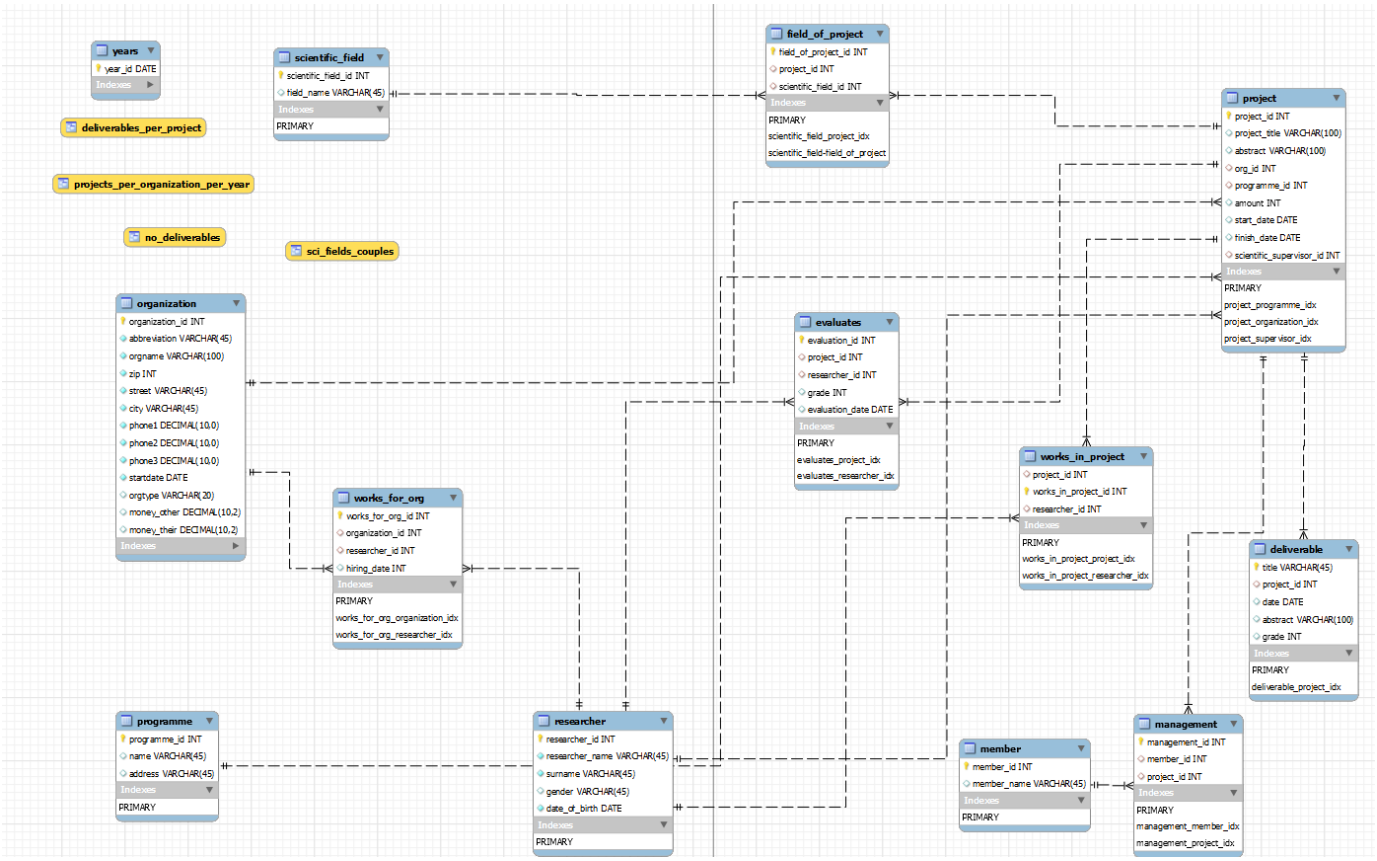
Βάσεις Δεδομένων

Αναφορά Εξαμηνιαίας Εργασίας

Νικόλαος Σφακιανάκης (el19130)Ανδρέας Χαραλαμπόπουλος (el19134)

Νικόλαος Χατζής (e19117)

1. Σχεσιακό Διάγραμμα



1.1 Στην αριστερή πάνω γωνία φαίνονται views που δημιουργήθηκαν για την απάντηση των ερωτημάτων.

1.2 Για τη δημιουργία των παραπάνω πινάκων στην SQL απαιτήσαμε να μην μένουν χωρίς τιμή (NULL) μερικά βασικά πεδία, όπως primary/foreign keys.
Όπως φαίνεται και στο αρχείο Tables_2.sqlχρησιμοποιήθηκαν οι παρακάτω εντολές για τη δημιουργία της βάσης και των πινάκων :

```
-- -----------------------------------------------------
-- Schema Project
-- -----------------------------------------------------
```

```
4  CREATE SCHEMA IF NOT EXISTS `Project` ;
5  USE `Project` ;
6
7  -- -----------------------------------------------------
8  -- Table `Project`.`programme`
9  -- -----------------------------------------------------
10 CREATE TABLE IF NOT EXISTS `Project`.`programme` (
11   `programme_id` INT NOT NULL AUTO_INCREMENT,
12   `name` VARCHAR(45) NULL,
13   `address` VARCHAR(45) NULL,
14   PRIMARY KEY (`programme_id`))
15 ENGINE = InnoDB;
16
17
18 -- -----------------------------------------------------
19 -- Table `Project`.`organization`
20 -- -----------------------------------------------------
21 CREATE TABLE IF NOT EXISTS `Project`.`organization` (
22   `organization_id` INT NOT NULL AUTO_INCREMENT,
23   `abbreviation` VARCHAR(45) NOT NULL,
24   `orgname` VARCHAR(100) NOT NULL,
25   `zip` INT NOT NULL,
26   `street` VARCHAR(45) NOT NULL,
27   `city` VARCHAR(45) NOT NULL,
28   `phone1` NUMERIC(10,0) NOT NULL DEFAULT -1,
29   `phone2` NUMERIC(10,0) NOT NULL DEFAULT -1,
30   `phone3` NUMERIC(10,0) NOT NULL DEFAULT -1,
31   `startdate` DATE NOT NULL,
32   `orgtype` VARCHAR(20) ,
33   `money_other` NUMERIC(10,2) NULL DEFAULT -1,
34   `money_their` NUMERIC(10,2) NULL DEFAULT -1,
35   PRIMARY KEY (`organization_id`),
36   CONSTRAINT chk_type CHECK (orgtype IN  ('                    ',
       '                              ', '                ')) -- mesa
     stis parentheseis einai oi leksis Panepistimio, Ereynitiko kentro
     , Etaireia
37   -- alla sta ellhnika kai den emfanizontai sto pdf
38   )
39 ENGINE = InnoDB;
40
41
42 -- -----------------------------------------------------
43 -- Table `Project`.`researcher`
44 -- -----------------------------------------------------
45 CREATE TABLE IF NOT EXISTS `Project`.`researcher` (
46   `researcher_id` INT NOT NULL AUTO_INCREMENT,
47   `researcher_name` VARCHAR(45) NOT NULL,
48   `surname` VARCHAR(45) NOT NULL,
49   `gender` VARCHAR(45) NULL,
50   `date_of_birth` DATE NOT NULL,
51   PRIMARY KEY (`researcher_id`))
52 ENGINE = InnoDB;
53
54
```

```
55 -- -----------------------------------------------------
56 -- Table `Project`.`project`
57 -- -----------------------------------------------------
58 CREATE TABLE IF NOT EXISTS `Project`.`project` (
59   `project_id` INT NOT NULL AUTO_INCREMENT,
60   `project_title` VARCHAR(100) NULL,
61   `abstract` VARCHAR(100) NULL,
62   `org_id` INT NULL,
63   `programme_id` INT NULL,
64   `amount` INT NULL,
65   `start_date` DATE NULL,
66   `finish_date` DATE NULL,
67   `scientific_supervisor_id` INT NULL,
68   CONSTRAINT CHK_Duration CHECK (DATEDIFF(finish_date,start_date)>
      364 and DATEDIFF(finish_date,start_date)< 1460),
69   CONSTRAINT CHK_Amount CHECK (amount>=100000 and amount <= 1000000),
70   PRIMARY KEY (`project_id`),
71   INDEX `project_programme_idx` (`programme_id` ASC) VISIBLE,
72   INDEX `project_organization_idx` (`org_id` ASC) VISIBLE,
73   INDEX `project_supervisor_idx` (`scientific_supervisor_id` ASC)
      INVISIBLE,
74   CONSTRAINT `project_programme`
75     FOREIGN KEY (`programme_id`)
76     REFERENCES `Project`.`programme` (`programme_id`)
77     ON DELETE CASCADE
78     ON UPDATE CASCADE,
79   CONSTRAINT `project_organization`
80     FOREIGN KEY (`org_id`)
81     REFERENCES `Project`.`organization` (`organization_id`)
82     ON DELETE CASCADE
83     ON UPDATE CASCADE,
84   CONSTRAINT `project_supervisor`
85     FOREIGN KEY (`scientific_supervisor_id`)
86     REFERENCES `Project`.`researcher` (`researcher_id`)
87     ON DELETE CASCADE
88     ON UPDATE CASCADE)
89 ENGINE = InnoDB;
90
91
92 -- -----------------------------------------------------
93 -- Table `Project`.`member`
94 -- -----------------------------------------------------
95 CREATE TABLE IF NOT EXISTS `Project`.`member` (
96   `member_id` INT NOT NULL AUTO_INCREMENT,
97   `member_name` VARCHAR(45) NULL,
98   PRIMARY KEY (`member_id`))
99 ENGINE = InnoDB;
100
101
102 -- -----------------------------------------------------
103 -- Table `Project`.`deliverable`
104 -- -----------------------------------------------------
105 CREATE TABLE IF NOT EXISTS `Project`.`deliverable` (
106   `deliverable_id` INT NOT NULL auto_increment,
```

```
  `title` VARCHAR(45) NOT NULL,
  `project_id` INT NULL,
  `date` DATE NULL,
  `abstract` VARCHAR(100) NULL,
  `grade` INT NULL,
  PRIMARY KEY (`deliverable_id`),
  INDEX `deliverable_project_idx` (`project_id` ASC) VISIBLE,
  CONSTRAINT `deliverable_project`
    FOREIGN KEY (`project_id`)
    REFERENCES `Project`.`project` (`project_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Project`.`scientific_field`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Project`.`scientific_field` (
  `scientific_field_id` INT NOT NULL AUTO_INCREMENT,
  `field_name` VARCHAR(45) NULL,
  PRIMARY KEY (`scientific_field_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Project`.`field_of_project`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Project`.`field_of_project` (
  `field_of_project_id` INT NOT NULL AUTO_INCREMENT,
  `project_id` INT NULL,
  `scientific_field_id` INT NULL,
  PRIMARY KEY (`field_of_project_id`),
  INDEX `scientific_field_project_idx` (`project_id` ASC) VISIBLE,
  CONSTRAINT `scientific_field-field_of_project`
    FOREIGN KEY (`scientific_field_id`)
    REFERENCES `Project`.`scientific_field` (`scientific_field_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `field_project`
    FOREIGN KEY (`project_id`)
    REFERENCES `Project`.`project` (`project_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Project`.`management`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Project`.`management` (
  `management_id` INT NOT NULL AUTO_INCREMENT,
  `member_id` INT NULL,
  `project_id` INT NULL,
```

```
  PRIMARY KEY (`management_id`),
  INDEX `management_member_idx` (`member_id` ASC) INVISIBLE,
  INDEX `management_project_idx` (`project_id` ASC) VISIBLE,
  CONSTRAINT `management_member`
    FOREIGN KEY (`member_id`)
    REFERENCES `Project`.`member` (`member_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `management_project`
    FOREIGN KEY (`project_id`)
    REFERENCES `Project`.`project` (`project_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Project`.`evaluates`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Project`.`evaluates` (
  `evaluation_id` INT NOT NULL AUTO_INCREMENT,
  `project_id` INT NULL,
  `researcher_id` INT NULL,
  `grade` INT NULL,
  `evaluation_date` DATE NULL,
  PRIMARY KEY (`evaluation_id`),
  INDEX `evaluates_project_idx` (`project_id` ASC) VISIBLE,
  INDEX `evaluates_researcher_idx` (`researcher_id` ASC) VISIBLE,
  CONSTRAINT `evaluates_project`
    FOREIGN KEY (`project_id`)
    REFERENCES `Project`.`project` (`project_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `evaluates_researcher`
    FOREIGN KEY (`researcher_id`)
    REFERENCES `Project`.`researcher` (`researcher_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Project`.`works_in_project`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Project`.`works_in_project` (
  `project_id` INT NULL,
  `works_in_project_id` INT NOT NULL AUTO_INCREMENT,
  `researcher_id` INT NULL,
  PRIMARY KEY (`works_in_project_id`),
  INDEX `works_in_project_project_idx` (`project_id` ASC) INVISIBLE,
  INDEX `works_in_project_researcher_idx` (`researcher_id` ASC)
   VISIBLE,
  CONSTRAINT `works_in_project_project`
    FOREIGN KEY (`project_id`)
```

```
214     REFERENCES `Project`.`project` (`project_id`)
215     ON DELETE CASCADE
216     ON UPDATE NO ACTION,
217   CONSTRAINT `works_in_project_researcher`
218     FOREIGN KEY (`researcher_id`)
219     REFERENCES `Project`.`researcher` (`researcher_id`)
220     ON DELETE CASCADE
221     ON UPDATE CASCADE)
222 ENGINE = InnoDB;
223
224
225 -- -----------------------------------------------------
226 -- Table `Project`.`works_for_org`
227 -- -----------------------------------------------------
228 CREATE TABLE IF NOT EXISTS `Project`.`works_for_org` (
229   `works_for_org_id` INT NOT NULL AUTO_INCREMENT,
230   `organization_id` INT NULL,
231   `researcher_id` INT NULL,
232   `hiring_date` INT NULL,
233   PRIMARY KEY (`works_for_org_id`),
234   INDEX `works_for_org_organization_idx` (`organization_id` ASC)
      VISIBLE,
235   INDEX `works_for_org_researcher_idx` (`researcher_id` ASC) VISIBLE,
236   CONSTRAINT `works_for_org_org`
237     FOREIGN KEY (`organization_id`)
238     REFERENCES `Project`.`organization` (`organization_id`)
239     ON DELETE CASCADE
240     ON UPDATE CASCADE,
241   CONSTRAINT `works_for_org_researcher`
242     FOREIGN KEY (`researcher_id`)
243     REFERENCES `Project`.`researcher` (`researcher_id`)
244     ON DELETE CASCADE
245     ON UPDATE CASCADE)
246 ENGINE = InnoDB;
247 -- -----------------------------------------------------
248 -- Table `Project`.`years`
249 -- -----------------------------------------------------
250 CREATE TABLE IF NOT EXISTS `Project`.`years` (
251   `year_id` DATE NOT NULL,
252   PRIMARY KEY (`year_id`))
253 ENGINE = InnoDB;
```

Σχόλια:

(α′) Στους οργανισμούς αποφασίστηκε ότι αρκεί να μπορούν να αποθηκευτούν μέχρι 3 τηλέφωνα, αν υπάρχουν μόνο 2 το 3ο αρχικοποιείται σε τιμή -1 αυτόματα.

(β′) Οι κατηγορίες οργανισμών έγιναν πεδίο του πίνακα οργανισμών, αφού η μόνη διαφορά ήταν στον τίτλο των προϋπολογισμών της κάθε κατηγορίας και στο ότι τα ερευνητικά κέντρα είχαν 2 προϋπολλογισμούς. Αυτή η απαίτηση καλύφθηκε με τη δημιουργία ενός πεδίου που να υποδεικνύει το είδος του οργανισμού και ανάλογα με το είδος το δεύτερο πεδίο προϋπολλογισμού είτε αρχικοποιείται σε -1 είτε παίρνει την τιμή που του δίνει ο πελάτης.

(γ′) Προσθέσαμε 2 CHECKS στον πίνακα των έργων για να σιγουρευτούμε ότι οι ημερομηνίες βγάζουν νόημα και ότι το ποσό επιχορήγησης είναι εντός τον αποδεκτών ορίων.

1.3 Για τα ερωτήματα του 3ου μέρους χρησιμοποιήσαμε τα εξής queries - views

3.1 Το πρώτο ερώτημα υλοποιήθηκε στο flask χωρίς τη χρήση κάποιου view με το my_query1 στο οποίο προστίτεθεται και μία κατάληξη (το mquery) για το στέλεχος αν έχει επιλεχθεί κάποιο στέλεχος από τον χρήστη.

```
    mquery = 'AND member.member_name = %s'

#end_query = """GROUP BY project.id"""

my_query1 = """ SELECT project.project_title, project.abstract, project.start_date,
project.finish_date, member.member_name, project.amount, project.project_id FROM
project LEFT JOIN management ON
project.project_id = management.project_id LEFT JOIN
member ON management.member_id = member.member_id
WHERE project.start_date > %s AND project.finish_date < %s
AND DATEDIFF(project.finish_date,project.start_date)/365> %s
AND DATEDIFF(project.finish_date,project.start_date)/365 < %s
"""
```

3.2 Για το δεύτερο υποερώτημα φτίαξαμε το ζητούμενο view και ένα αντίστοιχο που δείχνει οργανισμούς ανά έργα/επιχορηγήσεις, όπως φαίνεται παρακάτω.

```
CREATE OR REPLACE VIEW organizations (organization_id, orgname, project_id, project_title )
AS
SELECT organization.organization_id , organization.orgname,project.project_id, project.project_title
 FROM organization INNER JOIN project ON project.org_id = organization.organization_id
 ORDER BY organization.orgname;

CREATE OR REPLACE VIEW researchers (researcher_id, researcher_name, surname, project_id, project_title )
AS
SELECT researcher.researcher_id, researcher.researcher_name , researcher.surname,project.project_id, project.project_title
 FROM researcher INNER JOIN works_in_project ON works_in_project.researcher_id = researcher.researcher_id
 INNER JOIN project ON project.project_id = works_in_project.project_id
 ORDER BY researcher_id
```

3.3 Για το τρίτο υποερώτημα παίρνουμε την απάντηση με το παρακάτω query όπου $\%s$ το id του επιστημονικού πεδίου που διάλεξε ο χρήστης.

```
query2 = """SELECT p.project_title, r.researcher_name, r.surname FROM
field_of_project fp INNER JOIN project p ON p.project_id = fp.project_id
INNER JOIN works_in_project wp ON p.project_id = wp.project_id
INNER JOIN researcher r ON wp.researcher_id = r.researcher_id
WHERE p.start_date < CURDATE() and p.finish_date> CURDATE() and fp.scientific_field_id = %s
ORDER BY p.project_title
"""
```

3.4 Για το 4ο υποερώτημα πρώτα κατασκευάσαμε το παρακάτω view

```
CREATE OR REPLACE VIEW projects_per_organization_per_year (organization_id, organization_name, projects, yearr)
AS
SELECT o.organization_id, o.orgname, COUNT(*), y.year_id as yearr
FROM organization o INNER JOIN project p ON o.organization_id = p.org_id
INNER JOIN years y ON DATEDIFF(y.year_id,p.start_date)>0 and DATEDIFF(y.year_id,p.finish_date)<0
GROUP BY o.organization_id, yearr;
```

και στη συνέχεια εκτελέσαμε το εξής query :

```
select o.organization_name, o.organization_id, o.projects, o.yearr FROM
projects_per_organization_per_year o INNER JOIN projects_per_organization_per_year y
ON o.organization_id = y.organization_id
WHERE o.projects=y.projects AND o.projects>=10 AND YEAR(o.yearr)-YEAR(y.yearr)=-1;
```

3.5 Για το 5ο υποερώτημα πρώτα κατασκευάσαμε το παρακάτω view

```
CREATE OR REPLACE VIEW sci_fields_couples (first_field, second_field, projects)
AS
SELECT s.scientific_field_id as first_field, ss.scientific_field_id as second_field, COUNT(*) as projects
FROM scientific_field s INNER JOIN scientific_field ss
INNER JOIN field_of_project  f ON (s.scientific_field_id = f.scientific_field_id or ss.scientific_field_id = f.scientific_field_id  )
WHERE s.scientific_field_id != ss.scientific_field_id
GROUP BY s.scientific_field_id, ss.scientific_field_id
ORDER BY projects DESC;
```

και στη συνέχεια εκτελέσαμε το εξής query :

```
SELECT * FROM sci_fields_couples;
```

3.6 Για το 6ο υποερώτημα παίρνουμε την απάντηση μέσω του παρακάτω query

```
SELECT a.researcher_name, a.surname, COUNT(b.project_id) FROM researcher a
INNER JOIN works_in_project b ON a.researcher_id = b.researcher_id
INNER JOIN project c ON c.project_id = b.project_id
WHERE c.finish_date > CURDATE() AND DATEDIFF(CURDATE(),a.date_of_birth) /365 < 40
GROUP BY a.researcher_id
ORDER BY COUNT(b.project_id) DESC;
```

3.7 Για το 7ο υποερώτημα παίρνουμε την απάντηση μέσω του παρακάτω query

```
SELECT c.orgname, b.member_id, SUM(a.amount)
FROM project a INNER JOIN management b  ON b.project_id = a.project_id
INNER JOIN organization c ON a.org_id = c.organization_id WHERE c.orgtype = "Εταιρεία"
group by a.org_id, b.member_id
ORDER BY SUM(a.amount) DESC;
```

3.8 Τέλος για το τελευταίο query πρώτα κατασκευάζουμε το παρακάτω view που παρουσιάζει τα έργα χωρίς παραδοτέα

```
CREATE OR REPLACE VIEW no_deliverables (project_id, project_title, finish_date, start_date )
AS
SELECT a.project_id, a.project_title, a.finish_date, a.start_date
FROM project a
WHERE a.project_id NOT IN
(SELECT c.project_id FROM deliverable c WHERE c.date>CURDATE());
```

και ύστερα εκτελούμε το παρακάτω query

```
SELECT r.researcher_id, r.researcher_name,r.surname, COUNT(b.project_id)
FROM researcher r INNER JOIN works_in_project w ON r.researcher_id = w.researcher_id
INNER JOIN no_deliverables b ON w.project_id = b.project_id
GROUP BY r.researcher_id
ORDER BY COUNT(b.project_id) DESC ;
```

1.4 Μιας και απαιτούταν CRUD για όλους τους πίνακες και μιας και για να υπάρχει απάντηση σε όλα τα ερωτήματα απαιτούταν μεγάλος αριθμός έργων και ερευνητών, αποφασίστηκε να προσθεθούν indexes σε όλα τα primary και foreign keys .

1.5 Για την διαχείριση και ανάπτυξη της βάσης χρησιμοποιήθηκες το MySQL Workbench και για το στήσιμο του web server χρησιμοποιήθηκε το Flask (Python)και για τη σύνδεση της βάσης και του σέρβερ χρησιμοποιήθηκε HTML, CSS, Java Script. Για την επικοινωνία frontend -backend χρησιμοποιούνται μέθοδοι GET και POST .

1.6 Τα βήματα εγκατάστασης είναι:

1 Εγκατάσταση MySQL Workbench

2 Φτίαχνουμε νέα σύνδεση στο MySQL Workbenchμε τα στοιχεία που επιθυμούμε και τρέχουμε το αρχείο Tables_2.sql

3 Εγκαθιστούμε τη βιβλίοθηκη Faker (Python) και το MySQL connector για Python και τρέχουμε το αρχείο generator.py αντικαθιστώντας τα στοιχεία της συνδεσής μας στο MySQL Workbench.

4 Εγκαθιστούμε τη βιβλιοθήκη Flask(Python) μέσω της εντολής pip install Flask και αντικαθιστούμε τα στοιχεία της σύνδεσής μας για το MySQL Workbench στο αρχείο __init__.py . Τέλος τρέχουμε το main.py και συνδεόμαστε στο localhost για να δούμε την σελίδα της βάσης.

1.7 https://github.com/Deiadara/BashDedomenwn