Project Βάσεων Δεδομένων

2022-2023

A picture containing text

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Βάσεων Δεδομένων | | | | | |
| **Μέλη project: Κικίδης Χαρίτων**  **ΑΜ:1084595**  **Έτος φοίτησης:3o**  **e-mail:up1084595@upnet.gr** | | | | | |
| Το project το υλοποίησα ατομικά καθώς δεν βρήκα κάποιον συνεργάτη που να γνώριζα ότι θα έκανε την δουλειά που απαιτούταν. |  | Όλος ο κώδικας της SQL έγινε στο MySQL workbench 8.0.31 και το GUI σε javafx στο  IntelliJ IDE. |  | Όσον αφορά την βαθμολόγηση της εργασίας υποθέτω ότι θα βαθμολογηθώ με βάση το Project των 2 ατόμων. |  |
|  | |  | |  | |

**Κεφάλαιο 1: ΣΧΕΣΙΑΚΟ ΔΙΑΓΡΑΜΜΑ ER**

Παρακάτω βρίσκεται το ER διάγραμμα της βάσης δεδομένων που δημιούργησα.

Diagram

Description automatically generated

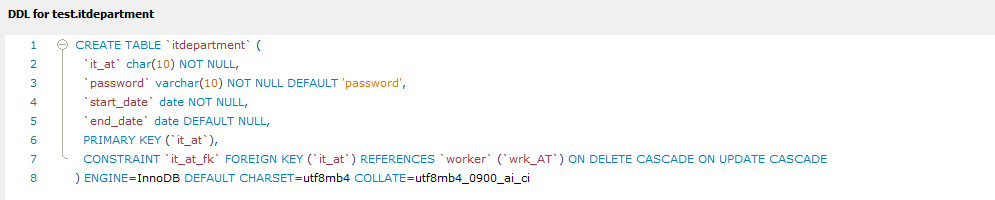
* Τα θεμέλια της βασης είχαν γίνει κατά την διάρκεια της 5ης εργαστηριακής άσκησης όπου δημιουργήθηκαν οι βασικοί πίνακες.
* **Προστέθηκαν σε αυτή τη βασική δομή οι εξής πίνακες:**

1. **itdepartment**
2. **logger**
3. **offers**
4. **reservation\_offers**

Οι πίνακες αυτοί ήταν απαραίτητοι για να εκπληρωθούν τα ζητούμενα στα ερωτήματα 3.1.3 και 3.1.4 και έχουν φτιαχτεί με βάση τα πεδία που αναφέρονται στην εκφώνηση της άσκησης.Δεν τροποποίησα τους προϋπάρχοντες πίνακες με νέα πεδία.

Συγκεκριμένα στους πίνακες:

**Itdepartment**->

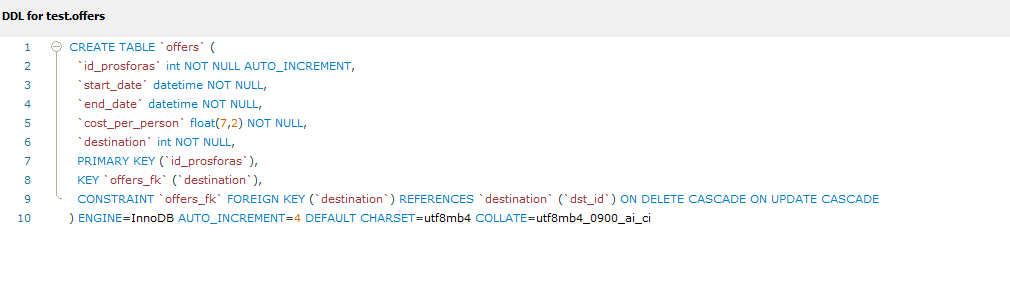


Έιναι ο admin της βάσης δεδομένων μας και αυτός είναι που χειρίζεται ουσιαστικά όλες τις αλλαγές και τις εισαγωγές νέας πληροφορίας.  
Είναι τύπου worker και έχει ένα μοναδικό it\_at που εξυπηρετεί σαν primary key του.Έιναι ο μοναδικός πίνακας που περιέχει το πεδίο password και απαιτείται ο κωδικός για την είσοδό του στην ΒΔ από το GUI. Τέλος,καταγράφεται η ημερομηνία που άρχισε την εργασία του και η ημερομηνία τέλος στην περίπτωση που τελειώσει η σύμβασή του.Τον χρειαζομαστε για το GUI κυρίως.

Όλες οι ενέργειές του αποθηκεύονται στον πίνακα logger,μέσω των trigger που έφτιαξα στο 3.1.2.1(Έφτιαξα trigger για κάθε πίνακα που υπάρχει στην ΒΔ,όχι μονο αυτούς που αναφέρονται).

Οι πίνακες offers και reservation\_offers προστέθηκαν για να γίνει δυνατό το ερώτημα 3.1.2.3.

Για το offers:



Δίνεται με την offers η δυνατότητα στον χρήστη να κάνει ένα offer για ένα trip δίνοντας μία προσφορά και ημερομηνίες ταξιδιού. Αν μαζευτούν αρκετές προσφορές για το ταξιδί το πρακτορείο θα το προσθέσει στον προορισμό του.

Το reservation\_offer περιέχει τα προσωπικά στοιχεία που δίνει ο χρήστης για το ταξίδι και την προκαταβολή που είναι πρόθυμος να δώσει.



Έπειτα από την δημιουργία των 2 αυτών πινάκων κάνουμε εισαγωγή όλων των απαραίτητων insert για να μπορεί η βάση να λειτουργήσει.

Στην συνέχεια θα κάνω εισαγωγή 60κ ονομάτων. Επειδή προσπάθησα να τα κανω import με ενα JSON αρχείο αλλά δεν γινόταν σωστά,για κάποιο λόγο,αποφάσισα να δημιουργήσω στο excel 20k ονόματα με την χρήση των ειδικών συναρτήσεων του Μicrosoft Excel και βάζω τα 20κ ονόματα για 3 offer trips που έχουνε δημιουργηθεί. Τα 60κ ονόματα θα τα έχω στο zip αρχείο αλλα δεν θα τα βάλω στο word για να αποφύγω την δυσαναγνωση του κειμένου.

Στο 3.1.3.2 έφτιαξα επίσης ολα τα trigger για το logger για κάθε πίνακα της βάσης στην περίπτωση που ο it κάνει INSERT,DELETE,UPDATE κάποιο πίνακα.

Παρακάτω είναι ο κώδικας για τις CREATE και τις βασικές INSERT καθώς και τα trigger για το logger.

**ΚΩΔΙΚΑΣ:**

**-- BRANCH--**

**DROP TABLE IF EXISTS branch;**

**CREATE TABLE IF NOT EXISTS branch**

**(**

**br\_code INT(11) UNIQUE AUTO\_INCREMENT NOT NULL ,**

**br\_street VARCHAR(30) NOT NULL,**

**br\_num INT(4) NOT NULL,**

**br\_city varchar(30),**

**PRIMARY KEY(br\_code)**

**);**

**-- WORKER--**

**DROP TABLE IF EXISTS worker;**

**CREATE TABLE IF NOT EXISTS worker (**

**wrk\_AT char(10) NOT NULL,**

**wrk\_name varchar(20) NOT NULL,**

**wrk\_lname varchar(20) NOT NULL,**

**wrk\_salary float(7,2) NOT NULL,**

**wrk\_br\_code int NOT NULL,**

**PRIMARY KEY (wrk\_AT),**

**CONSTRAINT worker\_fk1 FOREIGN KEY (wrk\_br\_code) REFERENCES branch(br\_code) ON DELETE CASCADE ON UPDATE CASCADE**

**);**

**-- GUIDE--**

**DROP TABLE IF EXISTS guide;**

**CREATE TABLE IF NOT EXISTS guide**

**(**

**gui\_AT CHAR(10) UNIQUE NOT NULL,**

**gui\_cv TEXT ,**

**PRIMARY KEY (gui\_AT),**

**CONSTRAINT FOREIGN KEY (gui\_AT) REFERENCES worker(wrk\_AT)**

**);**

**-- ADMIN--**

**DROP TABLE IF EXISTS admin;**

**CREATE TABLE IF NOT EXISTS admin**

**(**

**adm\_AT CHAR(10) NOT NULL UNIQUE,**

**adm\_type ENUM('LOGISTICS','ADMINISTRATIVE','ACCOUNTING') NOT NULL,**

**adm\_diploma VARCHAR(200) NOT NULL,**

**PRIMARY KEY (adm\_AT),**

**CONSTRAINT adm\_fk1 FOREIGN KEY (adm\_AT) REFERENCES worker(wrk\_AT)**

**ON UPDATE CASCADE ON DELETE CASCADE**

**);**

**-- DRIVER--**

**DROP TABLE IF EXISTS driver;**

**CREATE TABLE IF NOT EXISTS driver**

**(**

**drv\_AT CHAR(10) UNIQUE NOT NULL,**

**drv\_license ENUM('A','B','C','D') NOT NULL,**

**drv\_route ENUM('LOCAL','ABROAD') NOT NULL,**

**drv\_experience TINYINT(4),**

**PRIMARY KEY(drv\_AT),**

**CONSTRAINT drv\_fk FOREIGN KEY (drv\_AT) REFERENCES worker(wrk\_AT)**

**ON UPDATE CASCADE ON DELETE CASCADE**

**);**

**-- TRIP--**

**DROP TABLE IF EXISTS trip;**

**CREATE TABLE IF NOT EXISTS trip**

**(**

**tr\_id INT(11) NOT NULL AUTO\_INCREMENT,**

**tr\_departure DATETIME NOT NULL,**

**tr\_return DATETIME NOT NULL,**

**tr\_maxseats tinyint(4) NOT NULL,**

**tr\_cost FLOAT(7,2) NOT NULL,**

**tr\_br\_code INT(11) NOT NULL,**

**tr\_gui\_AT CHAR (10) NOT NULL,**

**tr\_drv\_AT CHAR (10) NOT NULL,**

**PRIMARY KEY(tr\_id),**

**CONSTRAINT tr\_guifk FOREIGN KEY (tr\_gui\_AT) REFERENCES guide(gui\_AT) ,**

**CONSTRAINT tr\_br\_code\_fk FOREIGN KEY (tr\_br\_code) REFERENCES branch(br\_code),**

**CONSTRAINT tr\_drvfk FOREIGN KEY (tr\_drv\_AT) REFERENCES driver(drv\_AT)**

**ON UPDATE CASCADE ON DELETE CASCADE);**

**-- EVENT--**

**DROP TABLE IF EXISTS Event;**

**CREATE TABLE IF NOT EXISTS Event(**

**ev\_tr\_id INT(11) NOT NULL,**

**ev\_start DATETIME NOT NULL,**

**ev\_end DATETIME NOT NULL,**

**ev\_descr TEXT,**

**PRIMARY KEY(ev\_tr\_id,ev\_start),**

**CONSTRAINT ev\_tr\_id\_fk FOREIGN KEY (ev\_tr\_id) REFERENCES trip(tr\_id)**

**ON UPDATE CASCADE ON DELETE CASCADE**

**);**

**-- PHONES--**

**DROP TABLE IF EXISTS phones;**

**CREATE TABLE IF NOT EXISTS phones**

**(**

**ph\_br\_code INT(11) UNIQUE NOT NULL ,**

**ph\_number CHAR(10) NOT NULL,**

**PRIMARY KEY(ph\_br\_code,ph\_number),**

**CONSTRAINT ph\_br\_code\_fk FOREIGN KEY (ph\_br\_code) REFERENCES branch(br\_code)**

**ON UPDATE CASCADE ON DELETE CASCADE**

**);**

**-- DESTINATION--**

**DROP TABLE IF EXISTS destination;**

**CREATE TABLE IF NOT EXISTS destination**

**(**

**dst\_id INT(11) UNIQUE AUTO\_INCREMENT NOT NULL ,**

**dst\_name VARCHAR(50) NOT NULL,**

**dst\_descr text,**

**dst\_rtype enum('LOCAL','ABROAD') DEFAULT NULL,**

**dst\_location INT(11) NOT NULL,**

**dst\_language VARCHAR(30),**

**PRIMARY KEY (dst\_id),**

**CONSTRAINT dst\_location\_fk FOREIGN KEY (dst\_location) REFERENCES destination(dst\_id) -- self reference edw--**

**ON UPDATE CASCADE ON DELETE CASCADE**

**);**

**-- TRAVEL\_TO--**

**DROP TABLE IF EXISTS travel\_to;**

**CREATE TABLE IF NOT EXISTS travel\_to**

**(**

**to\_tr\_id INT(11) NOT NULL,**

**to\_dst\_id INT (11) NOT NULL,**

**to\_arrival DATETIME NOT NULL,**

**to\_departure DATETIME NOT NULL,**

**PRIMARY KEY (to\_tr\_id,to\_dst\_id),**

**CONSTRAINT to\_tr\_id\_FK FOREIGN KEY (to\_tr\_id) REFERENCES trip(tr\_id),**

**CONSTRAINT to\_dst\_id\_fk FOREIGN KEY (to\_dst\_id) REFERENCES destination(dst\_id)**

**ON UPDATE CASCADE ON DELETE CASCADE**

**);**

**-- RESERVATION--**

**DROP TABLE IF EXISTS reservation;**

**CREATE TABLE IF NOT EXISTS reservation**

**(**

**res\_tr\_id INT(11) NOT NULL,**

**res\_seatnum tinyint(4) NOT NULL,**

**res\_name VARCHAR(20) NOT NULL,**

**res\_lname VARCHAR(20) NOT NULL,**

**res\_isadult ENUM('ADULT','MINOR') NOT NULL,**

**PRIMARY KEY(res\_tr\_id,res\_seatnum),**

**CONSTRAINT res\_tr\_id\_FK FOREIGN KEY (res\_tr\_id) REFERENCES trip(tr\_id)**

**ON UPDATE CASCADE ON DELETE CASCADE**

**);**

**-- MANAGER--**

**DROP TABLE IF EXISTS manages;**

**CREATE TABLE IF NOT EXISTS manages**

**(**

**mng\_adm\_AT CHAR(10) UNIQUE NOT NULL,**

**mng\_br\_code INT(11) UNIQUE NOT NULL,**

**PRIMARY KEY(mng\_adm\_AT,mng\_br\_code),**

**CONSTRAINT mng\_adm\_AT\_fk FOREIGN KEY (mng\_adm\_AT) REFERENCES admin(adm\_AT),**

**CONSTRAINT mng\_br\_code\_fk FOREIGN KEY (mng\_br\_code) REFERENCES branch(br\_code)**

**ON UPDATE CASCADE ON DELETE CASCADE**

**);**

**-- LANGUAGES --**

**DROP TABLE IF EXISTS languages;**

**CREATE TABLE IF NOT EXISTS languages**

**(**

**lng\_gui\_AT CHAR(10) UNIQUE NOT NULL,**

**lng\_language varchar(30) NOT NULL DEFAULT 'UNKNOWN',**

**PRIMARY KEY (lng\_gui\_AT),**

**CONSTRAINT lng\_gui\_AT\_fk FOREIGN KEY (lng\_gui\_AT) REFERENCES guide(gui\_AT)**

**ON UPDATE CASCADE ON DELETE CASCADE**

**);**

**DROP TABLE IF EXISTS itdepartment;**

**CREATE TABLE IF NOT EXISTS itdepartment**

**(**

**it\_at char(10) NOT NULL,**

**password varchar(10) DEFAULT 'password' NOT NULL,**

**start\_date DATE NOT NULL,**

**end\_date DATE DEFAULT NULL,**

**PRIMARY KEY(it\_at),**

**CONSTRAINT it\_at\_fk FOREIGN KEY (it\_at) REFERENCES worker(wrk\_AT)**

**ON UPDATE CASCADE ON DELETE CASCADE**

**);**

**-- table gia to log**

**DROP TABLE IF EXISTS logger;**

**CREATE TABLE IF NOT EXISTS logger**

**(**

**change\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**user VARCHAR(100),**

**actiontype VARCHAR(20),**

**changes VARCHAR(300),**

**dateandtime TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**

**);**

**-- TA TRIGGER GIA INSERT,DELETE KAI UPDATE**

**-- phones**

**DELIMITER $$**

**DROP TRIGGER IF EXISTS phonesinsert$$**

**CREATE TRIGGER phonesinsert**

**AFTER INSERT on phones**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.ph\_br\_code,' ',NEW.ph\_number);**

**INSERT INTO logger VALUES(NULL,current\_user(),'INSERT ON PHONES',changes,current\_timestamp());**

**END$$**

**-- guide**

**DROP TRIGGER IF EXISTS guideinsert$$**

**CREATE TRIGGER guideinsert**

**AFTER INSERT on guide**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.gui\_AT,' ',NEW.gui\_cv);**

**INSERT INTO logger VALUES(NULL,current\_user(),'INSERT ON GUIDE',changes,current\_timestamp());**

**END$$**

**-- manages**

**DROP TRIGGER IF EXISTS managesinsert$$**

**CREATE TRIGGER managesinsert**

**AFTER INSERT on manages**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.mng\_adm\_AT,' ',NEW.mng\_br\_code);**

**INSERT INTO logger VALUES(NULL,current\_user(),'INSERT ON MANAGES',changes,current\_timestamp());**

**END$$**

**-- languages**

**DROP TRIGGER IF EXISTS languagesinsert$$**

**CREATE TRIGGER languagesinsert**

**AFTER INSERT on languages**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.lng\_gui\_AT,' ',NEW.lng\_language);**

**INSERT INTO logger VALUES(NULL,current\_user(),'INSERT ON LANGUAGES',changes,current\_timestamp());**

**END$$**

**-- branch**

**DROP TRIGGER IF EXISTS branchinsert$$**

**CREATE TRIGGER branchinsert**

**AFTER INSERT on branch**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.br\_code,' ',NEW.br\_street,' ',NEW.br\_num,' ',NEW.br\_city);**

**INSERT INTO logger VALUES(NULL,current\_user(),'INSERT ON BRANCH',changes,current\_timestamp());**

**END$$**

**-- driver**

**DROP TRIGGER IF EXISTS driverinsert$$**

**CREATE TRIGGER driverinsert**

**AFTER INSERT on driver**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.drv\_AT,' ',NEW.drv\_license,' ',NEW.drv\_route,' ',NEW.drv\_experience);**

**INSERT INTO logger VALUES(NULL,current\_user(),'INSERT ON DRIVER',changes,current\_timestamp());**

**END$$**

**-- event**

**DROP TRIGGER IF EXISTS eventinsert$$**

**CREATE TRIGGER eventinsert**

**AFTER INSERT on event**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.ev\_tr\_id,' ',NEW.ev\_start,' ',NEW.ev\_end,' ',NEW.ev\_descr);**

**INSERT INTO logger VALUES(NULL,current\_user(),'INSERT ON EVENT',changes,current\_timestamp());**

**END$$**

**-- travelto**

**DROP TRIGGER IF EXISTS traveltoinsert$$**

**CREATE TRIGGER traveltoinsert**

**AFTER INSERT on travel\_to**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.to\_tr\_id,' ',NEW.to\_dst\_id,' ',NEW.to\_arrival,' ',NEW.to\_departure);**

**INSERT INTO logger VALUES(NULL,current\_user(),'INSERT ON TRAVEL TO',changes,current\_timestamp());**

**END$$**

**-- worker**

**DROP TRIGGER IF EXISTS workerinsert$$**

**CREATE TRIGGER workerinsert**

**AFTER INSERT on worker**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.wrk\_AT,' ',NEW.wrk\_name,' ',NEW.wrk\_lname,' ',NEW.wrk\_salary,' ',NEW.wrk\_br\_code);**

**INSERT INTO logger VALUES(NULL,current\_user(),'INSERT ON WORKER',changes,current\_timestamp());**

**END$$**

**-- RESERVATION**

**DROP TRIGGER IF EXISTS reservationinsert$$**

**CREATE TRIGGER reservationinsert**

**AFTER INSERT on reservation**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.res\_tr\_id,' ',NEW.res\_seatnum,' ',NEW.res\_name,' ',NEW.res\_lname,' ',NEW.res\_isadult);**

**INSERT INTO logger VALUES(NULL,current\_user(),'INSERT ON RESERVATION',changes,current\_timestamp());**

**END$$**

**-- admin**

**DROP TRIGGER IF EXISTS admininsert$$**

**CREATE TRIGGER admininsert**

**AFTER INSERT on admin**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.adm\_AT,' ',NEW.adm\_type,' ',NEW.adm\_diploma);**

**INSERT INTO logger VALUES(NULL,current\_user(),'INSERT ON ADMIN',changes,current\_timestamp());**

**END$$**

**-- DESTINATION**

**DROP TRIGGER IF EXISTS destinationinsert$$**

**CREATE TRIGGER destinationinsert**

**AFTER INSERT on destination**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.dst\_id,' ',NEW.dst\_name,' ',NEW.dst\_descr,' ',NEW.dst\_rtype,' ',NEW.dst\_language,' ',NEW.dst\_location);**

**INSERT INTO logger VALUES(NULL,current\_user(),'INSERT ON DESTINATION',changes,current\_timestamp());**

**END$$**

**-- trip**

**DROP TRIGGER IF EXISTS tripinsert$$**

**CREATE TRIGGER tripinsert**

**AFTER INSERT on trip**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.tr\_id,' ',NEW.tr\_departure,' ',NEW.tr\_return,' ',NEW.tr\_maxseats,' ',NEW.tr\_cost,' ',NEW.tr\_br\_code,' ',NEW.tr\_gui\_AT,' ',NEW.tr\_drv\_AT);**

**INSERT INTO logger VALUES(NULL,current\_user(),'INSERT ON TRIP',changes,current\_timestamp());**

**END$$**

**-- UPDATE TRIGGER**

**-- phones**

**DELIMITER $$**

**DROP TRIGGER IF EXISTS phonesUPDATE$$**

**CREATE TRIGGER phonesUPDATE**

**AFTER UPDATE on phones**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.ph\_br\_code,' ',NEW.ph\_number);**

**INSERT INTO logger VALUES(NULL,current\_user(),'UPDATE ON PHONES',changes,current\_timestamp());**

**END$$**

**-- guide**

**DROP TRIGGER IF EXISTS guideUPDATE$$**

**CREATE TRIGGER guideUPDATE**

**AFTER UPDATE on guide**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.gui\_AT,' ',NEW.gui\_cv);**

**INSERT INTO logger VALUES(NULL,current\_user(),'UPDATE ON GUIDE',changes,current\_timestamp());**

**END$$**

**-- manages**

**DROP TRIGGER IF EXISTS managesUPDATE$$**

**CREATE TRIGGER managesUPDATE**

**AFTER UPDATE on manages**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.mng\_adm\_AT,' ',NEW.mng\_br\_code);**

**INSERT INTO logger VALUES(NULL,current\_user(),'UPDATE ON MANAGES',changes,current\_timestamp());**

**END$$**

**-- languages**

**DROP TRIGGER IF EXISTS languagesUPDATE$$**

**CREATE TRIGGER languagesUPDATE**

**AFTER UPDATE on languages**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.lng\_gui\_AT,' ',NEW.lng\_language);**

**INSERT INTO logger VALUES(NULL,current\_user(),'UPDATE ON LANGUAGES',changes,current\_timestamp());**

**END$$**

**-- branch**

**DROP TRIGGER IF EXISTS branchUPDATE$$**

**CREATE TRIGGER branchUPDATE**

**AFTER UPDATE on branch**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.br\_code,' ',NEW.br\_street,' ',NEW.br\_num,' ',NEW.br\_city);**

**INSERT INTO logger VALUES(NULL,current\_user(),'UPDATE ON BRANCH',changes,current\_timestamp());**

**END$$**

**-- driver**

**DROP TRIGGER IF EXISTS driverUPDATE$$**

**CREATE TRIGGER driverUPDATE**

**AFTER UPDATE on driver**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.drv\_AT,' ',NEW.drv\_license,' ',NEW.drv\_route,' ',NEW.drv\_experience);**

**INSERT INTO logger VALUES(NULL,current\_user(),'UPDATE ON DRIVER',changes,current\_timestamp());**

**END$$**

**-- event**

**DROP TRIGGER IF EXISTS eventUPDATE$$**

**CREATE TRIGGER eventUPDATE**

**AFTER UPDATE on event**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.ev\_tr\_id,' ',NEW.ev\_start,' ',NEW.ev\_end,' ',NEW.ev\_descr);**

**INSERT INTO logger VALUES(NULL,current\_user(),'UPDATE ON EVENT',changes,current\_timestamp());**

**END$$**

**-- travelto**

**DROP TRIGGER IF EXISTS traveltoUPDATE$$**

**CREATE TRIGGER traveltoUPDATE**

**AFTER UPDATE on travel\_to**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.to\_tr\_id,' ',NEW.to\_dst\_id,' ',NEW.to\_arrival,' ',NEW.to\_departure);**

**INSERT INTO logger VALUES(NULL,current\_user(),'UPDATE ON TRAVEL TO',changes,current\_timestamp());**

**END$$**

**-- worker**

**DROP TRIGGER IF EXISTS workerUPDATE$$**

**CREATE TRIGGER workerUPDATE**

**AFTER UPDATE on worker**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.wrk\_AT,' ',NEW.wrk\_name,' ',NEW.wrk\_lname,' ',NEW.wrk\_salary,' ',NEW.wrk\_br\_code);**

**INSERT INTO logger VALUES(NULL,current\_user(),'UPDATE ON WORKER',changes,current\_timestamp());**

**END$$**

**-- RESERVATION**

**DROP TRIGGER IF EXISTS reservationUPDATE$$**

**CREATE TRIGGER reservationUPDATE**

**AFTER UPDATE on reservation**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.res\_tr\_id,' ',NEW.res\_seatnum,' ',NEW.res\_name,' ',NEW.res\_lname,' ',NEW.res\_isadult);**

**INSERT INTO logger VALUES(NULL,current\_user(),'UPDATE ON RESERVATION',changes,current\_timestamp());**

**END$$**

**-- admin**

**DROP TRIGGER IF EXISTS adminUPDATE$$**

**CREATE TRIGGER adminUPDATE**

**AFTER UPDATE on admin**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.adm\_AT,' ',NEW.adm\_type,' ',NEW.adm\_diploma);**

**INSERT INTO logger VALUES(NULL,current\_user(),'UPDATE ON ADMIN',changes,current\_timestamp());**

**END$$**

**-- DESTINATION**

**DROP TRIGGER IF EXISTS destinationUPDATE$$**

**CREATE TRIGGER destinationUPDATE**

**AFTER UPDATE on destination**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.dst\_id,' ',NEW.dst\_name,' ',NEW.dst\_descr,' ',NEW.dst\_rtype,' ',NEW.dst\_language,' ',NEW.dst\_location);**

**INSERT INTO logger VALUES(NULL,current\_user(),'UPDATE ON DESTINATION',changes,current\_timestamp());**

**END$$**

**-- trip**

**DROP TRIGGER IF EXISTS tripUPDATE$$**

**CREATE TRIGGER tripUPDATE**

**AFTER UPDATE on trip**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(NEW.tr\_id,' ',NEW.tr\_departure,' ',NEW.tr\_return,' ',NEW.tr\_maxseats,' ',NEW.tr\_cost,' ',NEW.tr\_br\_code,' ',NEW.tr\_gui\_AT,' ',NEW.tr\_drv\_AT);**

**INSERT INTO logger VALUES(NULL,current\_user(),'UPDATE ON TRIP',changes,current\_timestamp());**

**END$$**

**-- DELETE-----------**

**-- TA TRIGGER GIA DELETE,DELETE KAI DELETE**

**-- phones**

**DELIMITER $$**

**DROP TRIGGER IF EXISTS phonesDELETE$$**

**CREATE TRIGGER phonesDELETE**

**AFTER DELETE on phones**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(OLD.ph\_br\_code,' ',OLD.ph\_number);**

**INSERT INTO logger VALUES(NULL,current\_user(),'DELETE ON PHONES',changes,current\_timestamp());**

**END$$**

**-- guide**

**DROP TRIGGER IF EXISTS guideDELETE$$**

**CREATE TRIGGER guideDELETE**

**AFTER DELETE on guide**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(OLD.gui\_AT,' ',OLD.gui\_cv);**

**INSERT INTO logger VALUES(NULL,current\_user(),'DELETE ON GUIDE',changes,current\_timestamp());**

**END$$**

**-- manages**

**DROP TRIGGER IF EXISTS managesDELETE$$**

**CREATE TRIGGER managesDELETE**

**AFTER DELETE on manages**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(OLD.mng\_adm\_AT,' ',OLD.mng\_br\_code);**

**INSERT INTO logger VALUES(NULL,current\_user(),'DELETE ON MANAGES',changes,current\_timestamp());**

**END$$**

**-- languages**

**DROP TRIGGER IF EXISTS languagesDELETE$$**

**CREATE TRIGGER languagesDELETE**

**AFTER DELETE on languages**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(OLD.lng\_gui\_AT,' ',OLD.lng\_language);**

**INSERT INTO logger VALUES(NULL,current\_user(),'DELETE ON LANGUAGES',changes,current\_timestamp());**

**END$$**

**-- branch**

**DROP TRIGGER IF EXISTS branchDELETE$$**

**CREATE TRIGGER branchDELETE**

**AFTER DELETE on branch**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(OLD.br\_code,' ',OLD.br\_street,' ',OLD.br\_num,' ',OLD.br\_city);**

**INSERT INTO logger VALUES(NULL,current\_user(),'DELETE ON BRANCH',changes,current\_timestamp());**

**END$$**

**-- driver**

**DROP TRIGGER IF EXISTS driverDELETE$$**

**CREATE TRIGGER driverDELETE**

**AFTER DELETE on driver**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(OLD.drv\_AT,' ',OLD.drv\_license,' ',OLD.drv\_route,' ',OLD.drv\_experience);**

**INSERT INTO logger VALUES(NULL,current\_user(),'DELETE ON DRIVER',changes,current\_timestamp());**

**END$$**

**-- event**

**DROP TRIGGER IF EXISTS eventDELETE$$**

**CREATE TRIGGER eventDELETE**

**AFTER DELETE on event**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(OLD.ev\_tr\_id,' ',OLD.ev\_start,' ',OLD.ev\_end,' ',OLD.ev\_descr);**

**INSERT INTO logger VALUES(NULL,current\_user(),'DELETE ON EVENT',changes,current\_timestamp());**

**END$$**

**-- travelto**

**DROP TRIGGER IF EXISTS traveltoDELETE$$**

**CREATE TRIGGER traveltoDELETE**

**AFTER DELETE on travel\_to**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(OLD.to\_tr\_id,' ',OLD.to\_dst\_id,' ',OLD.to\_arrival,' ',OLD.to\_departure);**

**INSERT INTO logger VALUES(NULL,current\_user(),'DELETE ON TRAVEL TO',changes,current\_timestamp());**

**END$$**

**-- worker**

**DROP TRIGGER IF EXISTS workerDELETE$$**

**CREATE TRIGGER workerDELETE**

**AFTER DELETE on worker**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(OLD.wrk\_AT,' ',OLD.wrk\_name,' ',OLD.wrk\_lname,' ',OLD.wrk\_salary,' ',OLD.wrk\_br\_code);**

**INSERT INTO logger VALUES(NULL,current\_user(),'DELETE ON WORKER',changes,current\_timestamp());**

**END$$**

**-- RESERVATION**

**DROP TRIGGER IF EXISTS reservationDELETE$$**

**CREATE TRIGGER reservationDELETE**

**AFTER DELETE on reservation**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(OLD.res\_tr\_id,' ',OLD.res\_seatnum,' ',OLD.res\_name,' ',OLD.res\_lname,' ',OLD.res\_isadult);**

**INSERT INTO logger VALUES(NULL,current\_user(),'DELETE ON RESERVATION',changes,current\_timestamp());**

**END$$**

**-- admin**

**DROP TRIGGER IF EXISTS adminDELETE$$**

**CREATE TRIGGER adminDELETE**

**AFTER DELETE on admin**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(OLD.adm\_AT,' ',OLD.adm\_type,' ',OLD.adm\_diploma);**

**INSERT INTO logger VALUES(NULL,current\_user(),'DELETE ON ADMIN',changes,current\_timestamp());**

**END$$**

**-- DESTINATION**

**DROP TRIGGER IF EXISTS destinationDELETE$$**

**CREATE TRIGGER destinationDELETE**

**AFTER DELETE on destination**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(OLD.dst\_id,' ',OLD.dst\_name,' ',OLD.dst\_descr,' ',OLD.dst\_rtype,' ',OLD.dst\_language,' ',OLD.dst\_location);**

**INSERT INTO logger VALUES(NULL,current\_user(),'DELETE ON DESTINATION',changes,current\_timestamp());**

**END$$**

**-- trip**

**DROP TRIGGER IF EXISTS tripDELETE$$**

**CREATE TRIGGER tripDELETE**

**AFTER DELETE on trip**

**FOR EACH ROW**

**BEGIN**

**DECLARE changes VARCHAR(300);**

**SET changes=CONCAT(OLD.tr\_id,' ',OLD.tr\_departure,' ',OLD.tr\_return,' ',OLD.tr\_maxseats,' ',OLD.tr\_cost,' ',OLD.tr\_br\_code,' ',OLD.tr\_gui\_AT,' ',OLD.tr\_drv\_AT);**

**INSERT INTO logger VALUES(NULL,current\_user(),'DELETE ON TRIP',changes,current\_timestamp());**

**END$$**

**DELIMITER ;**

**-- BRANCH INSERT--**

**INSERT INTO branch VALUES (null,'Karaiskakh 1','1','Patra');**

**INSERT INTO branch VALUES (null,'Karaiskakh 2','2','Athina');**

**INSERT INTO branch VALUES (null,'Patrews 3','5','Thessalonikh');**

**-- WORKER INSERT--**

**INSERT INTO worker VALUES('AM1000','Thomas','Johnson','1500.00','1');**

**INSERT INTO worker VALUES('AM1001','Jeffry','McQuil','1650.13','2');**

**INSERT INTO worker VALUES('AM1002','Bob','Jefferson','2000.00','3');**

**INSERT INTO worker VALUES('AM1003','Michelle','Obama','1332.84','2');**

**INSERT INTO worker VALUES('AM1004','Anastasia','Oleksei','1580.00','2');**

**INSERT INTO worker VALUES('AM1005','John','Papamixahl','3200.35','1');**

**INSERT INTO worker VALUES('AM1006','Dixie','Wimpleton','1600.00','1');**

**INSERT INTO worker VALUES('AM1007','Willy','Deniro','1453.00','1');**

**INSERT INTO worker VALUES('AM1008','Thomas','Johnson','3920.00','1');**

**INSERT INTO worker VALUES('AM1009','Patrick','Asterias','1500.20','3');**

**INSERT INTO worker VALUES('AM1010','Zoe','Anniston','1903.50','3');**

**INSERT INTO worker VALUES('AM1011','Thomas','Pit','1244.50','3');**

**INSERT INTO worker VALUES('AM1012','Eugene','Guderian','1994.23','3');**

**INSERT INTO worker VALUES('AM1013','Helen','McGuill','2354.54','2');**

**INSERT INTO worker VALUES('AM1014','Maria','Papadopoulou','2123.22','2');**

**INSERT INTO worker VALUES('AM1015','Virginia','Barbory','1255.90','2');**

**INSERT INTO worker VALUES('AM1016','Takis','Tsoukalas','1069.00','3');**

**INSERT INTO worker VALUES('AM1017','Vasilis','Theodoroy','2531.98','3');**

**INSERT INTO worker VALUES('AM1018','Haris','Kikidis','5132.98','3');**

**-- GUIDE INSERT--**

**INSERT INTO guide VALUES ('AM1000','Tour Guide at Patras Greece and Achaia Area.8 year of expertice.Patience,teamwork and effectiveness are guaranteed.Fluent Knowledge in Greek,English and French');**

**INSERT INTO guide VALUES ('AM1001','Tour Guide in Athens and expert at ancient Greek times. 4 Years of Parthenon tours and 2 in the great relics located in the city of Athens.Fluent Knowledge in Greek,English and Italian');**

**INSERT INTO guide VALUES ('AM1016','Tour Guide in Thessalonikh.5 years of experience and in a proffesional agency. Amazing reviews in TripAdvisor and TripGuide.Fluent Knowledge in Greek,English,Italian and German');**

**insert into guide values ('AM1017','Tour Guide operating in Banshko,Bulgaria. Showcase of the city monuments and the famous ski resort.Fluent Knowledge of Greek,English,Bulgarian and Turkish');**

**-- LANGUAGES--**

**INSERT INTO languages VALUES('AM1000','Greek,English and French');**

**INSERT INTO languages VALUES('AM1001','Greek,English and Italian');**

**INSERT INTO languages VALUES('AM1016','Greek,English,German');**

**INSERT INTO languages VALUES('AM1017','Greek,English,Bulgarian');**

**-- ADMIN INSERT --**

**INSERT INTO admin VALUES('AM1002','LOGISTICS','Bachelor of Science in Logistics and Supply Chain Management from University of Patras');**

**INSERT INTO admin VALUES('AM1005','ADMINISTRATIVE','Masters Degree on Managment and Human Organization from Panteion University of Athens');**

**INSERT INTO admin VALUES('AM1014','ADMINISTRATIVE','Masters Degree on Managment and Business Expansion from Aristoteleion University of Thessalonikh');**

**INSERT INTO admin VALUES('AM1010','ADMINISTRATIVE','Masters Degree on Managment and Business Expansion from Aristoteleion University of Thessalonikh');**

**INSERT INTO admin VALUES('AM1003','ACCOUNTING','Bachelors Degree in Business Accounting from TEI Patras');**

**INSERT INTO admin VALUES('AM1006','ACCOUNTING','Bachelors Degree in Business Accounting from TEI Patras');**

**-- DRIVER INSERT--**

**INSERT INTO driver VALUES('AM1004','A','LOCAL','4');**

**INSERT INTO driver VALUES('AM1007','B','ABROAD','3');**

**INSERT INTO driver VALUES('AM1008','C','LOCAL','5');**

**INSERT INTO driver VALUES('AM1009','A','LOCAL','2');**

**INSERT INTO driver VALUES('AM1011','D','ABROAD','10');**

**INSERT INTO driver VALUES('AM1012','C','LOCAL','2');**

**INSERT INTO driver VALUES('AM1013','A','LOCAL','1');**

**INSERT INTO driver VALUES('AM1015','C','LOCAL','7');**

**-- TRIP INSERT--**

**INSERT INTO trip VALUES(null,'2022-7-1','2022-7-5','50','2600','1','AM1000','AM1004');**

**INSERT INTO trip VALUES(null,'2022-7-6','2022-7-10','100','1500','1','AM1000','AM1009');**

**INSERT INTO trip VALUES(null,'2022-7-9','2022-7-15','50','2800','2','AM1001','AM1008');**

**INSERT INTO trip VALUES(null,'2022-8-11','2022-8-18','87','1000','1','AM1000','AM1009');**

**INSERT INTO trip VALUES(null,'2022-7-16','2022-7-19','90','3500','2','AM1001','AM1013');**

**INSERT INTO trip VALUES(null,'2022-6-20','2022-6-30','77','1800','2','AM1001','AM1015');**

**INSERT INTO trip VALUES(null,'2022-6-2','2022-6-8','77','1800','3','AM1016','AM1012');**

**INSERT INTO trip VALUES(null,'2022-7-2','2022-7-11','56','1235','3','AM1016','AM1008');**

**INSERT INTO trip VALUES(null,'2022-01-19 00:00:00','2022-01-20 00:00:00','70','1453.34','3','AM1017','AM1007'); -- bulgaria**

**INSERT INTO trip VALUES(null,'2022-1-10','2022-1-18','53','1342','3','AM1017','AM1007'); -- bulgaria**

**INSERT INTO trip VALUES(null,'2022-1-20','2022-1-28','95','2380','3','AM1017','AM1011'); -- bulgaria**

**-- EVENT INSERT--**

**INSERT INTO event VALUES(1,'2022-07-01','2022-07-05','Trip to the ancient ruins from the once great cities of Patra,Aigio and Pyrgos. Food at local taverns and walk around the modern cities');**

**INSERT INTO event Values(2,'2022-07-06','2022-07-10','Event at the archaiological museum of Patras,the ancient Ruins,the castle and the Roman aqueduct.');**

**INSERT INTO event values(4,'2022-08-11','2022-08-18','Event at Hotel Astir where local artists and musicians will perform.Buffet and drinks included'); -- patra**

**insert into event values(3,'2022-07-09 00:00:00','2022-07-15 00:00:00','Event at the brand new archaiological museum of Athens before visiting the Parthenon.Walkaround the old market and the local shops.Visitation of the National observatory and Stavros Niarhos foundation ');**

**insert into event values(5,'2022-07-16 00:00:00','2022-07-19 00:00:00','Tour guides and local historians will showcase the great battle of Marathonas that took place in 490 B.C against the Persian Empire.');**

**insert into event values(6,'2022-06-20 00:00:00','2022-06-30 00:00:00','Visiting the most famous beaches alongside the beautiful green forests of Euvoia. Quick trip to the historical cities of Eretreia and Chalkida.');**

**insert into event values(7,'2022-06-02 00:00:00','2022-06-08 00:00:00','Event at the White Tower of Thessaloniki where a music bands will perform and walk around Aristotelous square');**

**insert into event values(8,'2022-07-02 00:00:00','2022-07-11 00:00:00','Visiting the grave of Philipp II,father of Alexander the Great, in Vergina and showcase in the modern museum featuring Macedonian equipment as well as normal day items in ancient Greece');**

**insert into event values(9,'2022-01-10 00:00:00','2022-01-18 00:00:00','Fun day at the famous Banshko ski resort. Followed by spa and sauna in mid-day and drinks at the evening party hosted by us.');**

**insert into event values(10,'2022-01-20 00:00:00','2022-01-28 00:00:00','Visiting the historical city of Sofia with remnats of the Byzantine Empire remaining in the city. Impressive architecture as the city has existed for more than 8000 years.');**

**-- MANAGES INSERT**

**INSERT INTO manages VALUES('AM1005','1');**

**INSERT INTO manages VALUES('AM1014','2');**

**INSERT INTO manages VALUES('AM1010','3');**

**-- PHONES INSERT--**

**INSERT INTO phones VALUES('1','2610522510');**

**INSERT INTO phones VALUES('2','210412210');**

**INSERT INTO phones VALUES('3','2310549218');**

**-- DESTINATION INSERT--**

**INSERT INTO destination VALUES(1,'Achaia','Visiting the cities of Patras,Pyrgos and Aigio','LOCAL',1,'Greek');**

**INSERT INTO destination VALUES(2,'Patra','Visiting the city of Patras and its ancient monuments','LOCAL',1,'Greek');**

**INSERT INTO destination VALUES(3,'Patra','Visiting the city of Patras and Achaia Claus','LOCAL',1,'Greek');**

**INSERT INTO destination VALUES(4,'Athens','Visiting the City of Athens and its monuments','LOCAL',4,'Greek');**

**INSERT INTO destination VALUES(5,'Marathonas','Visiting the location of Marathonas','LOCAL',4,'GREEK');**

**INSERT INTO destination VALUES(6,'Euvoia','Visiting the location of Euvoia','LOCAL',4,'GREEK');**

**INSERT INTO destination VALUES(7,'Thessaloniki','Visiting Thessaloniki and the areas around it','LOCAL',7,'GREEK');**

**INSERT INTO destination VALUES(8,'Vergina','Historical city of the Macedonian Empire and the grave of Philipp II','LOCAL',7,'GREEK');**

**INSERT INTO destination VALUES(9,'Bulgaria','Visiting the city of Banshko and its famous ski resort','ABROAD',9,'BULGARIAN');**

**INSERT INTO destination VALUES(10,'Sofia','City of Sofia,capital of Bulgaria','ABROAD',9,'BULGARIAN');**

**-- TRAVEL\_TO--**

**INSERT INTO travel\_to VALUES(1,1,'2022-07-01','2022-07-05 00:00:00');**

**INSERT INTO travel\_to VALUES(2,2,'2022-07-06 00:00:00','2022-07-10 00:00:00');**

**INSERT INTO travel\_to VALUES(4,3,'2022-08-11 00:00:00','2022-08-18 00:00:00'); -- patra--**

**INSERT INTO travel\_to VALUES(3,4,'2022-07-09 00:00:00','2022-07-15 00:00:00');**

**INSERT INTO travel\_to VALUES(5,5,'2022-07-16 00:00:00','2022-07-19 00:00:00');**

**INSERT INTO travel\_to VALUES(6,6,'2022-06-20 00:00:00','2022-06-30 00:00:00'); -- athina--**

**INSERT INTO travel\_to VALUES(7,7,'2022-06-02 00:00:00','2022-06-08 00:00:00');**

**INSERT INTO travel\_to VALUES(8,8,'2022-07-02 00:00:00','2022-07-011 00:00:00'); -- thessaloniki--**

**INSERT INTO travel\_to VALUES(9,9,'2022-01-10 00:00:00','2022-01-18 00:00:00');**

**INSERT INTO travel\_to VALUES(10,10,'2022-01-20 00:00:00','2022-01-28 00:00:00'); -- bulgaria**

**-- RESERVATION**

**INSERT INTO reservation values(1,4,'Georgios','Papadakis','ADULT');**

**INSERT INTO reservation values(2,2,'Maximos','Anastasioy','ADULT');**

**INSERT INTO reservation values(3,4,'Marios','Eskere','ADULT');**

**INSERT INTO reservation values(5,5,'Xaralampos','Papadopoylos','ADULT');**

**INSERT INTO reservation values(4,3,'Tasos','kekeogloy','ADULT');**

**INSERT INTO reservation values(6,2,'Lefteris','Pantazopoylos','ADULT');**

**INSERT INTO reservation values(7,3,'Elefteria','Vasilakopoyloy','MINOR');**

**INSERT INTO reservation values(8,4,'Kwnstantinos','Palaiologos','ADULT');**

**INSERT INTO reservation values(9,5,'Tommy','Shelby','MINOR');**

**INSERT INTO reservation values(10,5,'Charis','Lesjho','ADULT');**

**insert into reservation values(11,1,'Vasilis','Pasxalhs','MINOR');**

**INSERT INTO reservation values(1,15,'Itzel','Lesjho','ADULT');**

**INSERT INTO reservation values(2,20,'Harper','Malcolm','ADULT');**

**INSERT INTO reservation values(2,11,'Jazmine','Moyer','ADULT');**

**INSERT INTO reservation values(2,7,'Gaige','Moyer','MINOR');**

**INSERT INTO reservation values(3,6,'Gaige','Moyer','MINOR');**

**INSERT INTO reservation values(3,2,'Augustus','Kent','MINOR');**

**INSERT INTO reservation values(3,1,'Frederick','Kent','ADULT');**

**INSERT INTO reservation values(4,10,'Samir','Miranda','ADULT');**

**INSERT INTO reservation values(4,11,'Jamir','Samir','ADULT');**

**INSERT INTO reservation values(5,10,'Mehmet','Ali','MINOR');**

**INSERT INTO reservation values(5,11,'Mohammed','Lester','ADULT');**

**INSERT INTO reservation values(5,14,'Noah','Arcs','MINOR');**

**INSERT INTO reservation values(6,10,'Yasmin','Goodwin','ADULT');**

**INSERT INTO reservation values(6,11,'Saul','Goodman','ADULT');**

**INSERT INTO reservation values(6,12,'Jessy','White','ADULT');**

**INSERT INTO reservation values(6,13,'Skyler','White','ADULT');**

**INSERT INTO reservation values(6,15,'Walker','Junior','MINOR');**

**INSERT INTO reservation values(7,10,'Arthur','Shelby','ADULT');**

**INSERT INTO reservation values(7,11,'Paul','Walker','ADULT');**

**INSERT INTO reservation values(7,12,'TheRock','Opetras','ADULT');**

**INSERT INTO reservation values(7,13,'Losky','Vlospa','ADULT');**

**INSERT INTO reservation values(8,10,'Gilberto','Berry','MINOR');**

**INSERT INTO reservation values(8,11,'Arthur','Shelby','ADULT');**

**INSERT INTO reservation values(8,12,'Arthur','McAlister','ADULT');**

**INSERT INTO reservation values(9,10,'Eleni','Mamasoyla','ADULT');**

**INSERT INTO reservation values(9,11,'Georgia','Basa','MINOR');**

**INSERT INTO reservation values(9,12,'Koug','Dimi','MINOR');**

**INSERT INTO reservation values(10,7,'Ishain','Buckley','ADULT');**

**INSERT INTO reservation values(10,4,'Josue','Rich','ADULT');**

**INSERT INTO reservation values(11,4,'Gregory','Olsen','ADULT');**

**INSERT INTO reservation values(11,10,'Mateos','Olsen','ADULT');**

**INSERT INTO reservation values(11,12,'Josef','Stalin','MINOR');**

**-- ITDEPARTMENT**

**INSERT INTO itdepartment VALUES('AM1018','password','2022-01-01',NULL);**

**DROP TABLE IF EXISTS offers;**

**CREATE TABLE IF NOT EXISTS offers**

**(**

**id\_prosforas INT NOT NULL AUTO\_INCREMENT,**

**start\_date DATETIME NOT NULL,**

**end\_date DATETIME NOT NULL,**

**cost\_per\_person FLOAT(7,2) NOT NULL,**

**destination INT(11) NOT NULL,**

**PRIMARY KEY(id\_prosforas),**

**CONSTRAINT offers\_fk FOREIGN KEY (destination) REFERENCES destination(dst\_id)**

**ON UPDATE CASCADE ON DELETE CASCADE**

**);**

**DROP TABLE IF EXISTS offers\_reservation;**

**CREATE TABLE IF NOT EXISTS offers\_reservation**

**(**

**id\_kratisis INT NOT NULL AUTO\_INCREMENT,**

**lname VARCHAR(30) NOT NULL,**

**fname VARCHAR(30) NOT NULL,**

**id\_prosforas\_reserve INT NOT NULL,**

**prokatavoli FLOAT(7,2),**

**PRIMARY KEY(id\_kratisis),**

**CONSTRAINT id\_prosforas\_reserve\_FK FOREIGN KEY (id\_prosforas\_reserve) REFERENCES offers(id\_prosforas)**

**ON UPDATE CASCADE ON DELETE CASCADE**

**);**

**INSERT INTO offers VALUES(NULL,'2022-02-01','2022-02-28',350.40,1);**

**INSERT INTO offers VALUES(NULL,'2022-04-01','2022-04-30',1500.30,4);**

**INSERT INTO offers VALUES(NULL,'2022-05-10','2022-05-25',864.28,7);**

**ΚΕΦΑΛΑΙΟ 2:ΠΑΡΑΔΕΙΓΜΑΤΑ ΚΑΙ ΚΩΔΙΚΑΣ ΑΠΟ STORED PROCEDURES**

3.1.3.1 STORED PROCEDURE)

Το procedure αυτό θα πρέπει να εντοπίζει το branch με τους λιγότερους οδηγούς και να τον τοποθετεί εκεί.

Graphical user interface, text, application, Word, email

Description automatically generated

Όπως βλέπετε στο παραπάνω screenshot αυτό επιτυγχάνεται,καθώς το branch με τους λιγότερους οδηγούς πριν την εισαγωγή του νέου είναι το Α και μέτα το Call NewDriver με ένα select \* βλέπουμε τα αποτελέσματα.

Το Stored Procedure παίρνει 5 ορίσματα και με την χρήση του subquerie Q1 μέσα στην from επιστρέφει τον μικρότερο αριθμό από drivers σε μία μεταβλητή drvnumfinal.

Έπειτα με βάση αυτόν τον αριθμό ψάχνει το κατάστημα που ανήκει αυτος ο αριθμος και τοποθετεί τον driver εκεί.

**ΚΩΔΙΚΑΣ 3.1.3.1:**

-- STORED PROCEDURE

DELIMITER $$

DROP PROCEDURE IF EXISTS NewDriver$$ -- at , onoma,epwnymo,misthos,typos adeias,typos dromologioy,mines empeirias

CREATE PROCEDURE NewDriver(IN driverAT CHAR(10),IN firstname VARCHAR(20),IN lastname VARCHAR(20),IN misthos FLOAT(7,2),IN driverlicense ENUM('A','B','C','D'),IN driverroute ENUM('ABROAD','LOCAL'),IN driverexp TINYINT)

BEGIN

DECLARE drvnumfinal INT;

DECLARE branchnum INT;

DECLARE dumbtest INT;

SELECT MIN(drvnumber)

INTO drvnumfinal

FROM (SELECT wrk\_br\_code as wrk\_br,COUNT(wrk\_br\_code) as drvnumber -- kanei ena prwto from kai meta apo auto to from epilegei to minimum

FROM worker INNER JOIN driver ON wrk\_AT=drv\_AT

GROUP BY wrk\_br\_code) AS Q1; -- epistrefei to mikrotero arithmo apo drivers

SELECT wrk\_br\_code,Count(\*) as counter

INTO branchnum,dumbtest

FROM driver INNER JOIN worker ON drv\_AT=wrk\_AT

GROUP BY wrk\_br\_code

HAVING counter=drvnumfinal -- entopizei to katastima poy exei ton arithmo me toys ligoteroys drivers sto branchnum

ORDER BY wrk\_br\_code;

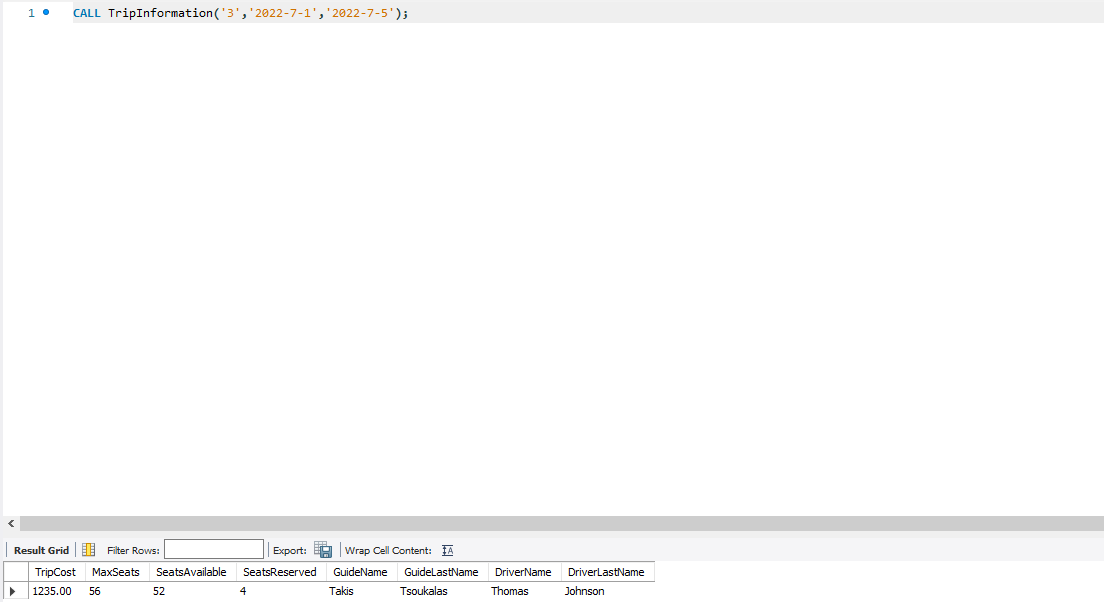
INSERT INTO worker VALUES(driverAT,firstname,lastname,misthos,branchnum);

INSERT INTO driver VALUES(driverAT,driverlicense,driverroute,driverexp);

END$$

DELIMITER ;

3.1.3.2 STORED PROCEDURE)



**ΠΕΡΙΓΡΑΦΗ:**

Με την TripInformation επιστρέφεται το ταξίδι που είναι σε αυτές τις ημερομηνίες ανάμεσα. Εάν είναι πολλα τα trips επιστρέφονται πολλα ταξίδια σε διαφορετικά tabs και ο χρήστης μπορεί να τα κοιτάξει ολα.Για την δημίουργια αυτού του stored procedure έκανα μία μικρή τροποποίηση από τα procedures στην άσκηση 5 και τα εφτίαξα έτσι ώστε το ένα να μας δίνει τις άδειες θέσεις και το άλλο των κατοχηρωμένων θέσεων. Τα συνδυάζω και τα δύο στο TripInformationκαι μαζί με ενα fetch που κάνει προσπέλαση τους returned πίνακες επιστρέφεται το σωστό αποτέλεσμα στον χρήστη.

**ΚΩΔΙΚΑΣ:**

-- PROCEDURE POY MAS DINEI TIS RESERVED THESEIS

DELIMITER $$

DROP PROCEDURE IF EXISTS ReservedSeats$$

CREATE PROCEDURE ReservedSeats(IN id\_trip INT,OUT eksodos INT)

BEGIN

DECLARE seatnummax INT;

DECLARE reservedseats INT;

SELECT COUNT(res\_tr\_id),trip.tr\_maxseats

INTO reservedseats,seatnummax

FROM reservation INNER JOIN trip ON reservation.res\_tr\_id=trip.tr\_id

WHERE res\_tr\_id=id\_trip

GROUP BY res\_tr\_id;

IF (reservedseats>=seatnummax) THEN

SIGNAL SQLSTATE VALUE '45000'

SET MESSAGE\_TEXT='There are no available seats for this trip';

ELSE

SET eksodos=reservedseats;

END IF;

END $$

DELIMITER ;

-- PROCEDURE POY MOY EPISTREFEI TON ARITHMO TWN KENWN THESEWN SE ENA TRIP

DELIMITER $$

DROP PROCEDURE IF EXISTS SeatEmptyCounter$$

CREATE PROCEDURE SeatEmptyCounter(IN id\_trip INT,OUT eksodos INT)

BEGIN

DECLARE seatnummax INT;

DECLARE reservedseats INT;

SELECT COUNT(res\_tr\_id),trip.tr\_maxseats

INTO reservedseats,seatnummax

FROM reservation INNER JOIN trip ON reservation.res\_tr\_id=trip.tr\_id

WHERE res\_tr\_id=id\_trip

GROUP BY res\_tr\_id;

IF (reservedseats>=seatnummax) THEN

SIGNAL SQLSTATE VALUE '45000'

SET MESSAGE\_TEXT='There are no available seats for this trip';

ELSE

SET eksodos=seatnummax-reservedseats;

END IF;

END $$

DELIMITER ;

DELIMITER $$

DROP PROCEDURE IF EXISTS TripInformation$$

CREATE PROCEDURE TripInformation(IN branchid INT,IN date1 DATE,IN date2 DATE)

BEGIN

DECLARE Trip\_ID\_return INT;

DECLARE anaxwrhsh DATE;

DECLARE epistrofh DATE;

DECLARE finished INT;

DECLARE SeatsAvailable INT;

DECLARE SeatsReserved INT;

DECLARE GuideAM CHAR(10);

DECLARE DriverAM CHAR(10);

DECLARE GuiName VARCHAR(20);

DECLARE GuiLname VARCHAR(20);

DECLARE DriverName VARCHAR(20);

DECLARE DriverLname VARCHAR(20);

DECLARE theseismax INT;

DECLARE tripcost FLOAT(7,2);

/\* arxiko date<departure kai departure<teliko

gia na min piasei ola mexri telos poy einai megalutera\*/

DECLARE TripIDCursor CURSOR FOR

SELECT tr\_id,tr\_departure,tr\_return,tr\_gui\_AT,tr\_drv\_AT,tr\_cost,tr\_maxseats FROM trip WHERE tr\_br\_code=branchid AND date1<tr\_departure AND tr\_departure<date2; -- fairnei kathe eggrafh me br\_code=branchid poy dinoyme

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished=1;

OPEN TripIDCursor;

SET finished=0;

REPEAT

FETCH TripIDCursor INTO Trip\_ID\_return,anaxwrhsh,epistrofh,GuideAM,DriverAM,tripcost,theseismax;

IF(finished=0) THEN

CALL ReservedSeats(Trip\_ID\_return,@SeatsReserved); -- epistrefei to sunolo twn kratimenwn thesewn

CALL SeatEmptyCounter(Trip\_ID\_return,@SeatsAvailable); -- epistrefei tis kenes theseis available

SELECT wrk\_name,wrk\_lname

INTO GuiName,GuiLname -- stoixeia guide

FROM worker

WHERE wrk\_AT=GuideAM;

SELECT wrk\_name,wrk\_lname

INTO DriverName,DriverLname

FROM worker -- stoixeia driver

WHERE wrk\_AT=DriverAM;

SELECT tripcost AS TripCost,theseismax AS MaxSeats,@SeatsAvailable AS SeatsAvailable,@SeatsReserved AS SeatsReserved,GuiName AS GuideName,GuiLname AS GuideLastName,DriverName,DriverLname as DriverLastName;

END IF;

UNTIL(finished=1)

END REPEAT;

CLOSE TripIDCursor;

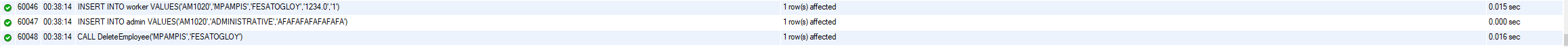
END$$

DELIMITER ;

**3.1.3.3 STORED PROCEDURE)**

**ΠΕΡΙΓΡΑΦΗ:**

Το procedure αυτό ήταν αρκετά απλό.Το μόνο περίπλοκο θέμα ηταν η περίπτωση που δύο άτομα εχουν το ιδιο ονομα και ο ένας είναι adminstrative και ο αλλος όχι γιαυτο ελέγχω και το AM για καθε προσπέλαση εγγραφής που γίνεται από το repeat για να εξασφαλίσω την σωστή διαγραφή έαν χρειαστεί. Στην συνέχεια εισάγω έναν admin για να επιδείξω την διαγραφή ότι λειτουργεί σωστά.



Text

Description automatically generated with low confidence

**ΚΩΔΙΚΑΣ:**

DELIMITER $$

DROP PROCEDURE IF EXISTS DeleteEmployee$$

CREATE PROCEDURE DeleteEmployee(IN fname VARCHAR(20),IN lname VARCHAR(20))

BEGIN

DECLARE firstname VARCHAR(20);

DECLARE lastname VARCHAR(20);

DECLARE amcheck char(10);

DECLARE finishedflag INT;

DECLARE jobtype TEXT;

DECLARE ismanager char(10);

DECLARE StatusCheck CURSOR FOR

SELECT wrk\_AT,wrk\_name,wrk\_lname FROM worker WHERE wrk\_name=fname AND wrk\_lname=lname;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

OPEN StatusCheck;

SET finishedflag=0;

-- xreiazomaste cursor dioti mporei na yparxei 2 atoma me to idio onoma

REPEAT

FETCH StatusCheck INTO amcheck,firstname,lastname;

IF (finishedflag=0) THEN

SELECT adm\_type

INTO jobtype

FROM admin

WHERE adm\_AT=amcheck; -- checkarei ton type toy work sto admin

SELECT mng\_adm\_AT

INTO ismanager

FROM manages -- epistrefei to AM an yparxei sto manages alliws gurizei NULL

WHERE mng\_adm\_AT=amcheck;

CASE

WHEN (jobtype='ADMINISTRATIVE' AND ismanager IS NOT NULL) THEN SELECT 'USER IS THE MANAGER OF A BRANCH AND DELETION IS NOT ALLOWED' AS OUTPUT;

WHEN (jobtype='ADMINISTRATIVE') THEN DELETE FROM worker WHERE wrk\_AT=amcheck;

ELSE SELECT 'USER IS NOT AN ADMINISTRATOR' AS OUTPUT;

END CASE;

END IF;

UNTIL(finishedflag=1)

END REPEAT;

CLOSE StatusCheck;

END$$

DELIMITER ;

INSERT INTO worker VALUES('AM1020','MPAMPIS','FESATOGLOY','1234.0','1');

INSERT INTO admin VALUES('AM1020','ADMINISTRATIVE','AFAFAFAFAFAFAFA');

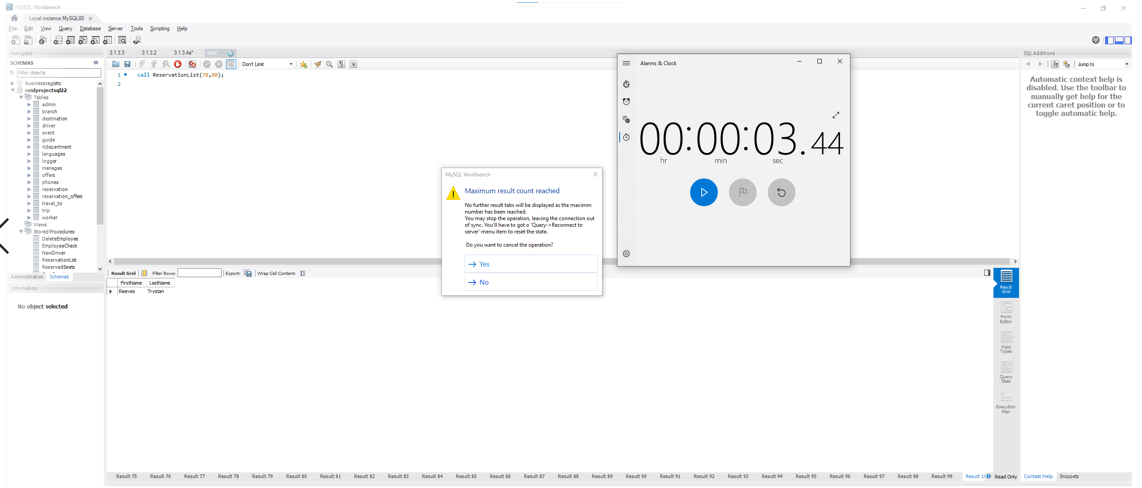
CALL DeleteEmployee('John','Papamixahl');

**3.1.3.4 STORED PROCEDURE)**

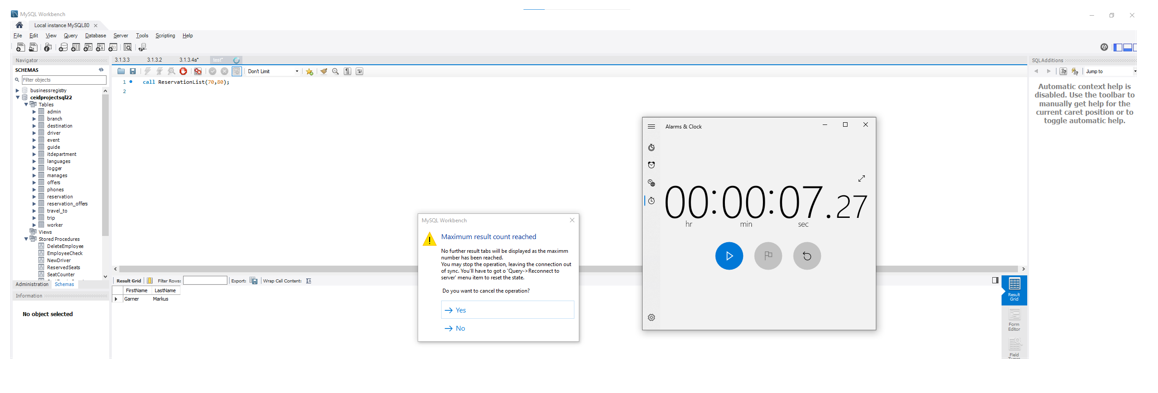
Δημιουργώ απλα δυο index στα πεδία που μας δίνετε και τρέχω τα procedures για να δω τα απολέσματα. Δεν έχουν κάτι ιδιαίτερο τα procedures αυτά απο θέμα υλοποίησης.

**ΤΟ Α:**

**Με ευρετήριο.**

****

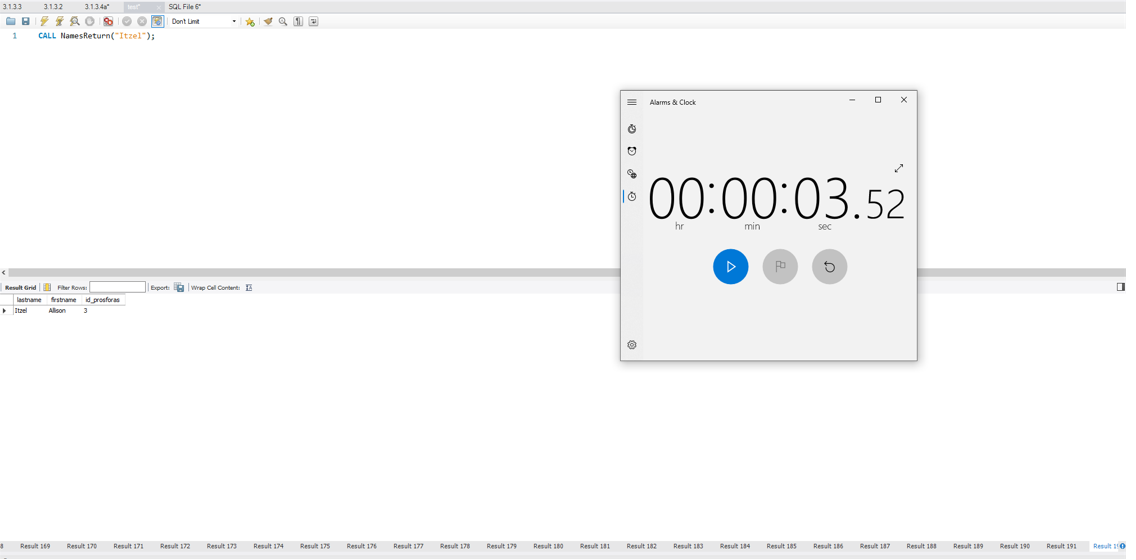
**Χωρίς ευρετήριο.**

****

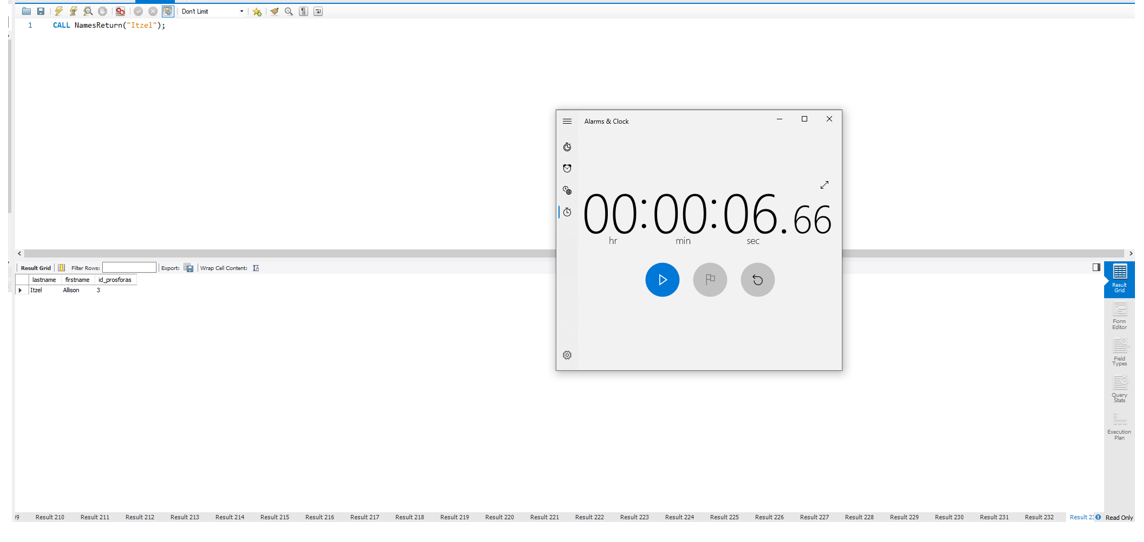
**Η διαφορά είναι σημαντική**

**Το Β:**

**Με ευρετήριο.**

****

**Χωρίς ευρετήριο**

****

**Πάλι σημειώνεται σημαντική διαφορά.**

**ΚΩΔΙΚΑΣ:**

-- DROP INDEX reservation\_index ON reservation\_offers;

CREATE INDEX reservation\_index

ON reservation\_offers (prokatavoli);\*/ -- thelei me index kai xwris index

DELIMITER $$

DROP PROCEDURE IF EXISTS ReservationList$$

CREATE PROCEDURE ReservationList(IN price1 FLOAT(7,2),IN price2 FLOAT(7,2))

BEGIN

DECLARE firstname VARCHAR(30);

DECLARE lastname VARCHAR(30);

DECLARE flag INT;

DECLARE ReservationCursor CURSOR FOR

SELECT fname,lname FROM reservation\_offers WHERE prokatavoli BETWEEN price1 AND price2;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET flag=1;

OPEN ReservationCursor;

SET flag=0;

REPEAT

FETCH ReservationCursor INTO firstname,lastname;

IF (flag=0)THEN

SELECT firstname as FirstName,lastname as LastName;

END IF;

UNTIL(flag=1)

END REPEAT;

CLOSE ReservationCursor;

END$$

DELIMITER ;

-- DROP INDEX namesindex ON reservation\_offers;

CREATE INDEX namesindex

ON reservation\_offers(lname);

DELIMITER $$

DROP PROCEDURE IF EXISTS NamesReturn$$

CREATE PROCEDURE NamesReturn(IN epwnumo VARCHAR(30))

BEGIN

DECLARE finishedflag INT;

DECLARE lastname VARCHAR(30);

DECLARE firstname VARCHAR(30);

DECLARE id\_prosforas INT;

DECLARE NameCursor CURSOR FOR

SELECT lname,fname,id\_prosforas\_reserve FROM reservation\_offers WHERE lname=epwnumo;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedflag=1;

OPEN NameCursor;

SET finishedflag=0;

REPEAT

FETCH NameCursor INTO lastname,firstname,id\_prosforas;

IF(finishedflag=0) THEN

SELECT lastname,firstname,id\_prosforas;

END IF;

UNTIL(finishedflag=1)

END REPEAT;

CLOSE NameCursor;

END$$

DELIMITER ;

**ΚΕΦΑΛΑΙΟ 3: TRIGGERS**

**1ST TRIGGER:**

****

DELIMITER $$

DROP TRIGGER IF EXISTS DateUpdate$$

CREATE TRIGGER DateUpdate

BEFORE UPDATE ON trip

FOR EACH ROW

BEGIN

DECLARE tripexistence INT;

SET tripexistence=NEW.tr\_id;

IF EXISTS(SELECT 1 FROM reservation WHERE res\_tr\_id=tripexistence ) THEN -- epistrefei true an yparxei record me ayto to tr\_id sto reservation pinaka

SIGNAL SQLSTATE VALUE '45000'

SET MESSAGE\_TEXT='You cannot edit this trip as there are reservations already!';

END IF;

END$$

DELIMITER ;

**2ND TRIGGER:**

UPDATE trip SET tr\_departure='2022-07-06 00:00:00',tr\_return='2022-07-15 00:00:00' WHERE tr\_id=1;



DELIMITER $$

DROP TRIGGER IF EXISTS PaycheckInsurance$$

CREATE TRIGGER PaycheckInsurance

BEFORE UPDATE ON worker

FOR EACH ROW

BEGIN

IF (OLD.wrk\_salary>NEW.wrk\_salary) THEN

SIGNAL SQLSTATE VALUE '45000'

SET MESSAGE\_TEXT='You cannot lower the salary of an employee!';

end if;

END$$

DELIMITER ;

UPDATE worker SET wrk\_salary='100.0' WHERE wrk\_AT='AM1000';

**ΚΕΦΑΛΑΙΟ 4: GUI IN JAVAFX**

Έκανα μερικό GUI σε JAVAFX καθώς έχει πάρα πολύ μεγάλο φόρτο εργασίας και είχα έλλειψη χρόνου με την εξεταστική κοντά ,αφού δούλεψα ατομικά. Έφτιαξα ένα βασικό GUI το οποίο παίρνει τα στοιχεία του it\_admin,όνομα και έναν κωδικό και συνδέεται στον τοπικό server του workbench. Έαν είναι λάθος τα στοιχεία του βγάζει παράθυρο που επιδεικνύει λάθος στοιχεία και τον προτρέπει να προσπαθήσει ξανά.

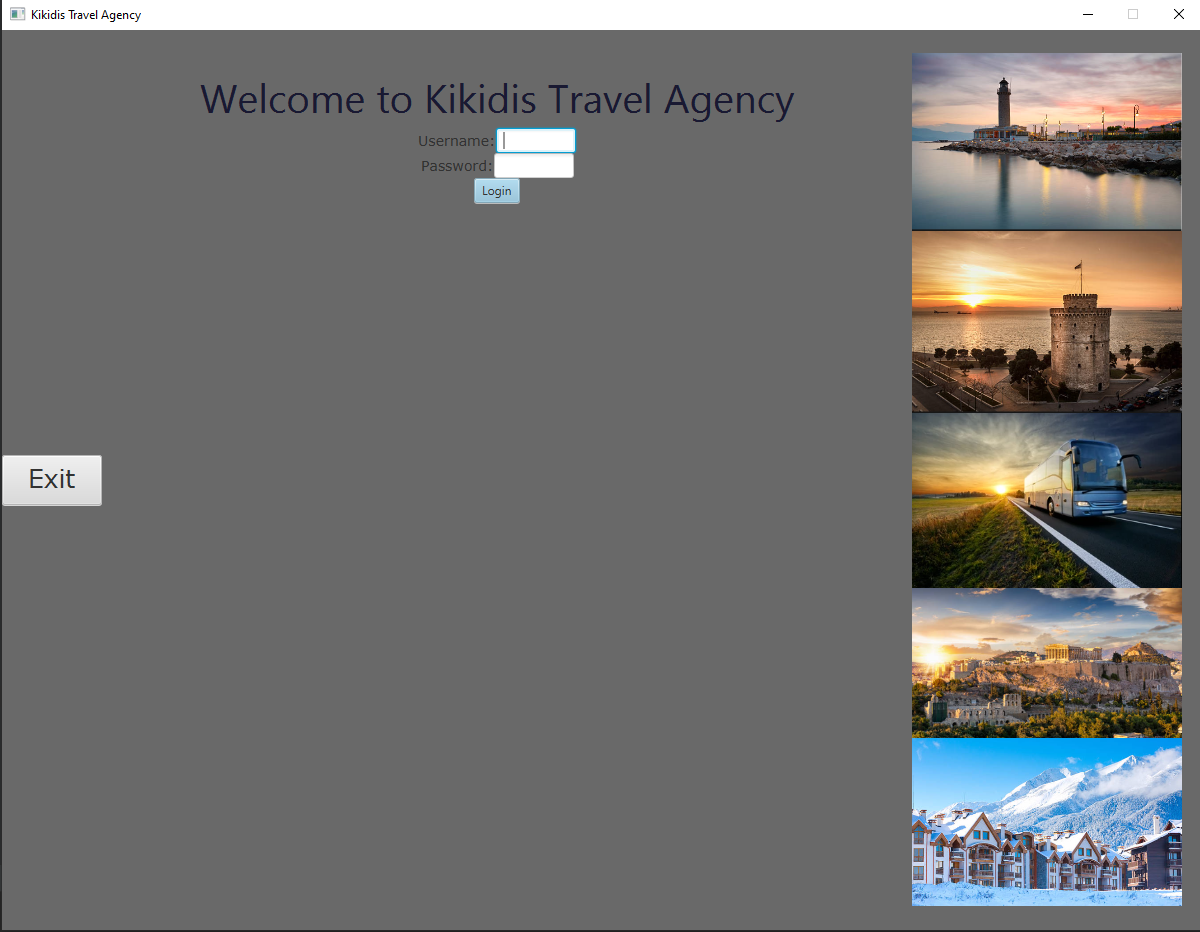
Εγώ δημιούργησα στο δικό μου local server τον χρήστη HarisKikidis που έχει δικαιώματα στην ΒΔ με κωδικό admin1234,σαν DBA,για να μην συνδεθώ με τον root. Θα πρέπει να κάνετε κάτι αντίστοιχο,άμα θέλετε να το δοκιμάσετε. Επίσης περιλαμβάνετε ένα local path που θα πρέπει να αλλαχτεί.

Έχω φτιάξει δύο κλάσεις την main και την SQLengine που θα περιέχουν αντίστοιχα τα αρχικά scenes και την λειτουργικότητα για τις INSERT,DELETE,UPDATE.

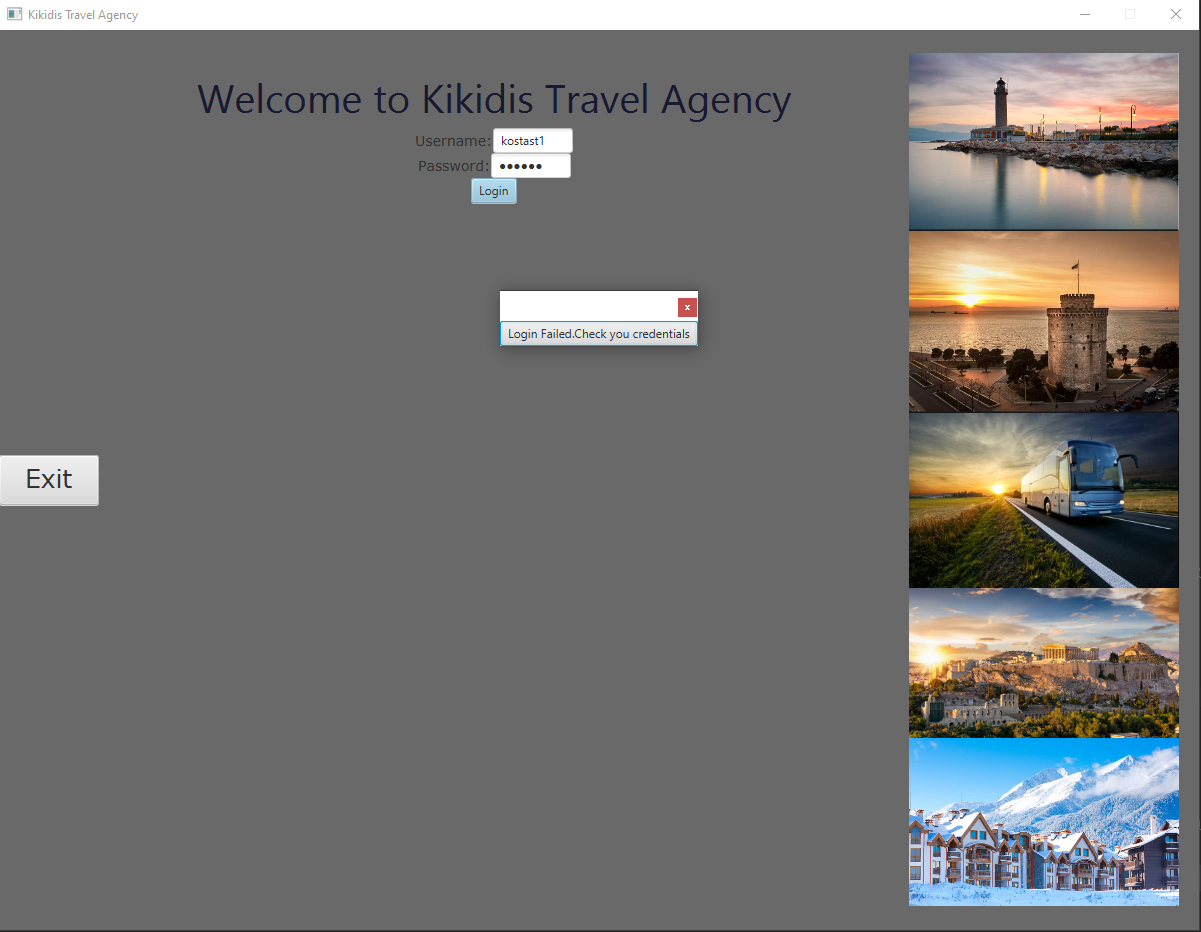
Στο αρχικό scene έχω χρησιμοποιήσει το HBOX για να χωρίσω τις διαστάσεις του παραθύρου σε 5 κομμάτια και να έχω βάλει αντίστοιχα σε κάθε κομμάτι ένα pane που περιέχει είτε Vbox ειτε Hbox. Με την σειρά του τα pane έχουν κουμπιά,labels και χρώματα. Ανάλογα με την λειτουργία κάθε κουμπιού γίνεται η αντίστοιχη ενέργεια.

Παρακάτω παραθέτω τα διάφορα screenshot και μέτα το login που ανοίγει νέο παράθυρο.

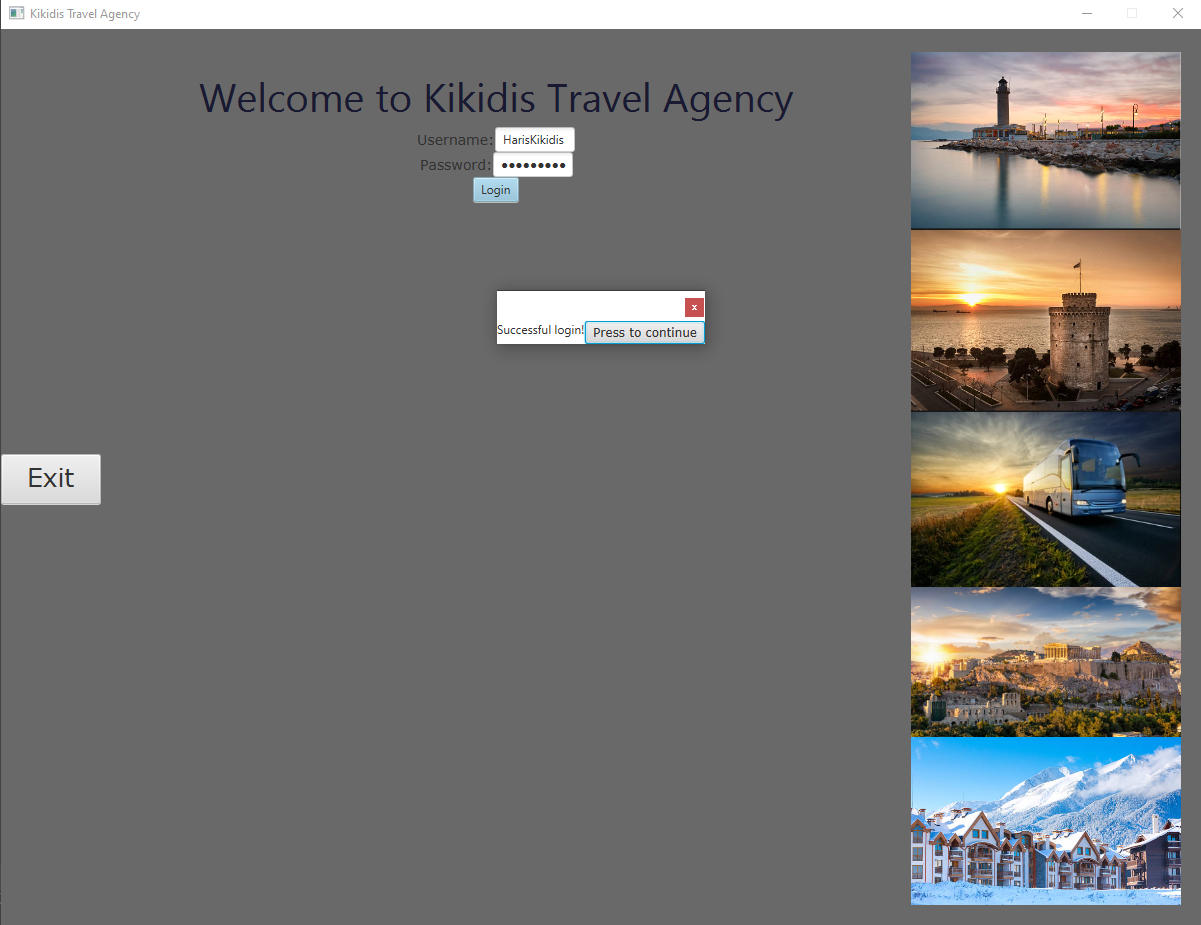
Main starting screen:

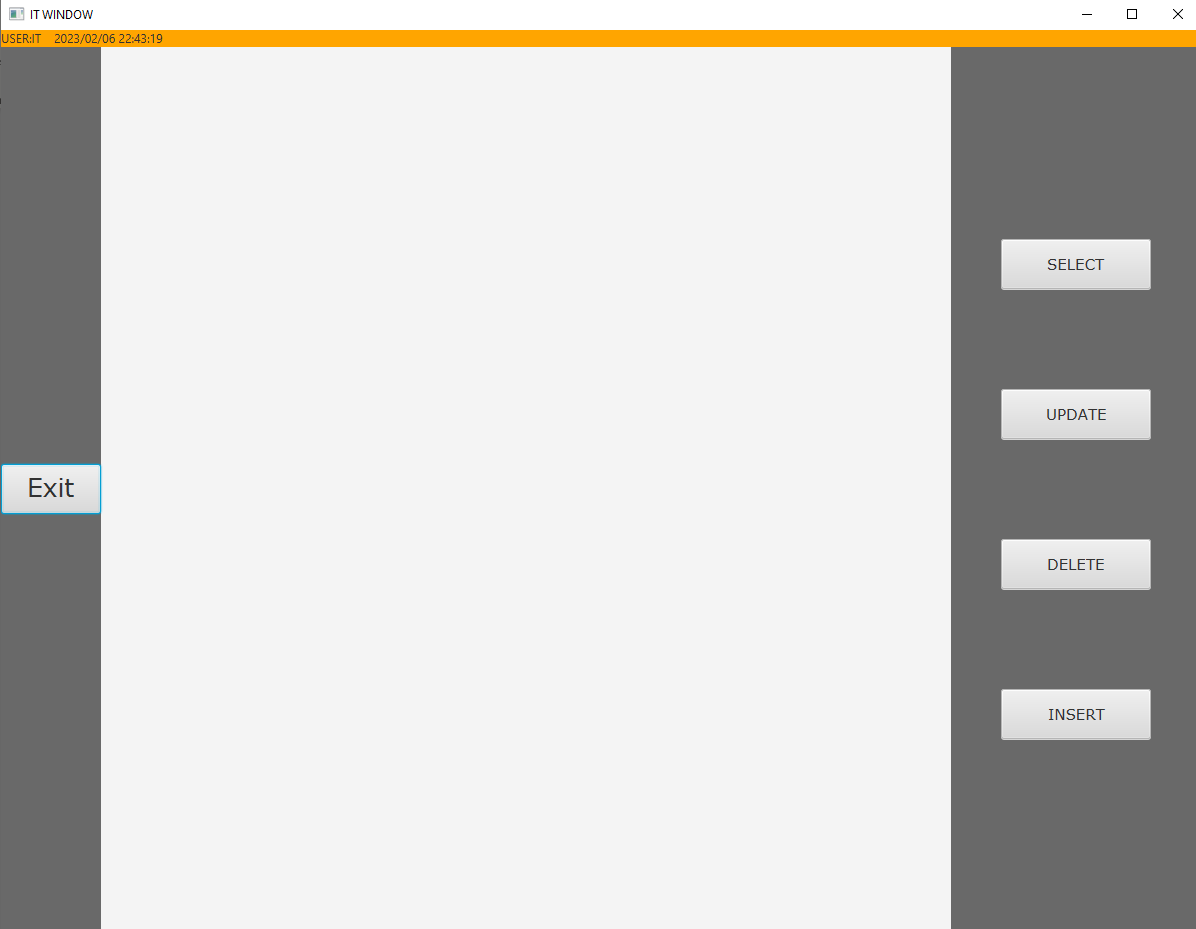


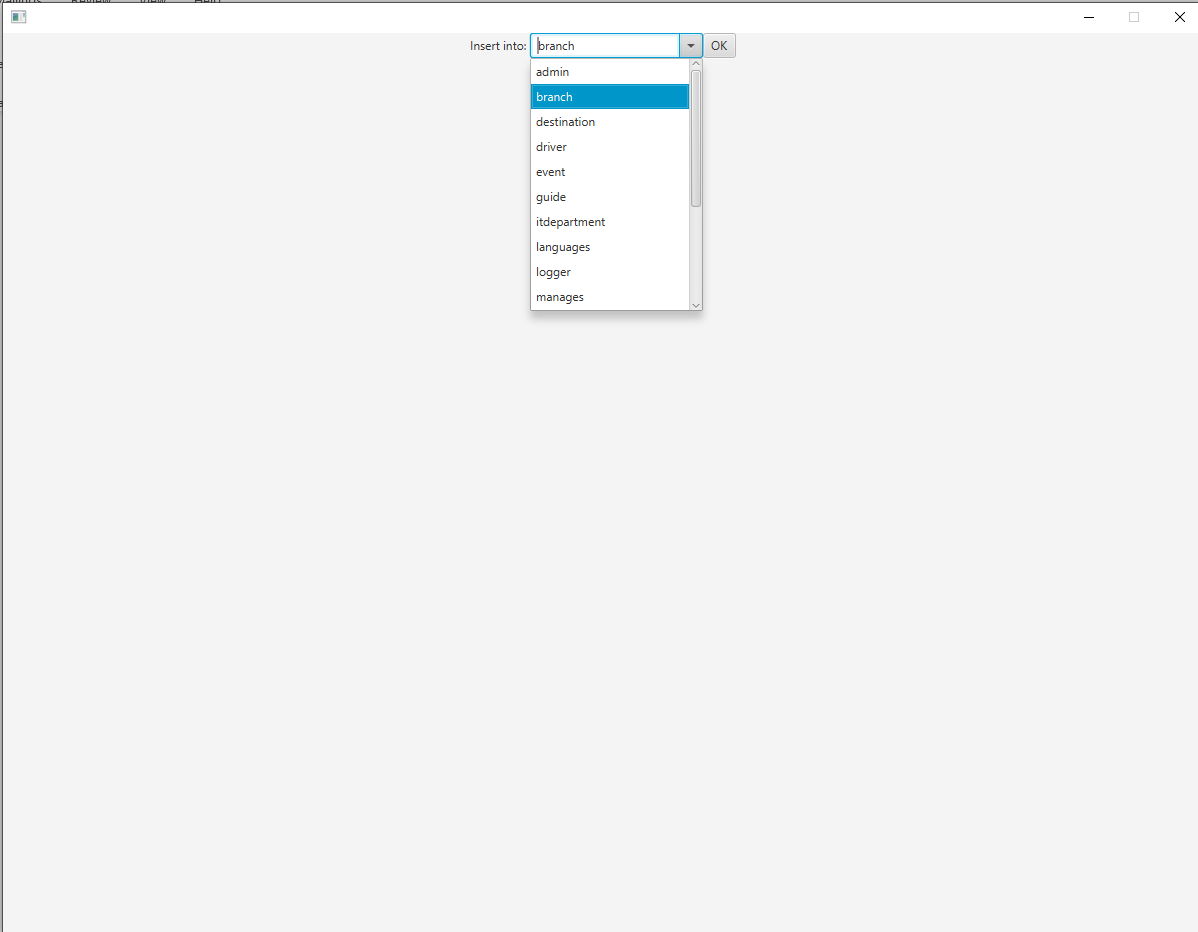
Failed login!



Successful login:



Το νεο παραθυρο που ανοιγει με επιτυχή σύνδεση.

Λειτουργικότητα του INSERT BOX:

Έδω τελειώνουν αυτά που έχω φτιάξει για το GUI λόγω έλλειψης χρόνου. Ξέρω ότι δεν είναι σοβαρό αποτέλεσμα αλλα το συμπεριέλαβα έτσι και αλλιώς για λίγα πράγματα που έχω φτιάξει.

**ΚΩΔΙΚΑΣ ΓΙΑ ΤΗΝ MAIN:**

import javafx.application.\*;  
import javafx.scene.control.Button;  
import javafx.scene.text.Font;  
import javafx.stage.\*;  
import javafx.scene.\*;  
import javafx.scene.layout.\*;  
import javafx.scene.control.\*;  
import javafx.geometry.\*;  
import javafx.scene.paint.\*;  
import java.sql.\*;  
import javafx.scene.image.\*;  
  
import javax.swing.plaf.basic.BasicViewportUI;  
  
  
public class Main extends Application {  
 TextField username;  
 PasswordField password;  
 Label loginlbl,patraslbl,athenslbl,thessalonikilbl,banshkolbl,buslbl,welcomescreen,userlbl,passwordlbl;  
 Button btnexit,btnenter;  
  
 @Override  
 public void start(Stage primaryStage) {  
 try {  
 primaryStage.setTitle("Kikidis Travel Agency"); //einai to root pane poy tha mpoyne ola ta ypoloipa  
 BorderPane root = new BorderPane();  
  
  
  
 //left edw tha valw ta buttons  
 StackPane leftPane = new StackPane();  
 leftPane.setBackground(new Background(new BackgroundFill(Color.*DIMGRAY*,CornerRadii.*EMPTY*,Insets.*EMPTY*)));  
 btnexit=new Button("Exit");  
 btnexit.setAlignment(Pos.*TOP\_CENTER*);  
 btnexit.setLayoutX(100);  
 btnexit.setLayoutY(100);  
 btnexit.setPrefSize(100,50);  
 btnexit.setFont(Font.*font*("Verdana",25));  
 btnexit.setOnAction(e->System.*exit*(0));  
 leftPane.getChildren().add(btnexit);  
 root.setLeft(leftPane);  
  
 //right edw eikones  
  
 VBox rightPane = new VBox();  
 patraslbl=new Label();  
 thessalonikilbl=new Label();  
 athenslbl=new Label();  
 banshkolbl=new Label();  
 buslbl=new Label();  
  
 Image patra=new Image("Patra.png");  
 Image thessaloniki=new Image("Thessaloniki.png");  
 Image athens=new Image("Athens.png");  
 Image banshko=new Image("banshko.png");  
 Image bus=new Image("bus.png");  
 ImageView patraview=new ImageView(patra);  
 ImageView thessalonikiview=new ImageView(thessaloniki); //eikones  
 ImageView banshkoview=new ImageView(banshko);  
 ImageView busview=new ImageView(bus);  
 ImageView athensview=new ImageView(athens);  
  
 patraview.setFitWidth(270);  
 patraview.setPreserveRatio(true);  
 patraview.setSmooth(true);  
 athensview.setFitWidth(270);  
 athensview.setPreserveRatio(true);  
 athensview.setSmooth(true);  
 thessalonikiview.setFitWidth(270);  
 thessalonikiview.setPreserveRatio(true); //edit eikones  
 thessalonikiview.setSmooth(true);  
 banshkoview.setFitWidth(270);  
 banshkoview.setPreserveRatio(true);  
 banshkoview.setSmooth(true);  
 busview.setFitWidth(270);  
 busview.setPreserveRatio(true);  
 busview.setSmooth(true);  
  
 rightPane.setBackground(new Background(new BackgroundFill(Color.*DIMGRAY*,CornerRadii.*EMPTY*,Insets.*EMPTY*)));  
 rightPane.getChildren().addAll(patraview,thessalonikiview,busview,athensview,banshkoview);  
 rightPane.setAlignment(Pos.*CENTER*);  
 rightPane.setPadding(new Insets(20));  
 root.setRight(rightPane);  
  
 //center edw tha einai ta login ktlp  
 VBox centerPane = new VBox();  
  
 centerPane.setBackground(new Background(new BackgroundFill(Color.*DIMGRAY*,CornerRadii.*EMPTY*,Insets.*EMPTY*)));  
 welcomescreen=new Label("Welcome to Kikidis Travel Agency");  
 welcomescreen.setFont(Font.*font*("Arial Bold",40));  
 welcomescreen.setTextFill(Color.*color*(0.1,0.1,0.2));  
 welcomescreen.setLayoutY(300);  
  
 userlbl=new Label("Username:");  
 passwordlbl=new Label("Password:");  
 userlbl.setFont(Font.*font*("Verdana",14));  
 passwordlbl.setFont(Font.*font*("Verdana",14));  
 username=new TextField();  
 password=new PasswordField();  
 username.setPromptText("eg KostasPapadopoylos");  
 username.setPrefWidth(80);  
 password.setPrefWidth(80);  
 btnenter=new Button("Login");  
 btnenter.setLayoutY(50);  
  
 HBox pane1=new HBox();  
 pane1.setAlignment(Pos.*CENTER*);  
 HBox pane2=new HBox(); //ftiaxnw 2 pane gia ta stoixeia toy xrhsth  
 pane2.setAlignment(Pos.*CENTER*);  
 pane1.getChildren().addAll(userlbl,username);  
 pane2.getChildren().addAll(passwordlbl,password);  
  
 btnenter.setDefaultButton(true); //molis pathsei enter o xrhsths energopoieitai to button  
 btnenter.setOnAction(e->getUserPass());  
 centerPane.getChildren().addAll(welcomescreen,pane1,pane2,btnenter);  
 centerPane.setAlignment(Pos.*TOP\_CENTER*);  
 centerPane.setPadding(new Insets(40));  
 root.setCenter(centerPane);  
  
 // scene me to BorderPane  
 Scene scene = new Scene(root, 1200, 900);  
 primaryStage.setScene(scene);  
 primaryStage.setResizable(false);  
 primaryStage.show();  
  
 }  
 catch(Exception e){  
 System.*out*.println(e.getMessage());  
 }  
 }  
 //tha checkarei an einai o admin aytos sto server kai bgazei katallhlo munima  
 public void getUserPass(){  
 Label successlbl= new Label("Successful login!");  
 Button fail=new Button("Login Failed.Check you credentials");  
 Button continuebtn =new Button("Press to continue");  
 continuebtn.setAlignment(Pos.*CENTER*);  
 continuebtn.setFont(Font.*font*("Verdana"));  
  
 HBox h1=new HBox(successlbl,continuebtn);  
 HBox h2=new HBox(fail);  
 Scene popup=new Scene(h1);  
 Scene tempscene2=new Scene(h2);  
 Pane testpane= new Pane();  
 testpane.setBackground(new Background(new BackgroundFill(Color.*BLACK*,CornerRadii.*EMPTY*,Insets.*EMPTY*)));  
  
  
 StageStyle style= StageStyle.*UTILITY*;  
 Modality modality= Modality.*APPLICATION\_MODAL*;  
  
 Stage popupwindow=new Stage();  
  
 String user;  
 String pass;  
 user=username.getText();  
 pass=password.getText();  
  
 boolean userauthentication;  
 SQLengine ref=new SQLengine();  
 userauthentication=ref.Authentication(user,pass); //epistrefei true ama yparxei o user  
  
 popupwindow.initStyle(style);  
 popupwindow.initModality(modality);  
 if(userauthentication){  
 popup.setFill(Color.*TRANSPARENT*);  
 h1.setStyle("-fx-background-color: rgba(0,0,0,0);"); //xrwmata me css  
 continuebtn.setOnAction(e->ref.itscreen());  
  
  
 popupwindow.setScene(popup);  
  
  
 }  
 else {  
 popup.setFill(Color.*TRANSPARENT*);  
 h1.setStyle("-fx-background-color: rgba(0,0,0,0);");  
  
 popupwindow.setScene(tempscene2);  
 }  
 popupwindow.show();  
 }  
  
  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

**ΚΩΔΙΚΑΣ ΓΙΑ ΤΗΝ SQLengine:**

import java.lang.\*;  
  
import javafx.scene.control.Button;  
import javafx.scene.text.Font;  
import javafx.scene.text.Text;  
import javafx.stage.\*;  
import javafx.scene.\*;  
import javafx.scene.layout.\*;  
import javafx.scene.control.\*;  
import javafx.geometry.\*;  
import javafx.scene.paint.\*;  
import java.sql.\*;  
import java.time.\*;  
import java.time.format.DateTimeFormatter;  
  
  
public class SQLengine {  
 private String tablechoose;  
 Stage mainscreen,insertstage,itWindow;  
 String username,password;  
 Label title,comboboxlbl;  
 Button buttoninsert,buttondel,buttonupdate,buttonselect,btnexitit,combobtn;  
 ComboBox<String> TableChoice;  
 Scene itscene,comboscene;  
 TextField txtValue;  
  
  
 //kanei authenticate ton xrhsth oti yparxei sto database kai einai sto itdepartment  
 public boolean Authentication(String username1,String password1){  
 String tempuser,temppass,workerselect,workerAM,fname,lname,fullusername;  
  
 try(Connection conn=DriverManager.*getConnection*(  
 "jdbc:mysql://localhost:3306/ceidprojectsql22",  
 username1,password1)){ //leitourgei  
 Statement stmt=conn.createStatement();  
 Statement stmt2=conn.createStatement();  
 String itcheckexists="SELECT it\_at,password FROM itdepartment"; //pairnei to AM kai to password apo to itdepartment  
  
 ResultSet itresult=stmt.executeQuery(itcheckexists);  
  
 if(conn!=null){System.*out*.println("LOGIN SUCCESSFUL");} //GIA TIN CONSOLA  
  
 while(itresult.next()){ //while gia tin periptwsh poy exoyme parapanw apo enan sto itdepartment  
 temppass=itresult.getString("password");  
 workerAM=itresult.getString("it\_at"); //epistrefei ton worker me to AM apo to itresult set  
  
 String workername="SELECT wrk\_name,wrk\_lname FROM worker WHERE wrk\_AT='"+workerAM+"'";  
 ResultSet workerresult= stmt2.executeQuery(workername); //tha epistrepsei fname kai lname toy worker  
 workerresult.next(); //einai null stin arxi kai prepei na paei sto monadiko stoixeio poy epistrefetai  
 fname=workerresult.getString("wrk\_name");  
 lname=workerresult.getString("wrk\_lname");  
 fullusername=fname.concat(lname); //concat gia na parw to fullusername  
  
  
 if((fullusername.equals(username1))&&(password1.equals(temppass))){ //checkarei to workername kai to password toy it ama yparxei,kai an nai epistrefei true sto login  
 return true;  
 } //doulevei  
 }  
  
 }catch(Exception e){  
 System.*out*.println(e.getMessage());  
 e.printStackTrace();  
 }  
 System.*out*.println("User not found");  
 return false;  
 }  
  
  
 public void itscreen(){ //edw new window  
  
 Label usernamelbl;  
 itWindow=new Stage();  
 itWindow.setTitle("IT WINDOW");  
 BorderPane itroot=new BorderPane();  
  
 //left edw tha valw ta buttons  
 StackPane leftPane = new StackPane();  
 leftPane.setBackground(new Background(new BackgroundFill(Color.*DIMGRAY*,CornerRadii.*EMPTY*,Insets.*EMPTY*)));  
 btnexitit=new Button("Exit");  
 btnexitit.setAlignment(Pos.*TOP\_CENTER*);  
 btnexitit.setLayoutX(100);  
 btnexitit.setLayoutY(100);  
 btnexitit.setPrefSize(100,50);  
 btnexitit.setFont(Font.*font*("Verdana",25));  
 btnexitit.setOnAction(e->System.*exit*(0));  
 leftPane.getChildren().add(btnexitit);  
 itroot.setLeft(leftPane);  
  
 //top pane  
 VBox topPane=new VBox();  
 DateTimeFormatter dtf = DateTimeFormatter.*ofPattern*("yyyy/MM/dd HH:mm:ss");  
 LocalDateTime datenow =LocalDateTime.*now*();  
 topPane.setBackground(new Background(new BackgroundFill(Color.*ORANGE*,CornerRadii.*EMPTY*,Insets.*EMPTY*)));  
 title=new Label("USER:IT \t"+ dtf.format(datenow));  
 topPane.setAlignment(Pos.*TOP\_LEFT*);  
 topPane.getChildren().add(title);  
 itroot.setTop(topPane);  
  
  
 //right pane  
 VBox rightPane=new VBox();  
 rightPane.setBackground(new Background(new BackgroundFill(Color.*DIMGRAY*,CornerRadii.*EMPTY*,Insets.*EMPTY*)));  
 buttonselect=new Button("SELECT");  
 buttondel=new Button("DELETE");  
 buttoninsert=new Button("INSERT");  
 buttonupdate=new Button("UPDATE");  
  
 buttonselect.setFont(Font.*font*("Verdana",15));  
 buttondel.setFont(Font.*font*("Verdana",15));  
 buttoninsert.setFont(Font.*font*("Verdana",15));  
 buttonupdate.setFont(Font.*font*("Verdana",15));  
  
 buttonselect.setPrefSize(150,50);  
 buttonupdate.setPrefSize(150,50);  
 buttondel.setPrefSize(150,50);  
 buttoninsert.setPrefSize(150,50);  
  
 buttoninsert.setOnAction(e->InsertValues());  
 buttondel.setOnAction(e->DeleteValues());  
 buttonupdate.setOnAction(e->UpdateValues());  
 buttonselect.setOnAction(e->SelectValues());  
  
 VBox Rpane1=new VBox(buttonselect);  
 VBox Rpane2=new VBox(buttonupdate);  
 VBox Rpane3=new VBox(buttondel);  
 VBox Rpane4=new VBox(buttoninsert);  
  
  
 rightPane.getChildren().addAll(Rpane1,Rpane2,Rpane3,Rpane4);  
 rightPane.setAlignment(Pos.*CENTER*);  
 rightPane.setSpacing(100);  
 rightPane.setPadding(new Insets(50));  
 itroot.setRight(rightPane);  
  
 //center  
  
  
 itscene=new Scene(itroot,1200,900);  
 itWindow.setScene(itscene);  
 itWindow.show();  
 }  
  
 public void setValue(){  
 if(TableChoice.getItems().contains(txtValue.getText()))  
 TableChoice.setValue(txtValue.getText());  
 }  
  
 public void InsertValues(){  
 comboboxlbl=new Label("Insert into: ");  
  
 combobtn=new Button("OK");  
 insertstage=new Stage();  
  
  
 BorderPane root=new BorderPane();  
 TableChoice=new ComboBox<>();  
 TableChoice.getItems().addAll("admin","branch", "destination", "driver", "event", "guide", "itdepartment",  
 "languages", "logger", "manages", "offers", "phones", "reservation", "reservation\_offers","travel\_to","trip","worker");  
 TableChoice.setEditable(true);  
 HBox TopPane=new HBox(comboboxlbl,TableChoice,combobtn);  
 TopPane.setStyle("-fx-background-color: rgba(1,0,0,0);");  
 TopPane.setAlignment(Pos.*CENTER*);  
 TopPane.setBackground(new Background(new BackgroundFill(Color.*DIMGRAY*,CornerRadii.*EMPTY*,Insets.*EMPTY*)));  
 root.setTop(TopPane);  
  
  
 TableChoice.getEditor().textProperty().addListener((observable,oldText,newText) ->{  
 TableChoice.setValue(newText);  
 } );  
  
  
 comboscene=new Scene(root,1200,900);  
 VBox.*setMargin*(TableChoice,new Insets(20,0,0,0));  
 insertstage.setScene(comboscene);  
 insertstage.setResizable(false);  
 insertstage.show();  
 }  
  
 public void SelectValues(){  
  
 }  
  
 public void UpdateValues(){  
  
 }  
 public void DeleteValues(){  
  
 }  
 //telos  
}

import java.lang.\*;  
  
import javafx.scene.control.Button;  
import javafx.scene.text.Font;  
import javafx.scene.text.Text;  
import javafx.stage.\*;  
import javafx.scene.\*;  
import javafx.scene.layout.\*;  
import javafx.scene.control.\*;  
import javafx.geometry.\*;  
import javafx.scene.paint.\*;  
import java.sql.\*;  
import java.time.\*;  
import java.time.format.DateTimeFormatter;  
  
  
public class SQLengine {  
 private String tablechoose;  
 Stage mainscreen,insertstage,itWindow;  
 String username,password;  
 Label title,comboboxlbl;  
 Button buttoninsert,buttondel,buttonupdate,buttonselect,btnexitit,combobtn;  
 ComboBox<String> TableChoice;  
 Scene itscene,comboscene;  
 TextField txtValue;  
  
  
 //kanei authenticate ton xrhsth oti yparxei sto database kai einai sto itdepartment  
 public boolean Authentication(String username1,String password1){  
 String tempuser,temppass,workerselect,workerAM,fname,lname,fullusername;  
  
 try(Connection conn=DriverManager.*getConnection*(  
 "jdbc:mysql://localhost:3306/ceidprojectsql22",  
 username1,password1)){ //leitourgei  
 Statement stmt=conn.createStatement();  
 Statement stmt2=conn.createStatement();  
 String itcheckexists="SELECT it\_at,password FROM itdepartment"; //pairnei to AM kai to password apo to itdepartment  
  
 ResultSet itresult=stmt.executeQuery(itcheckexists);  
  
 if(conn!=null){System.*out*.println("LOGIN SUCCESSFUL");} //GIA TIN CONSOLA  
  
 while(itresult.next()){ //while gia tin periptwsh poy exoyme parapanw apo enan sto itdepartment  
 temppass=itresult.getString("password");  
 workerAM=itresult.getString("it\_at"); //epistrefei ton worker me to AM apo to itresult set  
  
 String workername="SELECT wrk\_name,wrk\_lname FROM worker WHERE wrk\_AT='"+workerAM+"'";  
 ResultSet workerresult= stmt2.executeQuery(workername); //tha epistrepsei fname kai lname toy worker  
 workerresult.next(); //einai null stin arxi kai prepei na paei sto monadiko stoixeio poy epistrefetai  
 fname=workerresult.getString("wrk\_name");  
 lname=workerresult.getString("wrk\_lname");  
 fullusername=fname.concat(lname); //concat gia na parw to fullusername  
  
  
 if((fullusername.equals(username1))&&(password1.equals(temppass))){ //checkarei to workername kai to password toy it ama yparxei,kai an nai epistrefei true sto login  
 return true;  
 } //doulevei  
 }  
  
 }catch(Exception e){  
 System.*out*.println(e.getMessage());  
 e.printStackTrace();  
 }  
 System.*out*.println("User not found");  
 return false;  
 }  
  
  
 public void itscreen(){ //edw new window  
  
 Label usernamelbl;  
 itWindow=new Stage();  
 itWindow.setTitle("IT WINDOW");  
 BorderPane itroot=new BorderPane();  
  
 //left edw tha valw ta buttons  
 StackPane leftPane = new StackPane();  
 leftPane.setBackground(new Background(new BackgroundFill(Color.*DIMGRAY*,CornerRadii.*EMPTY*,Insets.*EMPTY*)));  
 btnexitit=new Button("Exit");  
 btnexitit.setAlignment(Pos.*TOP\_CENTER*);  
 btnexitit.setLayoutX(100);  
 btnexitit.setLayoutY(100);  
 btnexitit.setPrefSize(100,50);  
 btnexitit.setFont(Font.*font*("Verdana",25));  
 btnexitit.setOnAction(e->System.*exit*(0));  
 leftPane.getChildren().add(btnexitit);  
 itroot.setLeft(leftPane);  
  
 //top pane  
 VBox topPane=new VBox();  
 DateTimeFormatter dtf = DateTimeFormatter.*ofPattern*("yyyy/MM/dd HH:mm:ss");  
 LocalDateTime datenow =LocalDateTime.*now*();  
 topPane.setBackground(new Background(new BackgroundFill(Color.*ORANGE*,CornerRadii.*EMPTY*,Insets.*EMPTY*)));  
 title=new Label("USER:IT \t"+ dtf.format(datenow));  
 topPane.setAlignment(Pos.*TOP\_LEFT*);  
 topPane.getChildren().add(title);  
 itroot.setTop(topPane);  
  
  
 //right pane  
 VBox rightPane=new VBox();  
 rightPane.setBackground(new Background(new BackgroundFill(Color.*DIMGRAY*,CornerRadii.*EMPTY*,Insets.*EMPTY*)));  
 buttonselect=new Button("SELECT");  
 buttondel=new Button("DELETE");  
 buttoninsert=new Button("INSERT");  
 buttonupdate=new Button("UPDATE");  
  
 buttonselect.setFont(Font.*font*("Verdana",15));  
 buttondel.setFont(Font.*font*("Verdana",15));  
 buttoninsert.setFont(Font.*font*("Verdana",15));  
 buttonupdate.setFont(Font.*font*("Verdana",15));  
  
 buttonselect.setPrefSize(150,50);  
 buttonupdate.setPrefSize(150,50);  
 buttondel.setPrefSize(150,50);  
 buttoninsert.setPrefSize(150,50);  
  
 buttoninsert.setOnAction(e->InsertValues());  
 buttondel.setOnAction(e->DeleteValues());  
 buttonupdate.setOnAction(e->UpdateValues());  
 buttonselect.setOnAction(e->SelectValues());  
  
 VBox Rpane1=new VBox(buttonselect);  
 VBox Rpane2=new VBox(buttonupdate);  
 VBox Rpane3=new VBox(buttondel);  
 VBox Rpane4=new VBox(buttoninsert);  
  
  
 rightPane.getChildren().addAll(Rpane1,Rpane2,Rpane3,Rpane4);  
 rightPane.setAlignment(Pos.*CENTER*);  
 rightPane.setSpacing(100);  
 rightPane.setPadding(new Insets(50));  
 itroot.setRight(rightPane);  
  
 //center  
  
  
 itscene=new Scene(itroot,1200,900);  
 itWindow.setScene(itscene);  
 itWindow.show();  
 }  
  
 public void setValue(){  
 if(TableChoice.getItems().contains(txtValue.getText()))  
 TableChoice.setValue(txtValue.getText());  
 }  
  
 public void InsertValues(){  
 comboboxlbl=new Label("Insert into: ");  
  
 combobtn=new Button("OK");  
 insertstage=new Stage();  
  
  
 BorderPane root=new BorderPane();  
 TableChoice=new ComboBox<>();  
 TableChoice.getItems().addAll("admin","branch", "destination", "driver", "event", "guide", "itdepartment",  
 "languages", "logger", "manages", "offers", "phones", "reservation", "reservation\_offers","travel\_to","trip","worker");  
 TableChoice.setEditable(true);  
 HBox TopPane=new HBox(comboboxlbl,TableChoice,combobtn);  
 TopPane.setStyle("-fx-background-color: rgba(1,0,0,0);");  
 TopPane.setAlignment(Pos.*CENTER*);  
 TopPane.setBackground(new Background(new BackgroundFill(Color.*DIMGRAY*,CornerRadii.*EMPTY*,Insets.*EMPTY*)));  
 root.setTop(TopPane);  
  
  
 TableChoice.getEditor().textProperty().addListener((observable,oldText,newText) ->{  
 TableChoice.setValue(newText);  
 } );  
  
  
 comboscene=new Scene(root,1200,900);  
 VBox.*setMargin*(TableChoice,new Insets(20,0,0,0));  
 insertstage.setScene(comboscene);  
 insertstage.setResizable(false);  
 insertstage.show();  
 }  
  
 public void SelectValues(){  
  
 }  
  
 public void UpdateValues(){  
  
 }  
 public void DeleteValues(){  
  
 }  
 //telos  
}

**(ΤΕΛΟΣ ΑΝΑΦΟΡΑΣ)**