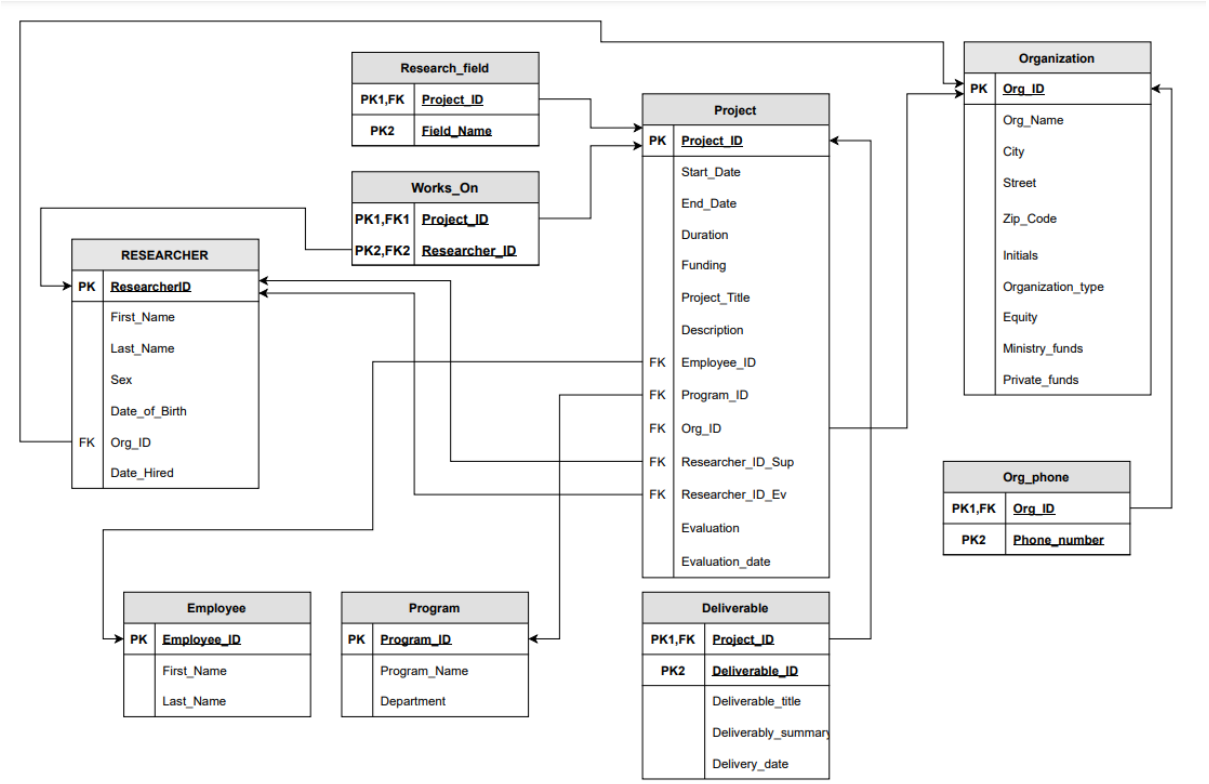
Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Βάσεις Δεδομένων
Εξαμηνιαία εργασία ,
Πέτρος Πανηγυράκης (ΑΜ 031119036)(3ο έτος)
Κωνσταντίνος Τσόκας(ΑΜ 03119073) (3ο έτος)
Οδυσσέας Κόνιας (ΑΜ 03119166) (3ο έτος)
Ομάδα 122

Το Ελληνικό Ίδρυμα Έρευνας και Καινοτομίας - ΕΛ.ΙΔ.Ε.Κ. είναι σημαντικός χρηματοδότης της ακαδημαϊκής έρευνας στην Ελλάδα. Οι ερευνητές υποβάλλουν ετησίως προτάσεις έργων σε διάφορα προγράμματα και αυτό, μέσω διαδικασίας αξιολόγησης, αποφασίζει ποια έργα θα χρηματοδοτηθούν. Αφού χρηματοδοτηθούν τα έργα (τα ποσά κυμαίνονται από 100.000€ έως 1.000.000€), ο οργανισμός που απασχολούνται οι ερευνητές διαχειρίζεται τα χρήματα για λογαριασμό των ερευνητών.
Ο Δ/ντης του ΕΛ.ΙΔ.Ε.Κ. σας κάλεσε να σχεδιάσετε και να υλοποιήσετε ένα σύστημα αποθήκευσης, διαχείρισης και ανάλυσης των πληροφοριών που συγκεντρώνονται από το ίδρυμα.

ΕΛ.ΙΔ.Ε.Κ.
Ελληνικό Ίδρυμα Έρευνας & Καινοτομίας

Αρχικά για την υλοποίηση της βάσης μας σχεδιάσαμε το relational diagram βασιζόμενοι στο ER diagram που είχαμε παραδώσει σε συνδυασμό με την προτεινόμενη λύση. Το αποτέλεσμα φαίνεται παρακάτω.



Επιπλέον ορίσαμε, ακολουθήσαμε και αργότερα εισάγαμε στην βάση μας ένα σύνολο κανόνων και περιορισμών:

1. Σε όλες τις σχέσεις έχει οριστεί ένα primary key για να μπορεί να γίνει σωστή διαχείριση.
2. Ο τύπος όλων των attributes είναι σαφώς ορισμένος.
3. Όλες οι N-1 σχέσεις του ER έχουν μετατραπεί σε attribute (Foreign key).
4. Όλες οι N-N σχέσεις του ER έχουν μετατραπεί σε έναν καινούργιο πίνακα που παραπέμπει στους δύο αρχικούς (πάλι μέσω Foreign key).

Οι παραπάνω κανόνες υλοποιήθηκαν στην sql κατα την δημιουργία των σχέσεων. Παρακάτω ακολουθούν περιορισμοί που αφορούν του περιορισμούς που τέθηκαν από τον πελάτη και άλλοι λογικοί περιορισμοί:

1. Κάθε έργο έχει έναν οργανισμό που το διαχειρίζεται ο οποίος δεν μπορεί να αλλάξει.
2. Κάθε έργο έχει έναν έναν ερευνητή που είναι ο επιστημονικός υπεύθυνος του.
3. Κάθε έργο έχει ελάχιστη διάρκεια 1 έτος - μέγιστη τα 4 έτη.
4. Το κάθε έργο προκειμένου να χρηματοδοτηθεί έχει αξιολογηθεί από έναν ερευνητή που δεν ανήκει στο δυναμικό του οργανισμού.
5. Τα ποσά χρηματοδότησης κυμαίνονται από 100.000€ έως 1.000.000€.
6. Το κάθε έργο λαμβάνει χρηματοδότηση από ένα πρόγραμμα.
7. Ο κάθε ερευνητής εργάζεται σε ένα (και μόνο) οργανισμό.
8. Το φύλο ενός ερευνητή μπορεί να πάρει τις τιμές 'male', 'female', 'other',

9. Οι οργανισμοί ανήκουν σε μια από τις 3 κατηγορίες:
   α) Πανεπιστήμια
   β) Ερευνητικά Κέντρα
   γ) Εταιρίες
10. Η ημερομηνία έναρξης ενός έργου (start_date) πρέπει χρονικά να είναι πριν από την ημερομηνία λήξης του (end_date)
11. Η αξιολόγηση του έργου μπορεί να έχει τιμές από 0-100.
12. Σε ένα έργο δουλεύουν μόνο ερευνητές του οργανισμού στον οποίο ανήκει το έργο.
13. Ο ερευνητής που αξιολόγησε ένα έργο δεν μπορεί να δουλεύει στον οργανισμό στον οποίο ανήκει το έργο και συνεπώς ούτε να δουλεύει σε αυτο.
14. Κάθε έργο έχει έναν υπάλληλο του ΕΛΙΔΕΚ να το ελέγχει.
15. Ο supervisor καθε έργου δουλεύει σε αυτό.
16. Όλα τα παραδοτέα πρέπει να υποβληθούν πριν την λήξη του έργου.

Όσα από τα παραπάνω δεν απαιτούν διασχεσιακό έλεγχο υλοποιούνται με απλά checks ενώ τα πιο πολύπλοκα με triggers πριν/μετα την εισαγωγή, επικαιροποίηση η διαγραφή δεδομένων.
Τέλος όσον αφορά τα χαρακτηριστικά των σχέσεων μερικά κρίνονται απαραίτητα για την εισαγωγή του στοιχείου στην βάση (όπως το όνομα ενός οργανισμού) ενώ άλλα δευτερεύοντα (όπως η διευθυνση τους οργανισμού). Στα πρώτα απαιτούμε να μην γίνεται δεκτή η τιμή “NULL” ενω στα δεύτερα όχι.

Η υλοποίηση της βάσης έγινε με sql. Ολόκληρος ο κώδικας βρίσκεται ανεβασμένος αλλα στην παρούσα αναφορά θα εξετάσουμε καίρια σημεία του για την επεξήγηση της λύσης μας.

Query για την δημιουργία της βάσης:

```sql
DROP DATABASE IF EXISTS elidekdb;
CREATE DATABASE elidekdb;
USE elidekdb;
```

Queries για την δημιουργία των σχέσεων:

```sql
CREATE TABLE program (
  program_id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  program_name VARCHAR(90) NOT NULL,
  department VARCHAR(90) DEFAULT NULL,
  PRIMARY KEY (program_id));

CREATE TABLE employee(
  employee_id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(45) NOT NULL,
  last_name VARCHAR(45) NOT NULL ,
  PRIMARY KEY (employee_id));

 CREATE TABLE organization(
 org_id INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
 org_name VARCHAR(60) NOT NULL,
 city VARCHAR(45) NULL,
 address VARCHAR(45) NULL,
 zip_code INT(5) NULL,
 initials VARCHAR(45) NOT NULL,
 organization_type VARCHAR(15) NOT NULL,
 ministry_funds INT(12) NULL,
 private_funds INT(12) NULL,
 equity INT(12) NULL,
 constraint has_type check(organization_type in ('Research_Center','Firm','University')),
 constraint budget  check((organization_type='Research_Center' AND equity=NULL) OR  (organization_type='University' AND equity=NULL
AND private_funds=NULL) OR (organization_type='Firm' AND private_funds=NULL AND ministry_funds=NULL) OR
(equity=NULL AND private_funds=NULL AND ministry_funds=NULL)),
    PRIMARY KEY (org_id)
    );

CREATE TABLE researcher (
  researcher_id int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  first_name varchar(45) NOT NULL,
  last_name varchar(45) NOT NULL,
  sex varchar(6) NOT NULL,
  constraint sex_value check (sex in ('male', 'female', 'other')),
  date_of_birth DATE,
  date_hired DATE,
  PRIMARY KEY (researcher_id),
  org_id int(10) UNSIGNED NOT NULL,
  CONSTRAINT researcher_works_for FOREIGN KEY (org_id) REFERENCES organization (org_id) ON UPDATE CASCADE
) ;
```

```sql
CREATE TABLE project(
  project_id int(10) unsigned NOT NULL AUTO_INCREMENT,
  start_date DATE NOT NULL,
  end_date DATE NOT NULL,
  CONSTRAINT time_continuity CHECK (start_date < end_date),
  duration INT(1) NULL,             -- tha to doume
  CONSTRAINT min_max_dur CHECK (duration>=1 AND duration<=4),
  funding INT(7) UNSIGNED NOT NULL,
  CONSTRAINT min_max_fund CHECK (funding>=100000 AND funding <= 1000000),
  project_title VARCHAR(45),
  project_description VARCHAR(255),
  employee_id INT(10) UNSIGNED NOT NULL,
  program_id INT(10) UNSIGNED NOT NULL,
  org_id int(10) UNSIGNED NOT NULL,
  researcher_id_sup INT(10) UNSIGNED NOT NULL,
  researcher_id_ev INT(10) UNSIGNED NOT NULL,
  evaluation INT(3) UNSIGNED NOT NULL,
  CONSTRAINT score CHECK (evaluation >= 0 AND evaluation <= 100),
  evaluation_date DATE NOT NULL,
  CONSTRAINT eval_first CHECK (evaluation_date < start_date),
  CONSTRAINT sup_not_eval CHECK (researcher_id_sup <> researcher_id_ev),
  CONSTRAINT project_emp_id FOREIGN KEY (employee_id) REFERENCES employee (employee_id) ON UPDATE CASCADE,
  CONSTRAINT project_prog_id FOREIGN KEY (program_id) REFERENCES program (program_id) ON UPDATE CASCADE,
  CONSTRAINT project_id_org FOREIGN KEY (org_id) REFERENCES organization (org_id) ON UPDATE CASCADE,
  CONSTRAINT project_res_sup FOREIGN KEY (researcher_id_sup) REFERENCES researcher (researcher_id) ON UPDATE CASCADE,
  CONSTRAINT project_res_ev FOREIGN KEY (researcher_id_ev) REFERENCES researcher (researcher_id) ON UPDATE CASCADE,
  PRIMARY KEY(project_id)
  );
```

```sql
  CREATE TABLE research_field(
  project_id int(10) unsigned NOT NULL AUTO_INCREMENT,
  CONSTRAINT res_field_id FOREIGN KEY (project_id) REFERENCES project (project_id) ON UPDATE CASCADE on delete cascade,
  field_name VARCHAR(100),
  CONSTRAINT pk_research_field PRIMARY KEY (project_id,field_name)
);

CREATE TABLE org_phone(
org_id int(10) unsigned not null,
constraint phone_org_id foreign key (org_id) references organization (org_id) on update cascade on delete cascade,
phone_number char(10) not null,
constraint pk_phone primary key (org_id, phone_number)
);
```

```sql
  CREATE TABLE works_on(
  project_id int(10) unsigned NOT NULL,
  researcher_id int(10) UNSIGNED NOT NULL,
  CONSTRAINT works_on_project FOREIGN KEY (project_id) REFERENCES project (project_id) ON UPDATE CASCADE on delete cascade,
  CONSTRAINT res_works_on FOREIGN KEY (researcher_id) REFERENCES researcher (researcher_id) ON UPDATE CASCADE on delete cascade,
  CONSTRAINT pk_works_on PRIMARY KEY (project_id,researcher_id)
  );

  CREATE TABLE deliverable(
  project_id int(10) unsigned NOT NULL,
  CONSTRAINT del_project FOREIGN KEY (project_id) REFERENCES project (project_id) ON UPDATE CASCADE on delete cascade,
  deliverable_id int(10) unsigned NOT NULL AUTO_INCREMENT,
  deliverable_title VARCHAR(45) NOT NULL,
  deliverable_summary VARCHAR(255),
  delivery_date DATE NOT NULL,
  KEY (deliverable_id),
  CONSTRAINT pk_delivery PRIMARY KEY (project_id,deliverable_id)
  );
```

Επιβεβαιώνεται πως κατά την δημιουργία του κάθε πίνακα ορίζονται αυστηρά οι τύποι καθε χαρακτηριστικού καθώς οι περιορισμοί τους (not null, null, primary key, unsigned).

Ταυτόχρονα ορίζονται και οι απλοί έλεγχοι που πρέπει να κάνει η βάση και αναφέραμε νωρίτερα. Ένα παράδειγμα είναι το “CONSTRAINT time_continuity CHECK (start_date < end_date)” που ελέγχει τον περιορισμό 10.
Τέλος ορίζουμε τα κλειδιά που χρειαζόμαστε για να γίνονται σωστά οι παραπομπές μεταξύ σχέσεων. Προφανώς η σειρά δημιουργίας των σχέσεων είναι τέτοια ώστε να υπάρχουν ήδη οι πίνακες στους οποίους θα αναφέρεται το foreign key.

Ακολουθούν τα constraints που καλύπτουν τους υπόλοιπους κανόνες:

Ο ερευνητής που αξιολόγησε ένα έργο δεν μπορεί να δουλεύει στον οργανισμό στον οποίο ανήκει το έργο και συνεπώς ούτε να δουλεύει σε αυτο.

```
DELIMITER $
CREATE TRIGGER no_insider_evaluators_ins BEFORE INSERT ON project
FOR EACH ROW
BEGIN
    IF ((SELECT COUNT(*) FROM researcher r
                        WHERE new.org_id = r.org_id AND r.researcher_id = new.researcher_id_ev ) > 0) THEN
    SIGNAL SQLSTATE '45000'
           SET MESSAGE_TEXT = 'check constraint on project failed - A researcher cannot evaluate a project for their organization';
    END IF;
END$
DELIMITER ;

DELIMITER $
CREATE TRIGGER no_insider_evaluators_upd BEFORE UPDATE ON project
FOR EACH ROW
BEGIN
    IF ((SELECT COUNT(*) FROM researcher r
                        WHERE new.org_id = r.org_id AND r.researcher_id = new.researcher_id_ev ) > 0) THEN
    SIGNAL SQLSTATE '45000'
           SET MESSAGE_TEXT = 'check constraint on project failed - A researcher cannot evaluate a project for their organization';
    END IF;
END$
DELIMITER ;
```

Σε ένα έργο δουλεύουν μόνο ερευνητές του οργανισμού στον οποίο ανήκει το έργο.

```
DELIMITER $
CREATE TRIGGER researchers_work_on_their_orgs_projects BEFORE INSERT ON works_on
FOR EACH ROW
BEGIN
    IF ((SELECT COUNT(*) FROM project p join researcher r
                        on p.org_id=r.org_id
                        WHERE p.project_id = new.project_id AND r.researcher_id = new.researcher_id) <> 1) THEN
    SIGNAL SQLSTATE '45000'
           SET MESSAGE_TEXT = 'check constraint on_works on failed - A researcher is loyal and cannot work on foreign projects';
    END IF;
END$
DELIMITER ;

DELIMITER $
CREATE TRIGGER researchers_work_on_their_orgs_projects_upd BEFORE UPDATE ON works_on
FOR EACH ROW
BEGIN
    IF ((SELECT COUNT(*) FROM project p join researcher r
                        on p.org_id=r.org_id
                        WHERE p.project_id = new.project_id AND r.researcher_id = new.researcher_id) <> 1) THEN
    SIGNAL SQLSTATE '45000'
           SET MESSAGE_TEXT = 'check constraint on_works on failed - A researcher is loyal and cannot work on foreign projects';
    END IF;
END$
DELIMITER ;
```

Ένας ερευνητής δεν πρέπει να δουλεύει σε κάποιο έργο για να αλλάξει τον οργανισμό που ανήκει. Επίσης δεν μπορεί να μεταφερθεί σε οργανισμό για τον οποίο έχει αξιολογήσει κάποιο έργο.

```
DELIMITER $
CREATE TRIGGER researcher_transfer BEFORE update on researcher
FOR EACH ROW
BEGIN
    IF (new.org_id<>old.org_id and( (SELECT COUNT(*) FROM works_on WHERE researcher_id = new.researcher_id) > 0 or
    (select count(*) from project where org_id = new.org_id  and researcher_id_ev = new.researcher_id) > 0)) THEN
    SIGNAL SQLSTATE '45000'
          SET MESSAGE_TEXT = 'check constraint on researcher failed - A researcher has to drop his projects before he swaps org';
    END IF;
END$
DELIMITER ;
```

Κάθε έργο έχει έναν οργανισμό που το διαχειρίζεται ο οποίος δεν μπορεί να αλλάξει.

```
DELIMITER $
CREATE TRIGGER project_belongs_to_org BEFORE update ON project
FOR EACH ROW
BEGIN
    IF ((SELECT COUNT(*) FROM project WHERE old.org_id <> new.org_id) > 0) THEN
    SIGNAL SQLSTATE '45000'
          SET MESSAGE_TEXT = 'check constraint project on failed - the projects organization is unchangeable';
    END IF;
END$
DELIMITER ;
```

Ο παρακάτω trigger χρησιμοποιείται για να υπολογίσει το derived attribute “duration” σε κάθε project.

```
DELIMITER $
CREATE TRIGGER calc_duration_ins before INSERT ON project
FOR EACH ROW
BEGIN
set new.duration= DATEDIFF(new.end_date, new.start_date) / 365.25 ;
END$
DELIMITER ;

DELIMITER $
CREATE TRIGGER calc_duration_upd before update ON project
FOR EACH ROW
BEGIN
set new.duration= DATEDIFF(new.end_date, new.start_date) / 365.25 ;
END$
DELIMITER ;
```

Όλα τα παραδοτέα πρέπει να υποβληθούν πριν την λήξη του έργου.

```
DELIMITER $
CREATE TRIGGER deliver_on_time before insert ON deliverable
FOR EACH ROW
BEGIN
IF ((SELECT COUNT(*) FROM project WHERE project_id = new.project_id AND
(new.delivery_date < start_date OR new.delivery_date > end_date )) > 0) THEN
    SIGNAL SQLSTATE '45000'
           SET MESSAGE_TEXT = 'check constraint on deliverable failed - This is not Back to the Future';
    END IF;
END$
DELIMITER ;

DELIMITER $
CREATE TRIGGER deliver_on_time_upd before update ON deliverable
FOR EACH ROW
BEGIN
IF ((SELECT COUNT(*) FROM project WHERE project_id = new.project_id AND
(new.delivery_date < start_date OR new.delivery_date > end_date )) > 0) THEN
    SIGNAL SQLSTATE '45000'
           SET MESSAGE_TEXT = 'check constraint on deliverable failed - This is not Back to the Future';
    END IF;
END$
DELIMITER ;
```

Ο supervisor καθε έργου δουλεύει σε αυτό.

```
DELIMITER $
CREATE TRIGGER add_supervisor_works_on_INSERT AFTER INSERT ON project
FOR EACH ROW
BEGIN
    INSERT INTO works_on (project_id, researcher_id) VALUES (new.project_id, new.researcher_id_sup);
END$
DELIMITER ;

DELIMITER $
CREATE TRIGGER add_supervisor_works_on_UPDATE AFTER UPDATE ON project
FOR EACH ROW
BEGIN
    REPLACE INTO works_on
    SET project_id = new.project_id, researcher_id = new.researcher_id_sup;
END$
DELIMITER ;
```

Δημιουργούμε επίσης τα δύο views που ζητούνται στο ερώτημα 3.2:

```
DROP VIEW IF EXISTS researchersprojects;
CREATE VIEW researchersprojects AS
select r.researcher_id ,r.first_name, r.last_name, p.project_id, p.project_title
FROM researcher r
INNER JOIN works_on w
ON r.researcher_id = w.researcher_id
INNER JOIN project p
ON p.project_id = w.project_id
ORDER BY r.researcher_id;

DROP VIEW IF EXISTS organizationsresearchers;
CREATE VIEW organizationsresearchers AS
SELECT r.researcher_id, r.first_name, r.last_name, r.org_id, org.org_name
FROM researcher r
INNER JOIN
organization org
ON r.org_id = org.org_id
ORDER BY r.org_id ASC;
```

Το δεύτερο view εμφανίζει τους researchers (id, ονομα, επίθετο) ανά οργανισμό.

Χρησιμοποιούμε ευρετήρια σε στήλες που χρησιμοποιούνται σε queries συχνά. Μέσω των ευρετηρίων ελαχιστοποιείται ο χρόνος ολοκλήρωσης του query. Η mysql δημιουργεί αυτόματα ευρετήρια στο primary key κάθε σχέσης τα οποία και χρησιμοποιούμε πολύ συχνά. Σε αυτά τα ευρετήρια επιλέγουμε να προσθέσουμε ένα στην στήλη field_name του research_field και ένα στις στήλες start_date και end_date του project που χρησιμοποιούνται για τα αιτήματα του χρήστη.

```
create index idx_field_name on research_field(field_name);
create index idx_projects_start_date on project(start_date);
create index idx_projects_end_date on project(end_date);
create index idx_researcher_org_id on researcher(org_id);
create index idx_project_org_id on project(org_id);
create index idx_project_researcher_id_ev on project(researcher_id_ev);
```

Τέλος εισάγουμε αναληθή δεδομένα (dummy data) στην βάση για να μπορούμε να ελέγξουμε τα queries που ζητούνται στο 3ο ερώτημα αλλά και την υλοποίηση του CRUD.

Για την ανάπτυξη και διαχείριση της βάσης χρησιμοποιήσαμε το DBMS client “mysql workbench”.
Για την δημιουργία του UI που θα επιτυγχάνει την υλοποίηση του CRUD και των ερωτημάτων που ζητούνται χρησιμοποιήσαμε php και html. Ο χρήστης μπορεί να επιλέξει και να ακολουθήσει τις οδηγίες που του δίνονται για να πετύχει το εκάστοτε αίτημα.
Η σύνδεση μεταξύ των δύο επιτυγχάνεται μέσω του Apache Server και την διαχειρίζεται η εφαρμογή xampp control panel.

Για την εγκατάσταση και εκτέλεση της εφαρμογής μας απαιτείται εγκατάσταση της εφαρμογής xampp control panel. Μέσω αυτού ενεργοποιούνται οι υπηρεσίες mysql (port 3306) και apache server (port 80, 443). Επίσης χρειάζεται η λήψη του φακέλου elidekwebapp και τοποθέτησή του στο φάκελο htdocs του xampp.

Τέλος σε κάποιον browser πηγαίνουμε στη διεύθυνση http://localhost/elidekwebapp.