

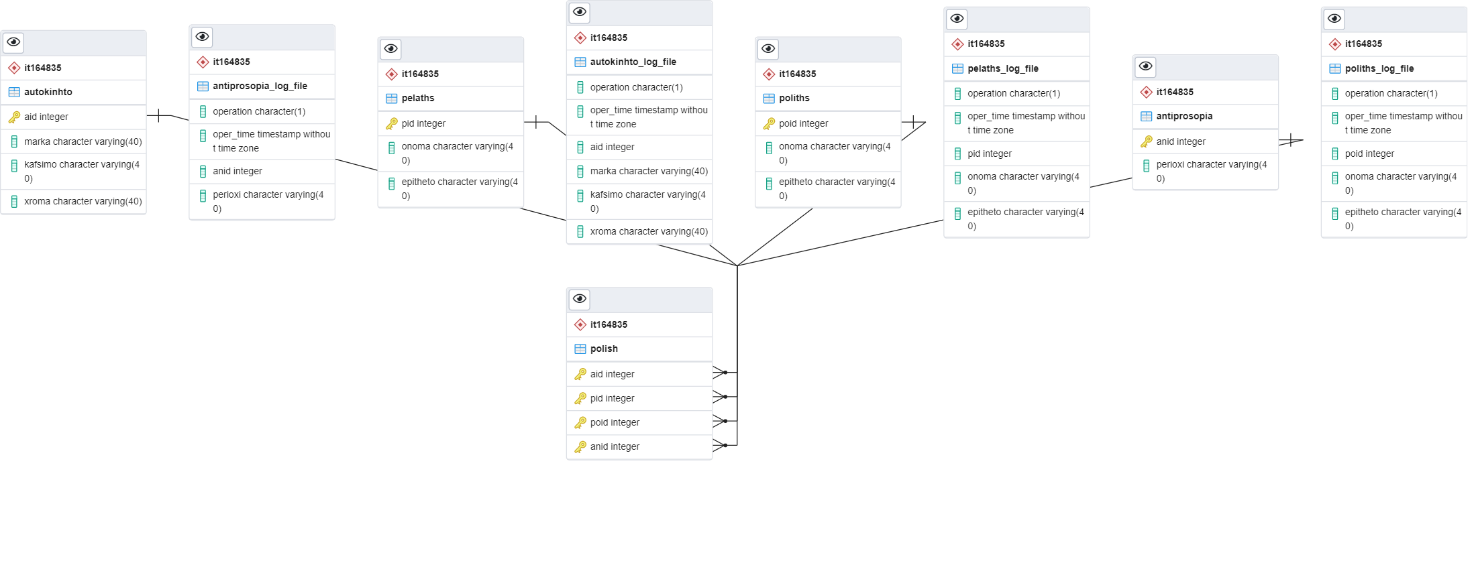
Τεχνολογία βάσεων Δεδομένων

Πληροφοριακό σύστημα μηχανογράφησης αντιπροσωπείας αυτοκινήτων

Χατζηστεφάνου Ευστάθιος 164835

Εργασία μαθήματος Τεχνολογία Βάσεων Δεδομένων με θέμα « Πληροφοριακό σύστημα μηχανογράφησης αντιπροσωπείας αυτοκινήτων».

# Διάγραμμα ER και κώδικας PostgreSQL.



Αρχικά κατασκευάστηκαν οι πίνακες από τους οποίους αποτελείται η βάση μας.

* 1ος πίνακας που κατασκευάστηκε ήταν ο πίνακας αντιπροσωπία.

create table antiprosopia

(anid integer not null constraint an\_key primary key,

perioxi varchar(40) not null);

Ο πίνακας αυτός αποτελείται από 2 μεταβλητές. Την anid (τύπου Integer) που είναι και το κύριο κλειδί του πίνακα και δηλώνει το id της αντιπροσωπίας και την perioxi (τύπου Varchar) όπου δηλώνει την περιοχή που εδρεύει η αντιπροσωπία.

* Ο 2ος πίνακας που κατασκευάστηκε ήταν ο πίνακας αυτοκίνητο.

create table autokinhto

(aid integer not null constraint c\_bid primary key,

marka varchar(40),

kafsimo varchar (40),

xroma varchar(40)

constraint c\_color check (xroma in ('Red','Blue','Light Green','Yellow','Black','Silver')));

Ο πίνακας αυτός αποτελείται από 4 μεταβλητές. Την aid (τύπου Integer) που είναι και το κύριο κλειδί του πίνακα και δηλώνει το id του αυτοκινήτου. Την marka (τύπου Varchar) όπου δηλώνει την μάρκα του αυτοκινήτου. Το kafsimo (τύπου Varchar) όπου δηλώνει το καύσιμο του αυτοκινήτου και το xroma (τύπου Varchar) όπου δηλώνει το χρώμα του αυτοκινήτου και μπορεί να πάρει μια από τις παρακάτω τιμές Red,Blue,Light Green,Yellow,Black,Silver.

* Ο 3ος πίνακας που κατασκευάστηκε ήταν ο πίνακας πελάτης.

create table pelaths

(pid integer not null constraint p\_key primary key,

onoma varchar(40) not null,

epitheto varchar(40) not null);

Ο πίνακας αυτός αποτελείται από 3 μεταβλητές. Την pid (τύπου Integer) που είναι και το κύριο κλειδί του πίνακα και δηλώνει το id του πελάτη. Την onoma (τύπου Varchar) όπου δηλώνει το όνομα του πελάτη. Την epitheto (τύπου Varchar) όπου δηλώνει το επίθετο του πελάτη. Οι μεταβλητές onoma και epitheto δεν μπορούν να είναι null.

* Ο 4ος πίνακας που κατασκευάστηκε ήταν ο πίνακας πωλητής.

create table poliths

(poid integer not null constraint po\_key primary key,

onoma varchar(40) not null,

epitheto varchar(40) not null);

Ο πίνακας αυτός αποτελείται από 3 μεταβλητές. Την poid (τύπου Integer) που είναι και το κύριο κλειδί του πίνακα και δηλώνει το id του πωλητή. Την onoma (τύπου Varchar) όπου δηλώνει το όνομα του πωλητή. Την epitheto (τύπου Varchar) όπου δηλώνει το επίθετο του πωλητή. Οι μεταβλητές onoma και epitheto δεν μπορούν να είναι null.

* Ο 5ος πίνακας που κατασκευάστηκε ήταν ο πίνακας πώληση.

create table polish

(aid integer not null constraint f\_key1 references autokinhto(aid),

pid integer not null constraint f\_key2 references pelaths(pid),

poid integer not null constraint f\_key3 references poliths(poid),

anid integer not null constraint f\_key4 references antiprosopia(anid),

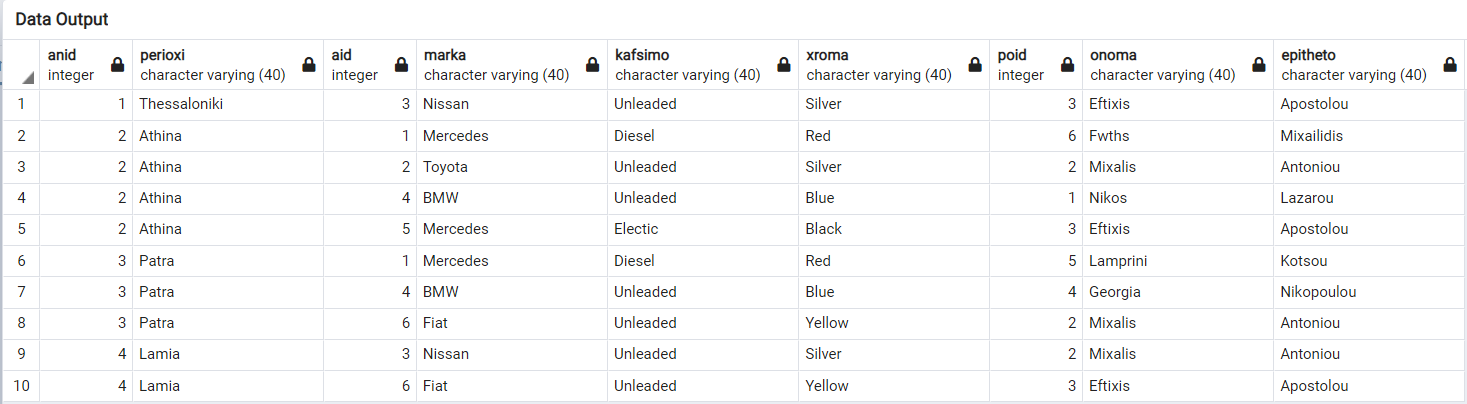
constraint prim\_key primary key(aid,pid,poid,anid));

Ο πίνακας αυτός αποτελείται από 4 μεταβλητές όσα και τα κύρια κλειδιά των παραπάνω πινάκων. Την aid (τύπου Integer) που δηλώνει το id του αυτοκινήτου. Την pid (τύπου Integer) που δηλώνει το id του πελάτη. Την poid (τύπου Integer) που δηλώνει το id του πωλητή. Την anid (τύπου Integer) που δηλώνει το id της αντιπροσωπίας. Τα παραπάνω κλειδιά δεν μπορούν να πάρουν την τιμή Null και ο συνδυασμός τους μας δίνει το κύριο κλειδί του πίνακα πώληση.

Στη συνέχεια κατασκευάστηκαν τα Functions που με τη βοήθεια τους εκτελούμε συγκεκριμένες διαδικασίες στην εφαρμογή μας.

Ένα από αυτά είναι το function epistrofh() το οποίο επιστρέφει ένα table view. Το view κατασκευάστηκε με join όλων των πινάκων από τους οποίους αποτελείται ο πίνακας polish και στη συνέχεια με select μόνο των στοιχείων που χρειαζόμασταν.

Το View:



Κατασκευή του Function:

create or replace FUNCTION epistrofh()

returns setof poliseis

AS

$func$

SELECT \*

FROM poliseis;

$func$

LANGUAGE sql;

Επόμενα Function υλοποιήθηκαν κατά τον ίδιο τρόπο για την επιστροφή των τιμών των πινάκων antiprosopia, autokinhto, pelaths, poliths. Στη συγκεκριμένη διαδικασία δεν χρειάστηκε η βοήθεια κάποιου View καθώς δεν χρειαζόταν κάποιο join παρά μόνο η εμφάνιση των στοιχείων του κάθε πίνακα ξεχωριστά.

* Για τον πίνακα antiprosopia:

create or replace FUNCTION epistrofhantiprosop()

returns setof antiprosopia

AS

$func$

SELECT \*

FROM antiprosopia;

$func$

LANGUAGE sql;

* Για τον πίνακα autokinhto:

create or replace FUNCTION epistrofhautokinhto()

returns setof autokinhto

AS

$func$

SELECT \*

FROM autokinhto;

$func$

LANGUAGE sql;

* Για τον πίνακα pelaths:

create or replace FUNCTION epistrofhpelaths()

returns setof pelaths

AS

$func$

SELECT \*

FROM pelaths;

$func$

LANGUAGE sql;

* Για τον πίνακα poliths:

create or replace FUNCTION epistrofhpoliths()

returns setof poliths

AS

$func$

SELECT \*

FROM poliths;

$func$

LANGUAGE sql;

Στη συνέχεια κατασκευάστηκαν τα Functions για εισαγωγή τιμών από τον user στους πίνακες antiprosopia, autokinhto, pelaths, poliths.

* Για τον πίνακα antiprosopia:

create or replace FUNCTION insertantiprosop(integer, varchar)

returns antiprosopia

AS

$func$

insert into antiprosopia (anid, perioxi) values ($1, $2)

returning antiprosopia

$func$

LANGUAGE SQL;

* Για τον πίνακα autokinhto:

create or replace FUNCTION insertauto(integer, varchar, varchar, varchar)

returns autokinhto

AS

$func$

insert into autokinhto (aid, marka, kafsimo, xroma) values ($1, $2, $3, $4)

returning autokinhto

$func$

LANGUAGE SQL;

* Για τον πίνακα pelaths:

create or replace FUNCTION insertpelaths(integer, varchar, varchar)

returns pelaths

AS

$func$

insert into pelaths (pid, onoma, epitheto) values ($1, $2, $3)

returning pelaths

$func$

LANGUAGE SQL;

* Για τον πίνακα poliths:

create or replace FUNCTION insertpoliths(integer, varchar, varchar)

returns poliths

AS

$func$

insert into poliths (poid, onoma, epitheto) values ($1, $2, $3)

returning poliths

$func$

LANGUAGE SQL;

Στη συνέχεια κατασκευάστηκαν τα Functions που δέχονται παραμέτρους για το update των πινάκων antiprosopia, autokinhto, pelaths, poliths.

* Για τον πίνακα antiprosopia:

create or replace FUNCTION updateantipro(integer, varchar)

returns antiprosopia

AS

$func$

Update antiprosopia set perioxi = $2

where anid = $1

RETURNING antiprosopia;

$func$

LANGUAGE SQL;

* Για τον πίνακα autokinhto:

create or replace FUNCTION updateauto(integer, varchar, varchar, varchar)

returns autokinhto

AS

$func$

Update autokinhto set marka = $2,

kafsimo = $3,

xroma = $4

where aid = $1

RETURNING autokinhto;

$func$

LANGUAGE SQL;

* Για τον πίνακα pelaths:

create or replace FUNCTION updatepelaths(integer, varchar, varchar)

returns pelaths

AS

$func$

Update pelaths set onoma = $2,

epitheto = $3

where pid = $1

RETURNING pelaths;

$func$

LANGUAGE SQL;

* Για τον πίνακα poliths:

create or replace FUNCTION updatepoliths(integer, varchar, varchar)

returns poliths

AS

$func$

Update poliths set onoma = $2,

epitheto = $3

where poid = $1

RETURNING poliths;

$func$

LANGUAGE SQL;

Στη συνέχεια κατασκευάστηκαν τα Functions που δέχονται παραμέτρους για τη διαγραφή των πινάκων antiprosopia, autokinhto, pelaths, poliths.

* Για τον πίνακα antiprosopia:

create or replace FUNCTION deleteantiprosop(integer)

returns antiprosopia

AS

$func$

delete from antiprosopia where anid = $1

returning antiprosopia

$func$

LANGUAGE SQL;

* Για τον πίνακα autokinhto:

create or replace FUNCTION deleteauto(integer)

returns autokinhto

AS

$func$

delete from autokinhto where aid = $1

returning autokinhto

$func$

LANGUAGE SQL;

* Για τον πίνακα pelaths:

create or replace FUNCTION deletepelaths(integer)

returns pelaths

AS

$func$

delete from pelaths where pid = $1

returning pelaths

$func$

LANGUAGE SQL;

* Για τον πίνακα poliths:

create or replace FUNCTION deletepoliths(integer)

returns poliths

AS

$func$

delete from poliths where poid = $1

returning poliths

$func$

LANGUAGE SQL;

Κατασκευή του Log file:

Κατασκευάστηκαν τα Log file antiprosopia\_log\_file, autokinhto\_log\_file, pelaths\_log\_file, poliths\_log\_file για την αποθήκευση της τροποποίησης των δεδομένων των αντίστοιχων πινάκων antiprosopia, autokinhto, pelaths, poliths. Οι πίνακες log file δέχονται τις παραμέτρους των αρχικών πινάκων καθώς και το operation που παίρνει τις τιμές I(insert ) D(delete) U(update) αλλά και το operation date που δέχεται σαν παράμετρο την ημερομηνία και ώρα τροποποίησης.

Στη συνέχεια κατασκευάζονται τα function που δίνουν τιμές στους log file πίνακες εάν έγινε κάτι από τα Insert, Delete, Update. Ο έλεγχος για το εάν υπήρξε Insert, Delete, Update γίνεται με την βοήθεια των Trigger.

* Για την antiprosopia:

create table antiprosopia\_log\_file

(operation char(1),

oper\_time timestamp NOT NULL,

anid integer not null,

perioxi varchar(40) not null);

create function createlogantip()

returns trigger as $$

BEGIN

IF (TG\_OP = 'DELETE') THEN

INSERT INTO antiprosopia\_log\_file SELECT 'D', now(), OLD.anid, OLD.perioxi;

RETURN OLD;

ELSIF (TG\_OP = 'UPDATE') THEN

INSERT INTO antiprosopia\_log\_file SELECT 'U', now(), NEW.anid, NEW.perioxi;

RETURN NEW;

ELSIF (TG\_OP = 'INSERT') THEN

INSERT INTO antiprosopia\_log\_file SELECT 'I', now(), NEW.anid, NEW.perioxi;

RETURN NEW;

END IF;

RETURN NULL;

END;

$$

LANGUAGE plpgsql;

create trigger keeplogantip

after insert or update or delete on antiprosopia

for each row execute procedure createlogantip();

* Για την autokinhto:

create table autokinhto\_log\_file

(operation char(1),

oper\_time timestamp NOT NULL,

aid integer not null,

marka varchar(40) not null,

kafsimo varchar(40) not null,

xroma varchar(40) not null);

create function createlogauto()

returns trigger as $$

BEGIN

IF (TG\_OP = 'DELETE') THEN

INSERT INTO autokinhto\_log\_file SELECT 'D', now(), OLD.aid, OLD.marka, OLD.kafsimo, OLD.xroma;

RETURN OLD;

ELSIF (TG\_OP = 'UPDATE') THEN

INSERT INTO autokinhto\_log\_file SELECT 'U', now(), NEW.aid, NEW.marka, NEW.kafsimo, NEW.xroma;

RETURN NEW;

ELSIF (TG\_OP = 'INSERT') THEN

INSERT INTO autokinhto\_log\_file SELECT 'I', now(), NEW.aid, NEW.marka, NEW.kafsimo, NEW.xroma;

RETURN NEW;

END IF;

RETURN NULL;

END;

$$

LANGUAGE plpgsql;

create trigger keeplogauto

after insert or update or delete on autokinhto

for each row execute procedure createlogauto();

Για την pelaths:

create table pelaths\_log\_file

(operation char(1),

oper\_time timestamp NOT NULL,

pid integer not null,

onoma varchar(40) not null,

epitheto varchar(40) not null);

create function createlogpelaths()

returns trigger as $$

BEGIN

IF (TG\_OP = 'DELETE') THEN

INSERT INTO pelaths\_log\_file SELECT 'D', now(), OLD.pid, OLD.onoma, OLD.epitheto;

RETURN OLD;

ELSIF (TG\_OP = 'UPDATE') THEN

INSERT INTO pelaths\_log\_file SELECT 'U', now(), NEW.pid, NEW.onoma, NEW.epitheto;

RETURN NEW;

ELSIF (TG\_OP = 'INSERT') THEN

INSERT INTO pelaths\_log\_file SELECT 'I', now(), NEW.pid, NEW.onoma, NEW.epitheto;

RETURN NEW;

END IF;

RETURN NULL;

END;

$$

LANGUAGE plpgsql;

create trigger keeplogpelaths

after insert or update or delete on pelaths

for each row execute procedure createlogpelaths();

Για την poliths:

create table poliths\_log\_file

(operation char(1),

oper\_time timestamp NOT NULL,

poid integer not null,

onoma varchar(40) not null,

epitheto varchar(40) not null);

create function createlogpoliths()

returns trigger as $$

BEGIN

IF (TG\_OP = 'DELETE') THEN

INSERT INTO poliths\_log\_file SELECT 'D', now(), OLD.poid, OLD.onoma, OLD.epitheto;

RETURN OLD;

ELSIF (TG\_OP = 'UPDATE') THEN

INSERT INTO poliths\_log\_file SELECT 'U', now(), NEW.poid, NEW.onoma, NEW.epitheto;

RETURN NEW;

ELSIF (TG\_OP = 'INSERT') THEN

INSERT INTO poliths\_log\_file SELECT 'I', now(), NEW.poid, NEW.onoma, NEW.epitheto;

RETURN NEW;

END IF;

RETURN NULL;

END;

$$

LANGUAGE plpgsql;

create trigger keeplogpoliths

after insert or update or delete on poliths

for each row execute procedure createlogpoliths();

Τέλος κατασκευάζουμε 4 function για την εμφάνιση των πινάκων antiprosopia\_log\_file, autokinhto\_log\_file, pelaths\_log\_file, poliths\_log\_file. Τα Function αυτά είναι:

Για antiprosopia\_log\_file:

create or replace FUNCTION epistofilogantip()

returns setof antiprosopia\_log\_file

AS

$func$

SELECT \*

FROM antiprosopia\_log\_file;

$func$

LANGUAGE sql;

Για autokinhto\_log\_file:

create or replace FUNCTION epistofilogauto()

returns setof autokinhto\_log\_file

AS

$func$

SELECT \*

FROM autokinhto\_log\_file;

$func$

LANGUAGE sql;

Για pelaths\_log\_file:

create or replace FUNCTION epistofilogpelaths()

returns setof pelaths\_log\_file

AS

$func$

SELECT \*

FROM pelaths\_log\_file;

$func$

LANGUAGE sql;

Για poliths\_log\_file:

create or replace FUNCTION epistofilogpoliths()

returns setof poliths\_log\_file

AS

$func$

SELECT \*

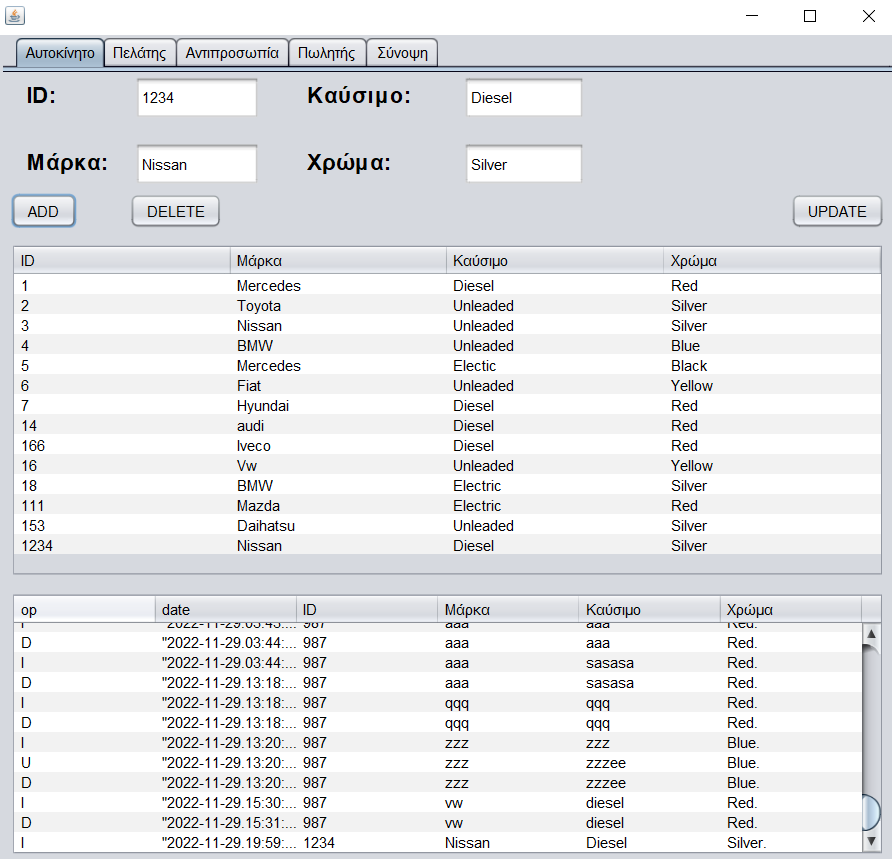
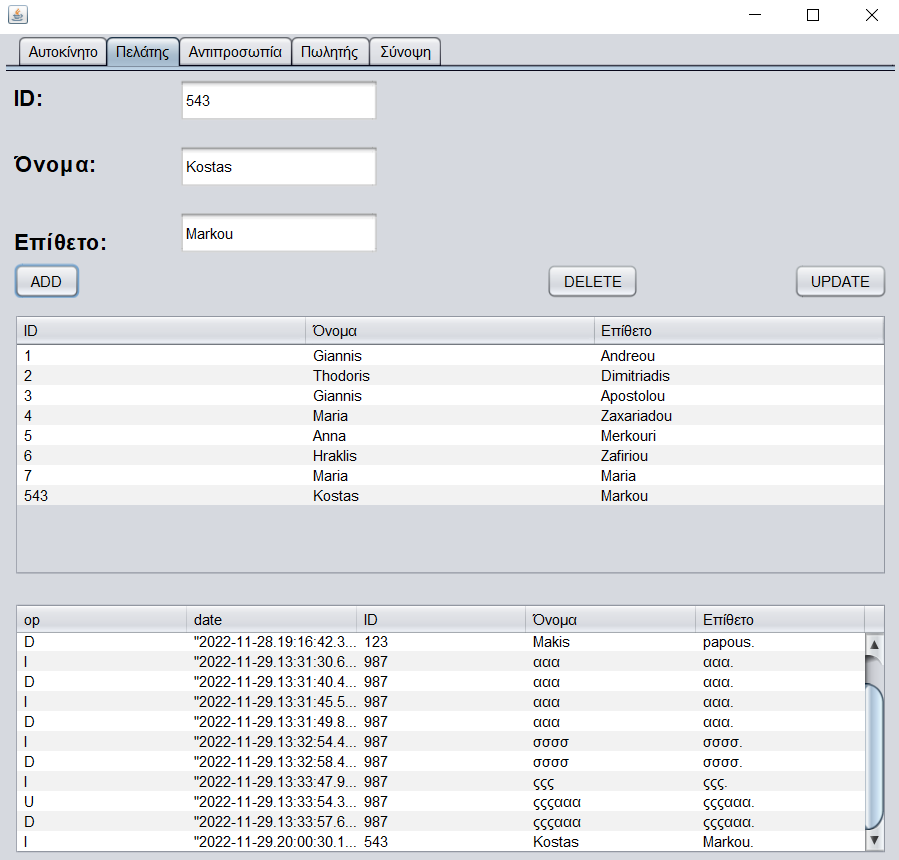
FROM poliths\_log\_file;

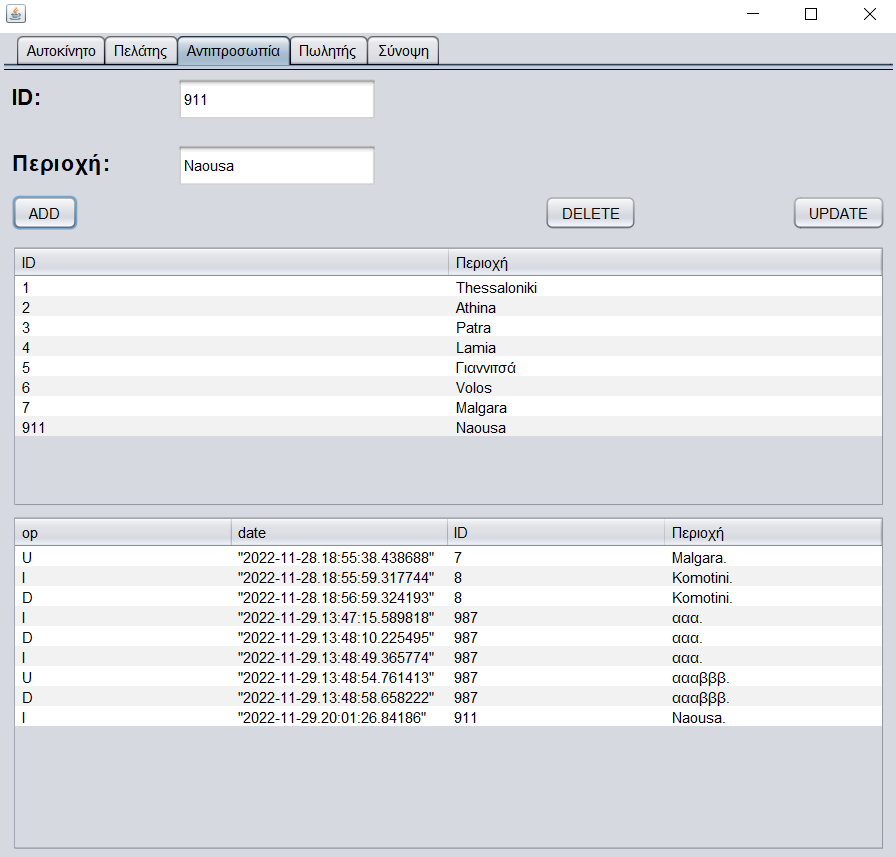
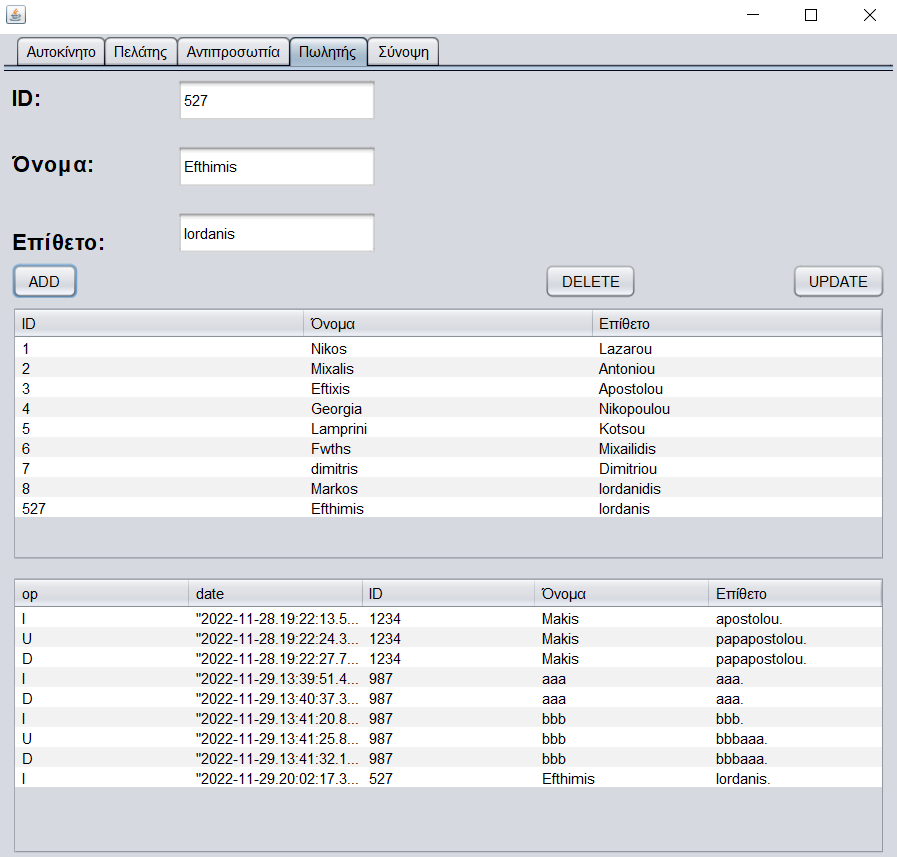
$func$

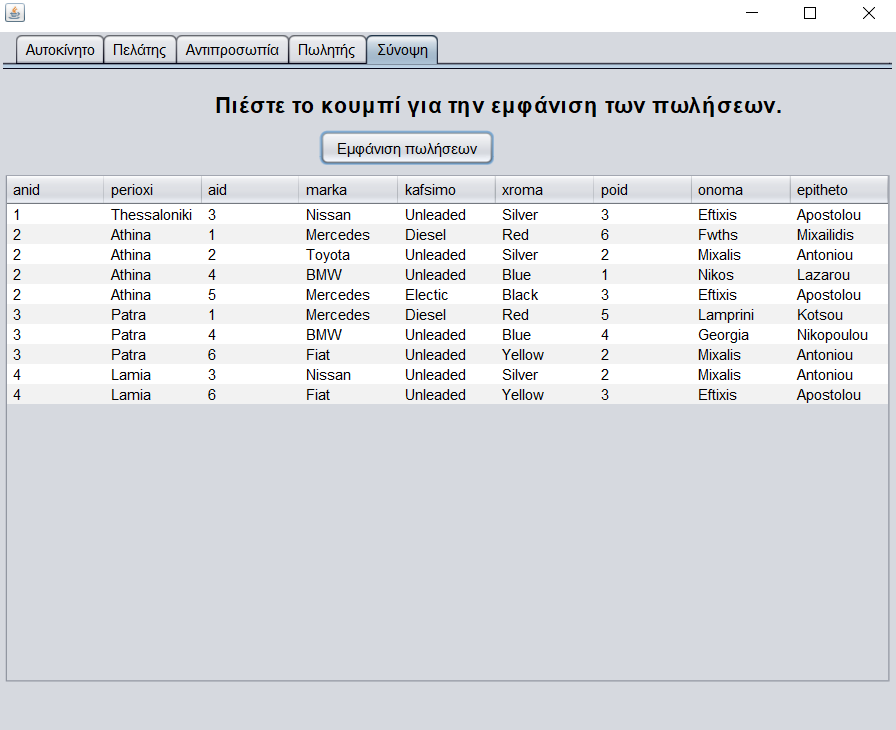
LANGUAGE sql;

# Υλοποίηση GUI με χρήση Java Swing και κώδικας.

Το GUI υλοποιήθηκε μέσω του προγράμματος NetBeans. Αρχικά δημιουργήθηκε ένα JFrame στο οποίο εγκαταστάθηκαν 5 Tabbed Pane. Κάθε ένα από αυτά αντιπροσωπεύει και ένα πεδίο της εφαρμογής. Το 1ο πεδίο είναι το αυτοκίνητο, εκεί θα βρούμε 4 Text Field που μπορούμε συμπληρώνοντας τα να επέμβουμε στα στοιχεία του πίνακα autokinhto κάνοντας οποιαδήποτε από τις διαδικασίες ADD, DELETE , UPDATE και να εμφανίσουμε τα αποτελέσματα με τα συνολικά στοιχεία του πίνακα αλλά και του αντίστοιχου Log File. Αντίστοιχα λειτουργούν και τα άλλα 4 πεδία. Το 2ο πεδίο αναφέρετε στους πελάτες. Το 3ο στην αντιπροσωπία , το 4ο στους πωλητές κα τέλος το 5ο στην σύνοψη των πωλήσεων – πώληση.



Κώδικας για την δημιουργία σύνδεσης στη βάση:

static String driverClassName = "org.postgresql.Driver" ;

static String url = "jdbc:postgresql://dblabs.it.teithe.gr:5432/it164835" ;

static Connection dbConnection = null;

static String username = "it164835";

static String passwd = "stathis1997";

static Statement statement = null;

static ResultSet rs = null;

static PreparedStatement insertauto = null;

public NewJFrame() {

initComponents();

Κώδικας για το πεδίο αυτοκίνητο εάν επιλέξουμε το Button Add:

if(jID.getText().equals("") || jMarka.getText().equals("")|| jKafsimo.getText().equals("")|| jXroma.getText().equals("")){

JOptionPane.showMessageDialog(this, "Βάλτε όλα τα δεδομένα.");

}else {

int vid = Integer.valueOf(jID.getText());

String vmarka = jMarka.getText();

String vkafsimo = jKafsimo.getText();

String vxroma = jXroma.getText();

System.out.println(vmarka);

try {

Class.forName (driverClassName);

} catch (ClassNotFoundException ex) {

}

try {

dbConnection = DriverManager.getConnection (url, username, passwd);

String insertString = "select insertauto(?,?,?,?)";

insertauto = dbConnection.prepareStatement(insertString);

} catch(SQLException ex) {

System.out.println("\n -- SQL Exception --- \n");

while(ex != null) {

System.out.println("Message: " + ex.getMessage());

ex = ex.getNextException();

try {

insertauto.close();

} catch (SQLException ex1) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex1);

}

try {

dbConnection.close();

} catch (SQLException ex1) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex1);

}

}

}

try {

insertauto.setInt(1, vid);

insertauto.setString(2, vmarka);

insertauto.setString(3, vkafsimo);

insertauto.setString(4, vxroma);

insertauto.executeUpdate();

} catch(SQLException ex) {

System.out.println("\n -- SQL Exception --- \n");

while(ex != null) {

System.out.println("Message: " + ex.getMessage());

ex = ex.getNextException();

}

}

try {

Class.forName (driverClassName);

try {

dbConnection = DriverManager.getConnection (url, username, passwd);

} catch (SQLException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

}

statement = dbConnection.createStatement();

String selectString = "SELECT epistrofhautokinhto() as ANSWER";

rs = statement.executeQuery(selectString);

//System.out.println("anid, perioxi, aid, marka, kafsimo, xroma, poid, onoma, epitheto " );

int count=0;

ArrayList<String[]> pinakas = new ArrayList<>();

while (rs.next()){

String autokinhta = rs.getString("ANSWER");

count++;

autokinhta = autokinhta.replaceAll("\\)"," ").replaceAll("\\(","");

String[] ss = autokinhta.split(" ");

for (String string : ss) {

pinakas.add(string.split(","));

}

}

Object[][] row = new Object[count][4];

int index = 0;

for(String[] pwlhsh : pinakas){

row[index][0]= pwlhsh[0];

row[index][1]= pwlhsh[1];

row[index][2]= pwlhsh[2];

row[index][3]= pwlhsh[3];

index++;

}

String[] cols = {"ID", "Μάρκα", "Καύσιμο", "Χρώμα"};

DefaultTableModel model = new DefaultTableModel(row,cols);

jTableauto.setModel(model);

statement.close();

dbConnection.close();

} catch (SQLException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

} catch (ClassNotFoundException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

}

try {

Class.forName (driverClassName);

try {

dbConnection = DriverManager.getConnection (url, username, passwd);

} catch (SQLException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

}

statement = dbConnection.createStatement();

String selectString = "SELECT epistofilogauto() as ANSWER";

rs = statement.executeQuery(selectString);

int count=0;

ArrayList<String[]> pinakas = new ArrayList<>();

while (rs.next()){

String autokinhta = rs.getString("ANSWER");

count++;

autokinhta = autokinhta.replaceAll("\\)"," ").replaceAll("\\(","").replaceAll(" ", ".");

String[] ss = autokinhta.split(" ");

for (String string : ss) {

pinakas.add(string.split(","));

}

}

Object[][] row = new Object[count][6];

int index = 0;

for(String[] pwlhsh : pinakas){

row[index][0]= pwlhsh[0];

row[index][1]= pwlhsh[1];

row[index][2]= pwlhsh[2];

row[index][3]= pwlhsh[3];

row[index][4]= pwlhsh[4];

row[index][5]= pwlhsh[5];

index++;

}

String[] cols = {"op", "date", "ID", "Μάρκα", "Καύσιμο", "Χρώμα"};

DefaultTableModel model = new DefaultTableModel(row,cols);

jTlogfileauto.setModel(model);

statement.close();

dbConnection.close();

} catch (SQLException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

} catch (ClassNotFoundException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

}

}

Κώδικας για το πεδίο αυτοκίνητο εάν επιλέξουμε το Button delete:

if(jID.getText().equals("")){

JOptionPane.showMessageDialog(this, "Δώσε ID για διαγραφή.");

}else {

int vid = Integer.valueOf(jID.getText());

try {

Class.forName (driverClassName);

} catch (ClassNotFoundException ex) {

}

try {

dbConnection = DriverManager.getConnection (url, username, passwd);

String insertString = "select deleteauto(?)";

insertauto = dbConnection.prepareStatement(insertString);

} catch(SQLException ex) {

System.out.println("\n -- SQL Exception --- \n");

while(ex != null) {

System.out.println("Message: " + ex.getMessage());

ex = ex.getNextException();

try {

insertauto.close();

} catch (SQLException ex1) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex1);

}

try {

dbConnection.close();

} catch (SQLException ex1) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex1);

}

}

}

try {

insertauto.setInt(1, vid);

//insertauto.setString(2, vmarka);

//insertauto.setString(3, vkafsimo);

//insertauto.setString(4, vxroma);

insertauto.executeUpdate();

} catch(SQLException ex) {

System.out.println("\n -- SQL Exception --- \n");

while(ex != null) {

System.out.println("Message: " + ex.getMessage());

ex = ex.getNextException();

}

}

try {

Class.forName (driverClassName);

try {

dbConnection = DriverManager.getConnection (url, username, passwd);

} catch (SQLException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

}

statement = dbConnection.createStatement();

String selectString = "SELECT epistrofhautokinhto() as ANSWER";

rs = statement.executeQuery(selectString);

//System.out.println("anid, perioxi, aid, marka, kafsimo, xroma, poid, onoma, epitheto " );

int count=0;

ArrayList<String[]> pinakas = new ArrayList<>();

while (rs.next()){

String autokinhta = rs.getString("ANSWER");

count++;

autokinhta = autokinhta.replaceAll("\\)"," ").replaceAll("\\(","");

String[] ss = autokinhta.split(" ");

for (String string : ss) {

pinakas.add(string.split(","));

}

}

Object[][] row = new Object[count][4];

int index = 0;

for(String[] pwlhsh : pinakas){

row[index][0]= pwlhsh[0];

row[index][1]= pwlhsh[1];

row[index][2]= pwlhsh[2];

row[index][3]= pwlhsh[3];

index++;

}

String[] cols = {"ID", "Μάρκα", "Καύσιμο", "Χρώμα"};

DefaultTableModel model = new DefaultTableModel(row,cols);

jTableauto.setModel(model);

statement.close();

dbConnection.close();

} catch (SQLException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

} catch (ClassNotFoundException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

}

}

try {

Class.forName (driverClassName);

try {

dbConnection = DriverManager.getConnection (url, username, passwd);

} catch (SQLException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

}

statement = dbConnection.createStatement();

String selectString = "SELECT epistofilogauto() as ANSWER";

rs = statement.executeQuery(selectString);

int count=0;

ArrayList<String[]> pinakas = new ArrayList<>();

while (rs.next()){

String autokinhta = rs.getString("ANSWER");

count++;

autokinhta = autokinhta.replaceAll("\\)"," ").replaceAll("\\(","").replaceAll(" ", ".");

String[] ss = autokinhta.split(" ");

for (String string : ss) {

pinakas.add(string.split(","));

}

}

Object[][] row = new Object[count][6];

int index = 0;

for(String[] pwlhsh : pinakas){

row[index][0]= pwlhsh[0];

row[index][1]= pwlhsh[1];

row[index][2]= pwlhsh[2];

row[index][3]= pwlhsh[3];

row[index][4]= pwlhsh[4];

row[index][5]= pwlhsh[5];

index++;

}

String[] cols = {"op", "date", "ID", "Μάρκα", "Καύσιμο", "Χρώμα"};

DefaultTableModel model = new DefaultTableModel(row,cols);

jTlogfileauto.setModel(model);

statement.close();

dbConnection.close();

} catch (SQLException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

} catch (ClassNotFoundException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

}

Κώδικας για το πεδίο αυτοκίνητο εάν επιλέξουμε το Button Update:

if(jID.getText().equals("") || jMarka.getText().equals("")|| jKafsimo.getText().equals("")|| jXroma.getText().equals("")){

JOptionPane.showMessageDialog(this, "Βάλτε όλα τα δεδομένα.");

}else {

int vid = Integer.valueOf(jID.getText());

String vmarka = jMarka.getText();

String vkafsimo = jKafsimo.getText();

String vxroma = jXroma.getText();

System.out.println(vmarka);

try {

Class.forName (driverClassName);

} catch (ClassNotFoundException ex) {

}

try {

dbConnection = DriverManager.getConnection (url, username, passwd);

String insertString = "select updateauto(?,?,?,?)";

insertauto = dbConnection.prepareStatement(insertString);

} catch(SQLException ex) {

System.out.println("\n -- SQL Exception --- \n");

while(ex != null) {

System.out.println("Message: " + ex.getMessage());

ex = ex.getNextException();

try {

insertauto.close();

} catch (SQLException ex1) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex1);

}

try {

dbConnection.close();

} catch (SQLException ex1) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex1);

}

}

}

try {

insertauto.setInt(1, vid);

insertauto.setString(2, vmarka);

insertauto.setString(3, vkafsimo);

insertauto.setString(4, vxroma);

insertauto.executeUpdate();

} catch(SQLException ex) {

System.out.println("\n -- SQL Exception --- \n");

while(ex != null) {

System.out.println("Message: " + ex.getMessage());

ex = ex.getNextException();

}

}

try {

Class.forName (driverClassName);

try {

dbConnection = DriverManager.getConnection (url, username, passwd);

} catch (SQLException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

}

statement = dbConnection.createStatement();

String selectString = "SELECT epistrofhautokinhto() as ANSWER";

rs = statement.executeQuery(selectString);

//System.out.println("anid, perioxi, aid, marka, kafsimo, xroma, poid, onoma, epitheto " );

int count=0;

ArrayList<String[]> pinakas = new ArrayList<>();

while (rs.next()){

String autokinhta = rs.getString("ANSWER");

count++;

autokinhta = autokinhta.replaceAll("\\)"," ").replaceAll("\\(","");

String[] ss = autokinhta.split(" ");

for (String string : ss) {

pinakas.add(string.split(","));

}

}

Object[][] row = new Object[count][4];

int index = 0;

for(String[] pwlhsh : pinakas){

row[index][0]= pwlhsh[0];

row[index][1]= pwlhsh[1];

row[index][2]= pwlhsh[2];

row[index][3]= pwlhsh[3];

index++;

}

String[] cols = {"ID", "Μάρκα", "Καύσιμο", "Χρώμα"};

DefaultTableModel model = new DefaultTableModel(row,cols);

jTableauto.setModel(model);

statement.close();

dbConnection.close();

} catch (SQLException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

} catch (ClassNotFoundException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

}

}

try {

Class.forName (driverClassName);

try {

dbConnection = DriverManager.getConnection (url, username, passwd);

} catch (SQLException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

}

statement = dbConnection.createStatement();

String selectString = "SELECT epistofilogauto() as ANSWER";

rs = statement.executeQuery(selectString);

int count=0;

ArrayList<String[]> pinakas = new ArrayList<>();

while (rs.next()){

String autokinhta = rs.getString("ANSWER");

count++;

autokinhta = autokinhta.replaceAll("\\)"," ").replaceAll("\\(","").replaceAll(" ", ".");

String[] ss = autokinhta.split(" ");

for (String string : ss) {

pinakas.add(string.split(","));

}

}

Object[][] row = new Object[count][6];

int index = 0;

for(String[] pwlhsh : pinakas){

row[index][0]= pwlhsh[0];

row[index][1]= pwlhsh[1];

row[index][2]= pwlhsh[2];

row[index][3]= pwlhsh[3];

row[index][4]= pwlhsh[4];

row[index][5]= pwlhsh[5];

index++;

}

String[] cols = {"op", "date", "ID", "Μάρκα", "Καύσιμο", "Χρώμα"};

DefaultTableModel model = new DefaultTableModel(row,cols);

jTlogfileauto.setModel(model);

statement.close();

dbConnection.close();

} catch (SQLException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

} catch (ClassNotFoundException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);

}

Κατά τον ίδιο τρόπο εργαζόμαστε και στα πεδία Πελάτης, Αντιπροσωπία, Πωλητής καλώντας φυσικά τις αντίστοιχες Function που αναλύσαμε στις παραπάνω σελίδες ώστε να μας επιστρέψουν τα κατάλληλα αποτελέσματα στους gui πίνακές μας αλλά και να κάνουν και τις ανάλογες μετατροπές στα δεδομένα της βάσης μας.