**ΣΥΣΤΗΜΑ Econometrica**

Σύντομη Περιγραφή

Αντικείμενο της τρίτης γραπτής εργασία είναι η δημιουργία της εφαρμογής Econometrica. Η εφαρμογή αυτή παρέχει την δυνατότητα στον χρήστη να επιλέγει μέσω μιας λίστας την χώρα της οποίας θέλει να επεξεργαστεί τα στοιχεία στην συνέχεια η εφαρμογή προβάλει δύο λίστες μία για το ΑΕΠ (Gross Domestic Product – GDP) και μία για την κατανάλωση καυσίμων (Oil Consumption) για την χώρα που έχει επιλέξει ο χρήστης. Επίσης η εφαρμογή παρέχει την δυνατότητα στον χρήστη της στατιστικής ανάλυσης σε μορφή διαγράμματος.

Η εφαρμογή Econometrica είναι ένα σύστημα – πελάτης (client) της υπηρεσίας ιστού (web service) quandl.com, η οποία είναι μία βάση δεδομένων που περιέχει οικονομικά στοιχεία (π.χ. τιμές μετοχών, ΑΕΠ, κατανάλωση καυσίμων) για όλες τις χώρες του κόσμου. Η άντληση των πληροφοριών από τη βάση αυτή γίνεται από την εφαρμογή Econometrica μέσω ενός ειδικού API (Application Programming Interface). Προκειμένου να γίνουν οι κλήσεις API, για να γίνει η σύνδεση με την βάση και να λάβει τις πληροφορίες από το quandl.com χρησιμοποιήθηκε ένα API KEY για να διασφαλιστεί ότι η σύνδεση με την βάση γίνεται μόνο από τους εγγεγραμμένους χρήστες του συστήματος quandl.com.

Στόχος του έργου

Βασικός στόχος του project είναι η δημιουργία μιας εφαρμογής της οποίας το κύριο χαρακτηριστικό γνώρισμα της θα είναι η ευχρηστία. Μέσω μιας διεπαφής χρήστη (Graphical User Interface- GUI), η οποία τόσο με τον σχεδιασμό της όσο και με την απλότητα της θα παρέχει με εύχρηστο και κατανοητό τρόπο την δυνατότητα στον τελικό χρήστη (end user) την εύκολη και γρήγορη πλοήγηση στην εφαρμογή. Επίσης μέσω κατάλληλων μηνυμάτων σε περίπτωση λανθασμένων ενεργειών ο χρήστης θα μπορεί να αντιλαμβάνεται άμεσα το σφάλμα που έκανε.

Χαρακτηριστικά και Λειτουργίες της Εφαρμογής Econometrica

Τα χαρακτηριστικά και οι λειτουργίες της εφαρμογής Econometrica είναι όπως παρακάτω:

* Εκτελούνται κλήσεις λειτουργιών του API και οι απαντήσεις έρχονται σε μορφή JSON.
* Το σύστημα περιλαμβάνει μενού με τις παρακάτω επιλογές:
  1. Επιλογή χώρας μέσω ενός drop down μενού
  2. Ανάκτηση λίστας με δεδομένα ΑΕΠ και δεδομένα κατανάλωσης καυσίμων.
  3. Αποθήκευση δεδομένων στην βάση.
  4. Διαγραφεί δεδομένων από την βάση
  5. Στατιστικά (Διάγραμμα σύγκρισης)

**παραδοχές**

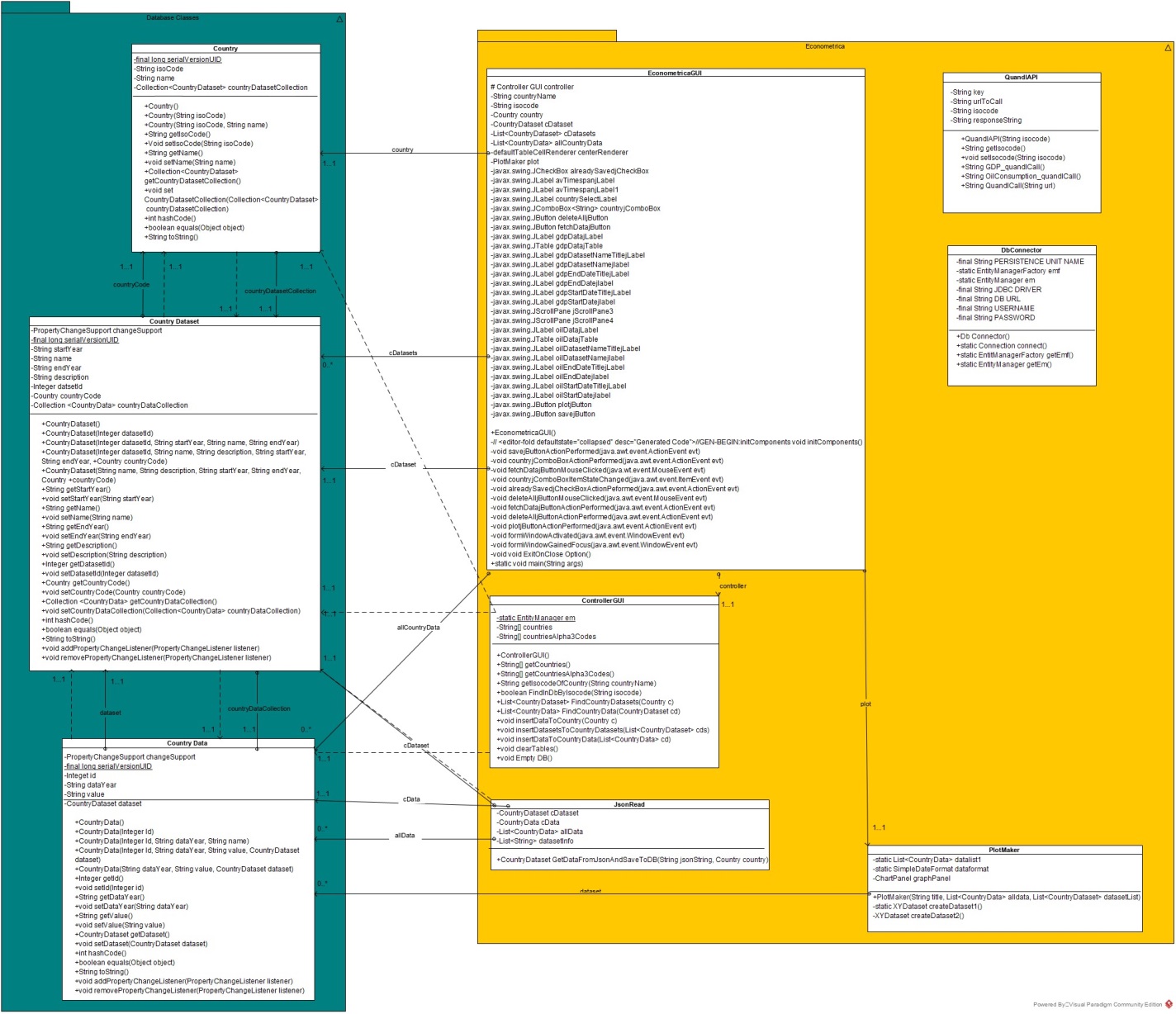
* Για τη λειτουργία της εφαρμογής ο χρήστης θα πρέπει να έχει πρόσβαση στο Internet.
* Ο χρήστης θα πρέπει την πρώτη φορά που θα εκκινήσει την εφαρμογή να κάνει εγκατάσταση της Βάσης Δεδομένων στον υπολογιστή του, από το αρχείο SQL που περιλαμβάνεται στο φάκελο της εργασίας.
  + Database name: Econometrica\_Data\_DB
  + username: ge3\_cpk
  + password: 1234
* Ο χρήστης θα πρέπει την πρώτη φορά που θα εκκινήσει την εφαρμογή να αποθηκεύσει στο Netbeans τις βιβλιοθήκες της Derby, της Gson, της jfreechart, jcommon, kotlin-stdlib, okio, okhttp και java db driver.

# ΥΛΟΠΟΙΗΣΗ ΕΡΓΟΥ



## Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java

**Α)**

****

**Β)**

Στην συνέχεια ακολουθεί ο κώδικας java για τις κλάσεις του συστήματος. Υπάρχουν εκτεταμένα σχόλια εσωτερικά του κώδικα. Παρακάτω θα υπάρξει μια μικρή περιγραφή για κάθε κλάση και ο κώδικας της κλάσης.

* Η κλάση QuandlAPI περιέχει τις μεθόδους για τις κλείσεις του API. Επίσης δημιουργεί τα endpoints. Στην κλάση αυτή είναι καταχωρημένο το API key.

QuandlAPI

package econometrica**;**

**import** java**.**io**.**IOException**;**

**import** javax**.**swing**.**JOptionPane**;**

**import** okhttp3**.**OkHttpClient**;**

**import** okhttp3**.**Request**;**

**import** okhttp3**.**Response**;**

public class QuandlAPI **{**

private final String key **=** "xyRco6LY7qzLkh5M7\_cy"**;**

private String urlToCall**;**

private String isocode**;**

private String responseString**;**

public String getIsocode**()** **{**

**return** isocode**;**

**}**

public void setIsocode**(**String isocode**)** **{**

**this.**isocode **=** isocode**;**

**}**

public QuandlAPI**(**String isocode**)** **{**

**this.**isocode **=** isocode**;**

**}**

public String GDP\_quandlCall**(){**

urlToCall **=** "https://www.quandl.com/api/v3/datasets/WWDI/" **+** isocode **+**

"\_NY\_GDP\_MKTP\_CN.json?api\_key=" **+** key**;**

**return** QuandlCall**(**urlToCall**);**

**}**

public String OilConsumption\_quandlCall**(){**

urlToCall **=** "https://www.quandl.com/api/v3/datasets/BP/OIL\_CONSUM\_" **+**

isocode **+** ".json?api\_key=" **+** key**;**

**return** QuandlCall**(**urlToCall**);**

**}**

public String QuandlCall**(**String url**){**

responseString **=** **null;**

OkHttpClient client **=** **new** OkHttpClient**();**

Request request **=** **new** Request**.**Builder**().**url**(**url**).**build**();**

**try** **(**Response response **=** client**.**newCall**(**request**).**execute**())** **{**

**if** **(**response**.**isSuccessful**()** **&&** response**.**body**()** **!=** **null)** **{**

responseString **=** response**.**body**().**string**();**

**}**

**}** **catch** **(**IOException e**)** **{**

JOptionPane**.**showMessageDialog**(null,** "Δεν είναι δυνατή η λήψη δεδομένων! Ελέγξτε τη σύνδεση με το internet."**);**

**}**

**return** responseString**;**

**}**

**}**

DbConnector

Η κλάση DbConnector , δημιουργεί την σύνδεση με την βάση δεδομένων. Επίσης δημιουργεί τον entity manager.

package econometrica**;**

**import** java**.**sql**.**Connection**;**

**import** java**.**sql**.**DriverManager**;**

**import** java**.**sql**.**SQLException**;**

**import** java**.**util**.**logging**.**Level**;**

**import** java**.**util**.**logging**.**Logger**;**

**import** javax**.**persistence**.**EntityManager**;**

**import** javax**.**persistence**.**EntityManagerFactory**;**

**import** javax**.**persistence**.**Persistence**;**

**import** javax**.**persistence**.**PersistenceContext**;**

**import** javax**.**swing**.**JOptionPane**;**

public final class DbConnector **{**

private static final String PERSISTENCE\_UNIT\_NAME **=** "EconometricaPU"**;**

private static EntityManagerFactory emf **=** **null;**

@PersistenceContext

private static EntityManager em **=** **null;**

//DRIVER Ο ΟΠΟΙΟΣ ΘΑ ΧΡΗΣΙΜΟΠΟΙΗΘΕΙ ΓΙΑ ΟΛΕΣ ΤΙΣ ΔΙΑΔΙΚΑΣΙΕΣ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

private static final String JDBC\_DRIVER **=** "com.mysql.jdbc.Driver"**;**

private static final String DB\_URL **=** "jdbc:derby://localhost:1527/Econometrica\_Data\_DB"**;**

//ΔΙΑΠΙΣΤΕΥΤΗΡΙΑ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

private static final String USERNAME **=** "ge3\_cpk"**;**

private static final String PASSWORD **=** "1234"**;**

public DbConnector**()** **{**

**try** **{**

//Register JDBC driver

Class**.**forName**(**"org.apache.derby.jdbc.ClientDriver"**);**

**}** **catch** **(**ClassNotFoundException ex**)** **{**

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

System**.**out**.**println**(**ex**);**

**}**

**}**

//ΔΑΙΔΙΚΑΣΙΑ ΣΥΝΔΕΣΗΣ ΜΕ ΤΗΝ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

public static Connection connect**(){**

Connection conn **=** **null;**

**try** **{**

//ΔΗΜΙΟΥΡΓΙΑ ΣΥΝΔΕΣΗΣ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ ΜΕΣΩ ΤΟΥ DRIVER ΚΑΙ ΤΩΝ ΔΙΑΠΙΣΤΕΥΤΗΡΙΩΝ

System**.**out**.**println**(**"Πραγματοποιείται σύνδεση στη βάση δεδομένων..."**);**

//ΔΗΜΙΟΥΡΓΙΑ ΤΟΥ Entity Manager ΓΙΑ ΤΗ ΣΥΝΔΕΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΜΕ ΤΗΝ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ.

emf **=** Persistence**.**createEntityManagerFactory**(**PERSISTENCE\_UNIT\_NAME**);**

em **=** emf**.**createEntityManager**();**

//ΚΑΤΑΧΩΡΗΣΗ ΣΤΗ ΜΕΤΑΒΛΗΤΗ conn ΤΥΠΟΥ Connection

conn **=** DriverManager**.**getConnection**(**DB\_URL**,** USERNAME**,** PASSWORD**);**

System**.**out**.**println**(**"----------------------------------------------------------------------------"**);**

System**.**out**.**println**(**"Συνδέθηκε στη βάση δεδομένων."**);**

**}catch(**SQLException ex**){**

System**.**out**.**println**(**ex**);**

//ΜΗΝΥΜΑ ΣΦΑΛΜΑΤΟΣ ΣΥΝΔΕΣΗΣ ΜΕ ΤΗΝ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

System**.**out**.**println**(**"----------------------------------------------------------------------------"**);**

JOptionPane**.**showMessageDialog**(null,** "Αποτυχία σύνδεσης με τη Βάση Δεδομένων. "**,**

"Σφάλμα"**,** JOptionPane**.**ERROR\_MESSAGE**);**

**}**

**return** conn**;**

**}**

public static EntityManagerFactory getEmf**()** **{**

**return** emf**;**

**}**

public static EntityManager getEm**()** **{**

**return** em**;**

**}**

**}**

ControllerGUI

Η κλάση ControllerGUI περιέχει μεθόδους για την εκτέλεση των queries για την ανάκτηση δεδομένων από την βάση, την αποθήκευση δεδομένων στην βάση, την διαγραφή ολόκληρης της βάσης. Επίσης μέσα στην κλάση γίνονται οι απαραίτητοι έλεγχοι για το αν η βάση είναι άδεια. Στον constructor της ControllerGUI υλοποιείται η σύνδεση μέσω της DbConnector, επίσης διαβάζει το αρχείο iso\_countries

package econometrica**;**

**import** databaseClasses**.\*;**

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**File**;**

**import** java**.**io**.**FileReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**util**.**List**;**

**import** javax**.**persistence**.**EntityManager**;**

**import** javax**.**persistence**.**Query**;**

**import** javax**.**swing**.**JOptionPane**;**

public class ControllerGUI **{**

private static EntityManager em**;**

private final String**[]** countries **=** **new** String**[**240**];**

private final String**[]** countriesAlpha3Codes **=** **new** String**[**240**];**

//CONSTRUCTOR

public ControllerGUI**(){**

//ΕΛΕΓΧΟΣ ΑΝ ΔΕΝ ΕΧΕΙ ΓΙΝΕΙ ΣΥΝΔΕΣΗ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

//ΚΑΙ ΚΛΗΣΗ ΤΗΣ ΚΛΑΣΗΣ ΣΥΝΔΕΣΗΣ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

**if** **(**em **==** **null){**

//ΚΛΗΣΗ FINAL CLASS DbConnector ΓΙΑ ΣΥΝΔΕΣΗ ΣΤΗ ΒΑΣΗ

//ΚΑΙ ΔΗΜΙΟΥΡΓΙΑ entity factory ΚΑΙ entity manager

DbConnector**.**connect**();**

em **=** DbConnector**.**getEm**();**

**}**

//ΔΙΑΒΑΣΜΑ ΔΙΑΘΕΣΙΜΩΝ ΧΩΡΩΝ ΚΑΙ alpha3 isocode ΑΠΟ ΑΡΧΕΙΟ .csv

**try** **{**

File file **=** **new** File**(**"iso-countries.csv"**);**

FileReader fileReader **=** **new** FileReader**(**file**);**

BufferedReader bufferedReader **=** **new** BufferedReader**(**fileReader**);**

String lineCountry**;**

String**[]** attributeCountry**;**

int i **=** 0**;**

**while** **((**lineCountry **=** bufferedReader**.**readLine**())** **!=** **null)** **{**

attributeCountry **=** lineCountry**.**split**(**";"**);**

//ΚΑΤΑΧΩΡΙΣΗ ΟΝΟΜΑΤΩΝ ΧΩΡΩΝ ΣΕ ΠΙΝΑΚΑ

**if(!**attributeCountry**[**0**].**equals**(**"Name"**)){**

countries**[**i**]** **=** attributeCountry**[**0**];**

**}else{**

countries**[**i**]** **=** "Choose a country"**;**

**}**

//ΚΑΤΑΧΩΡΙΣΗ alpha3 isocodes ΧΩΡΩΝ ΣΕ ΠΙΝΑΚΑ

countriesAlpha3Codes**[**i**]** **=** attributeCountry**[**2**];**

i**++;**

**}**

**}**

**catch** **(**IOException e**)** **{**

**}**

**}**

public String**[]** getCountries**()** **{**

**return** countries**;**

**}**

public String**[]** getCountriesAlpha3Codes**()** **{**

**return** countriesAlpha3Codes**;**

**}**

//ΜΕΘΟΔΟΣ ΕΥΡΕΣΗΣ isocode ΣΥΓΚΕΚΡΙΜΕΝΗΣ ΧΩΡΑΣ

public String getIsocodeOfCountry**(**String countryName**){**

int i**;**

**for** **(**i **=** 0**;** i**<**countries**.**length**;** i**++){**

**if(**countryName**.**equals**(**countries**[**i**])){**

**break;**

**}**

**}**

**return** countriesAlpha3Codes**[**i**];**

**}**

//ΜΕΘΟΔΟΣ ΑΝΑΖΗΤΗΣΗΣ ΚΑΤΑΧΩΡΗΜΕΝΗΣ ΕΓΓΡΑΦΗΣ

public boolean FindInDbByIsocode**(**String isocode**){**

Query q **=** em**.**createNamedQuery**(**"Country.findByIsoCode"**,** Country**.**class**);**

q**.**setParameter**(**"isoCode"**,** isocode**);**

//ΕΠΙΣΤΡΕΦΕΙ false ΑΝ ΔΕΝ ΥΠΑΡΧΕΙ ΚΑΤΑΧΩΡΙΣΗ ΚΑΙ true ΑΝ ΥΠΑΡΧΕΙ

**return** **!**q**.**getResultList**().**isEmpty**();**

**}**

//ΜΕΘΟΔΟΣ ΑΝΑΖΗΤΗΣΗΣ CountryDataset ΜΕ ΒΑΣΗ ΤΟ isocode

public List**<**CountryDataset**>** FindCountryDatasets**(**Country c**){**

**try** **{**

Query q **=** em**.**createNamedQuery**(**"CountryDataset.findByCountryCode"**,** CountryDataset**.**class**);**

q**.**setParameter**(**"countryCode"**,** c**);**

**return** **(**List**<**CountryDataset**>)** q**.**getResultList**();**

**}**

**catch** **(**Exception ex**){**

**return** **null;**

**}**

**}**

//ΜΕΘΟΔΟΣ ΑΝΑΖΗΤΗΣΗΣ CountryData ΜΕ ΒΑΣΗ ΤΟ Dataset id

public List**<**CountryData**>** FindCountryData**(**CountryDataset cd**){**

**try** **{**

Query q **=** em**.**createNamedQuery**(**"CountryData.findByDataset"**,** CountryData**.**class**);**

q**.**setParameter**(**"dataset"**,** cd**);**

**return** **(**List**<**CountryData**>)** q**.**getResultList**();**

**}**

**catch** **(**Exception ex**){**

**return** **null;**

**}**

**}**

//ΜΕΘΟΔΟΣ ΑΠΟΘΗΚΕΥΣΗΣ ΔΕΔΟΜΕΝΩΝ ΣΤΟΝ ΠΙΝΑΚΑ Country

public void insertDataToCountry**(**Country c**){**

**try{**

Query q **=** em**.**createNamedQuery**(**"Country.findAll"**);**

em**.**getTransaction**().**begin**();**

em**.**persist**(**c**);**

em**.**getTransaction**().**commit**();**

**}**

**catch** **(**Exception ex**)** **{**

System**.**out**.**println**(**ex**.**getMessage**());**

JOptionPane**.**showMessageDialog**(null,** "Προκλήθηκε σφάλμα κατά την εισαγωγή δεδομένων στη βάση. \n "**,**

"Σφάλμα"**,** JOptionPane**.**ERROR\_MESSAGE**);**

**}**

**}**

//ΜΕΘΟΔΟΣ ΑΠΟΘΗΚΕΥΣΗΣ ΔΕΔΟΜΕΝΩΝ ΣΤΟΝ ΠΙΝΑΚΑ Country\_Dataset

public void insertDatasetsToCountryDataset**(**List**<**CountryDataset**>** cds**){**

**try{**

Query q **=** em**.**createNamedQuery**(**"CountryDataset.findAll"**);**

em**.**getTransaction**().**begin**();**

//ΕΠΑΝΑΛΗΠΤΙΚΗ ΔΙΑΔΙΚΑΣΙΑ ΓΙΑ ΚΑΤΑΧΩΡΗΣΗ ΤΗΣ ΛΙΣΤΑΣ cds

cds**.**forEach**((**dataset**)** **->** **{**

//ΚΑΤΑΧΩΡΗΣΗ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

em**.**persist**(**dataset**);**

**});**

//ΑΠΟΘΗΚΕΥΣΗ ΤΗΣ ΚΑΤΑΧΩΡΗΣΗΣ

em**.**getTransaction**().**commit**();**

**}**

**catch** **(**Exception ex**)** **{**

System**.**out**.**println**(**ex**.**getMessage**());**

JOptionPane**.**showMessageDialog**(null,** "Προκλήθηκε σφάλμα κατά την εισαγωγή δεδομένων στη βάση. \n "**,**

"Σφάλμα"**,** JOptionPane**.**ERROR\_MESSAGE**);**

**}**

**}**

//ΜΕΘΟΔΟΣ ΑΠΟΘΗΚΕΥΣΗΣ ΔΕΔΟΜΕΝΩΝ ΣΤΟΝ ΠΙΝΑΚΑ Country\_Data

public void insertDataToCountryData**(**List**<**CountryData**>** cd**){**

**try{**

Query q **=** em**.**createNamedQuery**(**"CountryData.findAll"**);**

em**.**getTransaction**().**begin**();**

//ΕΠΑΝΑΛΗΠΤΙΚΗ ΔΙΑΔΙΚΑΣΙΑ ΓΙΑ ΚΑΤΑΧΩΡΗΣΗ ΤΗΣ ΛΙΣΤΑΣ cd

cd**.**forEach**((**data**)** **->** **{**

//ΚΑΤΑΧΩΡΗΣΗ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

em**.**persist**(**data**);**

**});**

//ΑΠΟΘΗΚΕΥΣΗ ΤΗΣ ΚΑΤΑΧΩΡΗΣΗΣ

em**.**getTransaction**().**commit**();**

**}**

**catch** **(**Exception ex**)** **{**

System**.**out**.**println**(**ex**.**getMessage**());**

JOptionPane**.**showMessageDialog**(null,** "Προκλήθηκε σφάλμα κατά την εισαγωγή δεδομένων στη βάση. \n "**,**

"Σφάλμα"**,** JOptionPane**.**ERROR\_MESSAGE**);**

**}**

**}**

//ΜΕΘΟΔΟΣ ΔΙΑΓΡΑΦΗΣ ΟΛΩΝ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΤΗΣ ΒΑΣΗΣ ΜΕ ΤΗΡΗΣΗ ΤΗΣ ΣΕΙΡΑ ΑΝΑΦΟΡΑΣ ΚΛΕΙΔΙΩΝ

public void clearTables**()** **{**

**try{**

Query q1 **=** em**.**createNamedQuery**(**"CountryData.deleteAll"**);**

Query q2 **=** em**.**createNamedQuery**(**"CountryDataset.deleteAll"**);**

Query q3 **=** em**.**createNamedQuery**(**"Country.deleteAll"**);**

//ΔΙΑΓΡΑΦΗ ΔΕΔΟΜΕΝΩΝ ΠΙΝΑΚΑ COUNTRY\_DATA

em**.**getTransaction**().**begin**();**

q1**.**executeUpdate**();**

em**.**getTransaction**().**commit**();**

//ΔΙΑΓΡΑΦΗ ΔΕΔΟΜΕΝΩΝ ΠΙΝΑΚΑ COUNTRY\_DATASET

em**.**getTransaction**().**begin**();**

q2**.**executeUpdate**();**

em**.**getTransaction**().**commit**();**

//ΔΙΑΓΡΑΦΗ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΠΙΝΑΚΑ COUNTRY

em**.**getTransaction**().**begin**();**

q3**.**executeUpdate**();**

em**.**getTransaction**().**commit**();**

**}**

**catch** **(**Exception e**){**

//ΣΕ ΠΕΡΙΠΤΩΣΗ ΣΦΑΛΜΑΤΟΣ ΕΠΑΝΑΦΟΡΑ

em**.**getTransaction**().**rollback**();**

**}**

**}**

//ΜΕΘΟΔΟΣ ΕΛΕΓΧΟΥ ΑΔΕΙΑΣ ΒΑΣΗΣ

public boolean EmptyDB**(){**

boolean empty **=** **false;**

**try** **{**

Query q **=** em**.**createNamedQuery**(**"Country.findAll"**,** Country**.**class**);**

empty **=** q**.**getResultList**().**isEmpty**();**

**}**

**catch** **(**Exception ex**){**

JOptionPane**.**showMessageDialog**(null,** "Προκλήθηκε σφάλμα κατά την αναζήτηση δεδομένων από τη βάση. \n "**,**

"Σφάλμα"**,** JOptionPane**.**ERROR\_MESSAGE**);**

**}**

**return** empty**;**

**}**

**}**

JsonRead

Η κλάση JsonRead, περιέχει τις μεθόδους που παίρνουν τα Json\_string από το quandl και αφού γίνει ο διαχωρισμός των πληροφοριών δημιουργεί αντικείμενα τα οποία παρουσιάζονται στους πίνακες της βάσης.

package econometrica**;**

**import** com**.**google**.**gson**.**JsonArray**;**

**import** com**.**google**.**gson**.**JsonElement**;**

**import** com**.**google**.**gson**.**JsonObject**;**

**import** com**.**google**.**gson**.**JsonParser**;**

**import** databaseClasses**.\*;**

**import** java**.**text**.**DecimalFormat**;**

**import** java**.**text**.**ParseException**;**

**import** java**.**text**.**SimpleDateFormat**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Date**;**

**import** java**.**util**.**List**;**

**import** java**.**util**.**logging**.**Level**;**

**import** java**.**util**.**logging**.**Logger**;**

public class JsonRead **{**

CountryDataset cDataset**;** //ΑΝΤΙΚΕΙΜΕΝΟ CountryDataset

CountryData cData**;** //ΑΝΤΙΚΕΙΜΕΝΟ CountryData

List**<**CountryData**>** allData **=** **new** ArrayList**<>();** //ΛΙΣΤΑ ΜΕ ΟΛΑ ΤΑ CountryData ΤΗΣ ΧΩΡΑΣ

List**<**String**>** datasetInfo**;** //ΛΙΣΤΑ ΜΕ ΠΛΗΡΟΦΟΡΙΕΣ ΤΟΥ dataset ΓΙΑ ΕΜΦΑΝΙΣΗ ΣΤΟ GUI

public CountryDataset GetDataFromJsonAndSaveToDB**(**String jsonString**,** Country country**){**

datasetInfo **=** **new** ArrayList**<>();**

//ΔΙΑΧΕΙΡΙΣΗ ΔΕΔΟΜΕΝΩΝ ΤΥΠΟΥ JSON

JsonParser parser **=** **new** JsonParser**();**

JsonElement jsonTree **=** parser**.**parse**(**jsonString**);**

**if(**jsonTree**.**isJsonObject**()){**

JsonObject jsonObject **=** jsonTree**.**getAsJsonObject**();**

JsonElement dataset **=** jsonObject**.**get**(**"dataset"**);**

**if(**dataset**.**isJsonObject**()){**

JsonObject datasetObj **=** dataset**.**getAsJsonObject**();**

String name **=** datasetObj**.**get**(**"name"**).**getAsString**();**

String description **=** datasetObj**.**get**(**"description"**).**getAsString**();**

String startDate **=** datasetObj**.**get**(**"start\_date"**).**getAsString**();**

String endDate **=** datasetObj**.**get**(**"end\_date"**).**getAsString**();**

JsonElement data **=** datasetObj**.**get**(**"data"**);**

datasetInfo**.**add**(**name**);**

datasetInfo**.**add**(**startDate**);**

datasetInfo**.**add**(**endDate**);**

//DATEFORMAT ΓΙΑ ΔΙΑΧΕΙΡΙΣΗ ΗΜΕΡΟΜΗΝΙΩΝ

SimpleDateFormat dateformat **=** **new** SimpleDateFormat**(**"yyyy"**);**

String startYear**,** endYear**;**

Date sDate **=** **null,** eDate **=** **null;**

**try** **{**

sDate **=** dateformat**.**parse**(**startDate**);**

eDate **=** dateformat**.**parse**(**endDate**);**

**}** **catch** **(**ParseException ex**)** **{**

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

**}**

startYear **=** dateformat**.**format**(**sDate**);**

endYear **=** dateformat**.**format**(**eDate**);**

//ΔΗΜΙΟΥΡΓΙΑ ΛΙΣΤΑΣ ΑΝΤΙΚΕΙΜΕΝΩΝ CountryDataset ΜΕ ΣΚΟΠΟ ΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΣΤΗ ΒΑΣΗ

cDataset **=** **new** CountryDataset**(**name**,** description**,** startYear**,** endYear**,** country**);**

//ΔΙΑΧΕΙΡΙΣΗ ΔΕΔΟΜΕΝΩΝ ΤΥΠΟΥ JSON\_ARRAY

**if(**data**.**isJsonArray**()){**

JsonArray dataArray **=** data**.**getAsJsonArray**();**

**for(**int i**=**0**;** i**<**dataArray**.**size**();** i**++){**

JsonElement idata **=** dataArray**.**get**(**i**);**

**if(**idata**.**isJsonArray**()){**

JsonArray yearDataArray **=** idata**.**getAsJsonArray**();**

String date **=** yearDataArray**.**get**(**0**).**getAsString**();**

Double value **=** yearDataArray**.**get**(**1**).**getAsDouble**();**

Date yearDate **=** **null;**

//ΜΕΤΑΤΡΟΠΗ dataDate String ΣΕ Date

**try** **{**

yearDate **=** dateformat**.**parse**(**date**);**

**}** **catch** **(**ParseException ex**)** **{**

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

**}**

//ΜΕΤΑΤΡΟΠΗ Date ΣΕ String ΜΕ ΔΙΑΤΗΡΗΣΗ ΜΟΝΟ ΤΟΥ ΕΤΟΥΣ

String year**=** dateformat**.**format**(**yearDate**);**

//ΜΕΤΑΤΡΟΠΕΑΣ ΤΙΜΗΣ ΣΕ 2 ΔΕΚΑΔΙΚΑ

DecimalFormat df1 **=** **new** DecimalFormat**(**"0.00"**);**

cData **=** **new** CountryData**(**year**,** df1**.**format**(**value**),** cDataset**);**

allData**.**add**(**cData**);**

**}**

**}**

**}**

**}**

**}**

**return** cDataset**;**

**}**

**}**

PlotMaker

Η κλάση PlotMaker, δημιουργεί τα 2 dataset για το oil consumption και το GDP και εμφανίζει το διάγραμμα.

package econometrica**;**

**import** databaseClasses**.\*;**

**import** org**.**jfree**.**ui**.**ApplicationFrame**;**

**import** java**.**awt**.**Color**;**

**import** java**.**text**.**NumberFormat**;**

**import** java**.**text**.**ParseException**;**

**import** java**.**text**.**SimpleDateFormat**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Date**;**

**import** java**.**util**.**List**;**

**import** java**.**util**.**Locale**;**

**import** java**.**util**.**logging**.**Level**;**

**import** java**.**util**.**logging**.**Logger**;**

**import** org**.**jfree**.**chart**.**ChartFactory**;**

**import** org**.**jfree**.**chart**.**ChartPanel**;**

**import** org**.**jfree**.**chart**.**JFreeChart**;**

**import** org**.**jfree**.**chart**.**axis**.**DateAxis**;**

**import** org**.**jfree**.**chart**.**axis**.**NumberAxis**;**

**import** org**.**jfree**.**chart**.**labels**.**StandardXYToolTipGenerator**;**

**import** org**.**jfree**.**chart**.**plot**.**XYPlot**;**

**import** org**.**jfree**.**chart**.**renderer**.**xy**.**StandardXYItemRenderer**;**

**import** org**.**jfree**.**chart**.**renderer**.**xy**.**XYItemRenderer**;**

**import** org**.**jfree**.**data**.**time**.**TimeSeries**;**

**import** org**.**jfree**.**data**.**time**.**TimeSeriesCollection**;**

**import** org**.**jfree**.**data**.**time**.**Year**;**

**import** org**.**jfree**.**data**.**xy**.**XYDataset**;**

public class PlotMaker **extends** ApplicationFrame**{**

//ΛΙΣΤΕΣ ΓΙΑ ΔΙΑΧΩΡΙΣΜΟ ΤΩΝ ΑΝΤΙΚΕΙΜΕΝΩΝ ΤΩΝ 2 DATASETS

private static List**<**CountryData**>** datalist1**,** datalist2**;**

static SimpleDateFormat dateformat **=** **new** SimpleDateFormat**(**"yyyy"**);**

final ChartPanel graphPanel**;**

//CONSTRUCTOR

public PlotMaker**(**String title**,** List**<**CountryData**>** alldata**,** List**<**CountryDataset**>** datasetList**){**

**super(**title**);**

final String chartTitle **=** title**;**

datalist1 **=** **new** ArrayList**<>();**

datalist2 **=** **new** ArrayList**<>();**

//ΔΙΑΧΩΡΙΣΜΟΣ ΑΝΤΙΚΕΙΜΕΝΩΝ Country\_Data ΓΙΑ ΤΟ ΚΑΘΕ DATASET ΑΠΟ ΤΗ ΛΙΣΤΑ

//alldata Η ΟΠΟΙΑ ΠΕΡΙΕΧΕΙ ΟΛΑ ΤΑ Country\_Data ΤΗΣ ΧΩΡΑΣ

**for** **(**int i **=** 0**;** i**<**alldata**.**size**();** i**++){**

**if(**alldata**.**get**(**i**).**getDataset**().**getName**().**startsWith**(**"Oil"**)){**

datalist1**.**add**(**alldata**.**get**(**i**));**

**}else{**

datalist2**.**add**(**alldata**.**get**(**i**));**

**}**

**}**

final XYDataset dataset **=** createDataset1**();**

final JFreeChart graph **=** ChartFactory**.**createTimeSeriesChart**(**

chartTitle**,**

"Year"**,**

"GDP"**,**

dataset**,**

**true,**

**true,**

**false);**

final XYPlot plot **=** graph**.**getXYPlot**();**

final NumberAxis axis2 **=** **new** NumberAxis**(**"Oil Consumption"**);**

axis2**.**setAutoRangeIncludesZero**(false);**

plot**.**setRangeAxis**(**1**,** axis2**);**

plot**.**setDataset**(**1**,** createDataset2**());**

plot**.**mapDatasetToRangeAxis**(**1**,** 1**);**

final XYItemRenderer renderer **=** plot**.**getRenderer**();**

renderer**.**setToolTipGenerator**(**StandardXYToolTipGenerator**.**getTimeSeriesInstance**());**

**if** **(**renderer **instanceof** StandardXYItemRenderer**)** **{**

final StandardXYItemRenderer rr **=** **(**StandardXYItemRenderer**)** renderer**;**

rr**.**setShapesFilled**(true);**

**}**

final StandardXYItemRenderer renderer2 **=** **new** StandardXYItemRenderer**();**

renderer2**.**setSeriesPaint**(**0**,** Color**.**BLUE**);**

renderer**.**setToolTipGenerator**(**StandardXYToolTipGenerator**.**getTimeSeriesInstance**());**

plot**.**setRenderer**(**1**,** renderer2**);**

final DateAxis axis **=** **(**DateAxis**)**plot**.**getDomainAxis**();**

axis**.**setDateFormatOverride**(**dateformat**);**

graphPanel **=** **new** ChartPanel**(**graph**);**

graphPanel**.**setPreferredSize**(new** java**.**awt**.**Dimension**(**1500**,** 800**));**

setContentPane**(**graphPanel**);**

**}**

//ΜΕΘΟΔΟΣ ΔΗΜΙΟΥΡΓΙΑΣ XYDataset

private static XYDataset createDataset1**(){**

Date year **=** **null;**

Number value **=** **null;**

final TimeSeries s1 **=** **new** TimeSeries**(**"Gross Domestic Product (GDP)"**,** Year**.**class**);**

//NumberFormat ΓΙΑ ΑΝΤΙΜΕΤΩΠΙΣΗ ΤΟΥ ΚΟΜΜΑ ΣΤΟΝ ΑΡΙΘΜΟ

NumberFormat nf **=** NumberFormat**.**getInstance**(**Locale**.**ENGLISH**);**

//ΓΙΑ ΚΑΘΕ ΑΝΤΙΚΕΙΜΕΝΟ Country\_Data ΤΟΥ dataset1 ΧΡΗΣΙΜΟΠΟΙΟΥΜΕ ΤΙΜΗ ΚΑΙ ΕΤΟΣ

**for** **(**int i **=** 0**;** i**<**datalist2**.**size**();** i**++){**

**try** **{**

value **=** nf**.**parse**(**datalist2**.**get**(**i**).**getValue**());**

year **=** dateformat**.**parse**(**datalist2**.**get**(**i**).**getDataYear**());**

**}** **catch** **(**ParseException ex**)** **{**

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

**}**

s1**.**add**(new** Year**(**year**),** value**);**

**}**

final TimeSeriesCollection dataset **=** **new** TimeSeriesCollection**();**

dataset**.**addSeries**(**s1**);**

**return** dataset**;**

**}**

//ΜΕΘΟΔΟΣ ΔΗΜΙΟΥΡΓΙΑΣ XYDataset

private XYDataset createDataset2**()** **{**

Date year **=** **null;**

Number value **=** **null;**

final TimeSeries s2 **=** **new** TimeSeries**(**"Oil Consumption"**,** Year**.**class**);**

//NumberFormat ΓΙΑ ΑΝΤΙΜΕΤΩΠΙΣΗ ΤΟΥ ΚΟΜΜΑ ΣΤΟΝ ΑΡΙΘΜΟ

NumberFormat nf **=** NumberFormat**.**getInstance**(**Locale**.**ENGLISH**);**

**for** **(**int i **=** 0**;** i**<**datalist1**.**size**();** i**++){**

**try** **{**

value **=** nf**.**parse**(**datalist1**.**get**(**i**).**getValue**());**

year **=** dateformat**.**parse**(**datalist1**.**get**(**i**).**getDataYear**());**

**}** **catch** **(**ParseException ex**)** **{**

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

**}**

s2**.**add**(new** Year**(**year**),** value**);**

**}**

final TimeSeriesCollection dataset **=** **new** TimeSeriesCollection**();**

dataset**.**addSeries**(**s2**);**

**return** dataset**;**

**}**

**}**

**Γ)**

**Κώδικας POJO Κλάσεων**

Μετά τη δημιουργία της βάσης δεδομένων «Econometrica\_Data\_DB» χρησιμοποιήθηκε ο οδηγός του NetBeans για τη δημιουργία κατάλληλων κλάσεων οντοτήτων που σχετίζονται με τη βάση. Οι κλάσεις αυτές που δημιουργήθηκαν είναι οι POJO κλάσεις και ουσιαστικά αποτελούν τη «μετάφραση» της βάσης δεδομένων σε κλάσεις Java. Δεδομένου ότι η βάση δεδομένων περιλαμβάνει 3 διακριτούς πίνακες, έχουν δημιουργηθεί και 3 διαφορετικές κλάσεις POJO. Οι παραχθείσες κλάσεις τοποθετήθηκαν σε χωριστό πακέτο με την ονομασία «databaseClasses», του οποίου η δομή είναι η εξής:

|  |  |
| --- | --- |
| **Κλάση** | **Πίνακας** |
| **Country.java** | Αντιστοιχεί στον πίνακα Country |
| **CountryData.java** | Αντιστοιχεί στον πίνακα Country\_Data |
| **CountryDataset.java** | Αντιστοιχεί στον πίνακα Country\_Dataset |

**Περιγραφή πακέτου «**databaseClasses**»**: Πρόκειται για το πακέτο που δεν περιλαμβάνει τίποτα άλλο, παρά τις POJO κλάσεις που αντιστοιχούν στη βάση δεδομένων και είναι απαραίτητες για τη λειτουργία της εφαρμογής και την επικοινωνία της με τους πίνακες της βάσης (ενημέρωση και ανάκτηση).

Ο μοναδικός κώδικας που αναπτύχθηκε είναι τα παρακάτω queries:

Country

//NamedQuery ΓΙΑ ΔΙΑΓΡΑΦΗ ΟΛΩΝ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΤΟΥ ΠΙΝΑΚΑ COUNTRY

, @NamedQuery(name = "Country.deleteAll", query = "DELETE FROM Country")})

CountryData

//NamedQuery ΓΙΑ ΕΥΡΕΣΗ Data ΜΕ ΒΑΣΗ ΤΟ dataset

**,** @NamedQuery**(**name **=** "CountryData.findByDataset"**,** query **=** "SELECT c FROM CountryData c WHERE c.dataset = :dataset ORDER BY c.dataYear DESC"**)**

//NamedQuery ΓΙΑ ΔΙΑΓΡΑΦΗ ΟΛΩΝ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΤΟΥ ΠΙΝΑΚΑ COUNTRY\_DATA

**,** @NamedQuery**(**name **=** "CountryData.deleteAll"**,** query **=** "DELETE FROM CountryData"**)})**

CountryDataset

//NamedQuery ΓΙΑ ΕΥΡΕΣΗ Dataset ΜΕ ΒΑΣΗ ΤΟ countryCode

**,** @NamedQuery**(**name **=** "CountryDataset.findByCountryCode"**,** query **=** "SELECT c FROM CountryDataset c WHERE c.countryCode = :countryCode"**)**

//NamedQuery ΓΙΑ ΔΙΑΓΡΑΦΗ ΟΛΩΝ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΤΟΥ ΠΙΝΑΚΑ COUNTRY\_DATASET

**,** @NamedQuery**(**name **=** "CountryDataset.deleteAll"**,** query **=** "DELETE FROM CountryDataset"**)})**

## Δημιουργία GUI Εφαρμογής

Στη συνέχεια ακολουθεί ο κώδικας Java για την κλάση του συστήματος (γραφικές διεπαφές χρήστη/GUI). Για την κλάση αυτή δεν παρατίθεται το τμήμα του κώδικα που παράγεται αυτόματα, αλλά μόνο το κομμάτι εκείνο που έχει αναπτυχθεί από την ομάδα ανάπτυξης του έργου. Σημειώνεται ότι ο κώδικας είναι σχολιασμένος εσωτερικά.

EconemetricaGUI

package econometrica**;**

**import** databaseClasses**.\*;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**List**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**JOptionPane**;**

**import** javax**.**swing**.**table**.**DefaultTableCellRenderer**;**

**import** javax**.**swing**.**table**.**DefaultTableModel**;**

**import** org**.**jfree**.**ui**.**RefineryUtilities**;**

public class EconometricaGUI **extends** javax**.**swing**.**JFrame **{**

protected ControllerGUI controller**;**

String countryName **=** ""**;**

String isocode**;**

Country country**;** //ΑΝΤΙΚΕΙΜΕΝΟ Country

CountryDataset cDataset**;** //ΑΝΤΙΚΕΙΜΕΝΟ CountryDataset

List**<**CountryDataset**>** cDatasets**;** //ΛΙΣΤΑ ΜΕ ΤΑ CountryDatasets ΤΗΣ ΧΩΡΑΣ

List**<**CountryData**>** allCountryData**;** //ΛΙΣΤΑ ΜΕ ΟΛΑ ΤΑ CountryData ΤΗΣ ΧΩΡΑΣ

DefaultTableCellRenderer centerRenderer **=** **new** DefaultTableCellRenderer**();**

private PlotMaker plot**;**

/\*\*

\* Creates new form EconometricaGUI

\*/

public EconometricaGUI**()** **{**

//ΔΗΜΙΟΥΡΓΙΑ GUI CONTROLLER

controller **=** **new** ControllerGUI**();**

initComponents**();**

//ΠΡΟΣΘΗΚΗ Listener ΓΙΑ ΕΛΕΓΧΟ ΤΟΥ ΚΟΥΜΠΙΟΥ ΚΛΕΙΣΙΜΑΤΟΣ

//ΤΟΥ ΚΕΝΤΡΙΚΟΥ ΠΑΡΑΘΥΡΟΥ

**this.**addWindowListener**(new** java**.**awt**.**event**.**WindowAdapter**()** **{**

@Override

public void windowClosing**(**java**.**awt**.**event**.**WindowEvent windowEvent**)** **{**

//ΚΛΗΣΗ ΜΕΘΟΔΟΥ ΕΠΙΛΟΓΗΣ ΓΙΑ ΕΞΟΔΟ ΑΠΟ ΕΦΑΡΜΟΓΗ

ExitOnCloseOption**();**

**}**

**});**

**}**

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings**(**"unchecked"**)**

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents**()** **{**

countrySelectjLabel **=** **new** javax**.**swing**.**JLabel**();**

fetchDatajButton **=** **new** javax**.**swing**.**JButton**();**

oilDatajLabel **=** **new** javax**.**swing**.**JLabel**();**

oilDatasetNameTitlejLabel **=** **new** javax**.**swing**.**JLabel**();**

gdpDatajLabel **=** **new** javax**.**swing**.**JLabel**();**

gdpDatasetNameTitlejLabel **=** **new** javax**.**swing**.**JLabel**();**

oilDatasetNamejlabel **=** **new** javax**.**swing**.**JLabel**();**

gdpDatasetNamejlabel **=** **new** javax**.**swing**.**JLabel**();**

avTimespanjLabel **=** **new** javax**.**swing**.**JLabel**();**

avTimespanjLabel1 **=** **new** javax**.**swing**.**JLabel**();**

oilStartDateTitlejLabel **=** **new** javax**.**swing**.**JLabel**();**

gdpStartDateTitlejLabel **=** **new** javax**.**swing**.**JLabel**();**

oilEndDateTitlejLabel **=** **new** javax**.**swing**.**JLabel**();**

gdpEndDateTitlejLabel **=** **new** javax**.**swing**.**JLabel**();**

oilStartDatejlabel **=** **new** javax**.**swing**.**JLabel**();**

oilEndDatejlabel **=** **new** javax**.**swing**.**JLabel**();**

gdpStartDatejlabel **=** **new** javax**.**swing**.**JLabel**();**

gdpEndDatejlabel **=** **new** javax**.**swing**.**JLabel**();**

savejButton **=** **new** javax**.**swing**.**JButton**();**

plotjButton **=** **new** javax**.**swing**.**JButton**();**

deleteAlljButton **=** **new** javax**.**swing**.**JButton**();**

alreadySavedjCheckBox **=** **new** javax**.**swing**.**JCheckBox**();**

countryjComboBox **=** **new** javax**.**swing**.**JComboBox**<>();**

jScrollPane3 **=** **new** javax**.**swing**.**JScrollPane**();**

oilDatajTable **=** **new** javax**.**swing**.**JTable**();**

jScrollPane4 **=** **new** javax**.**swing**.**JScrollPane**();**

gdpDatajTable **=** **new** javax**.**swing**.**JTable**();**

setDefaultCloseOperation**(**javax**.**swing**.**WindowConstants**.**DO\_NOTHING\_ON\_CLOSE**);**

setTitle**(**"Econometrica"**);**

addWindowFocusListener**(new** java**.**awt**.**event**.**WindowFocusListener**()** **{**

public void windowGainedFocus**(**java**.**awt**.**event**.**WindowEvent evt**)** **{**

formWindowGainedFocus**(**evt**);**

**}**

public void windowLostFocus**(**java**.**awt**.**event**.**WindowEvent evt**)** **{**

**}**

**});**

addWindowListener**(new** java**.**awt**.**event**.**WindowAdapter**()** **{**

public void windowActivated**(**java**.**awt**.**event**.**WindowEvent evt**)** **{**

formWindowActivated**(**evt**);**

**}**

**});**

countrySelectjLabel**.**setFont**(new** java**.**awt**.**Font**(**"Tahoma"**,** 1**,** 16**));** // NOI18N

countrySelectjLabel**.**setText**(**"Select Country:"**);**

fetchDatajButton**.**setText**(**"Fetch Data"**);**

fetchDatajButton**.**setToolTipText**(**"Αναζήτηση δεδομένων"**);**

fetchDatajButton**.**addMouseListener**(new** java**.**awt**.**event**.**MouseAdapter**()** **{**

public void mouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**

fetchDatajButtonMouseClicked**(**evt**);**

**}**

**});**

fetchDatajButton**.**addActionListener**(new** java**.**awt**.**event**.**ActionListener**()** **{**

public void actionPerformed**(**java**.**awt**.**event**.**ActionEvent evt**)** **{**

fetchDatajButtonActionPerformed**(**evt**);**

**}**

**});**

oilDatajLabel**.**setFont**(new** java**.**awt**.**Font**(**"Tahoma"**,** 1**,** 16**));** // NOI18N

oilDatajLabel**.**setText**(**"OIL DATA"**);**

oilDatasetNameTitlejLabel**.**setFont**(new** java**.**awt**.**Font**(**"Tahoma"**,** 1**,** 16**));** // NOI18N

oilDatasetNameTitlejLabel**.**setText**(**"Dataset Name"**);**

gdpDatajLabel**.**setFont**(new** java**.**awt**.**Font**(**"Tahoma"**,** 1**,** 16**));** // NOI18N

gdpDatajLabel**.**setText**(**"GDP DATA"**);**

gdpDatasetNameTitlejLabel**.**setFont**(new** java**.**awt**.**Font**(**"Tahoma"**,** 1**,** 16**));** // NOI18N

gdpDatasetNameTitlejLabel**.**setText**(**"Dataset Name"**);**

oilDatasetNamejlabel**.**setFont**(new** java**.**awt**.**Font**(**"Dialog"**,** 0**,** 14**));** // NOI18N

oilDatasetNamejlabel**.**setText**(**"-"**);**

gdpDatasetNamejlabel**.**setFont**(new** java**.**awt**.**Font**(**"Dialog"**,** 0**,** 14**));** // NOI18N

gdpDatasetNamejlabel**.**setText**(**"-"**);**

avTimespanjLabel**.**setFont**(new** java**.**awt**.**Font**(**"Tahoma"**,** 1**,** 16**));** // NOI18N

avTimespanjLabel**.**setText**(**"Available Timespan:"**);**

avTimespanjLabel1**.**setFont**(new** java**.**awt**.**Font**(**"Tahoma"**,** 1**,** 16**));** // NOI18N

avTimespanjLabel1**.**setText**(**"Available Timespan:"**);**

oilStartDateTitlejLabel**.**setText**(**"Start Date:"**);**

gdpStartDateTitlejLabel**.**setText**(**"Start Date:"**);**

oilEndDateTitlejLabel**.**setText**(**"End Date:"**);**

gdpEndDateTitlejLabel**.**setText**(**"End Date:"**);**

oilStartDatejlabel**.**setText**(**"-"**);**

oilEndDatejlabel**.**setText**(**"-"**);**

gdpStartDatejlabel**.**setText**(**"-"**);**

gdpEndDatejlabel**.**setText**(**"-"**);**

savejButton**.**setText**(**"Save"**);**

savejButton**.**setToolTipText**(**"Αποθήκευση"**);**

savejButton**.**setEnabled**(false);**

savejButton**.**addActionListener**(new** java**.**awt**.**event**.**ActionListener**()** **{**

public void actionPerformed**(**java**.**awt**.**event**.**ActionEvent evt**)** **{**

savejButtonActionPerformed**(**evt**);**

**}**

**});**

plotjButton**.**setText**(**"Plot"**);**

plotjButton**.**setToolTipText**(**"Προβολή διαγράμματος"**);**

plotjButton**.**setEnabled**(false);**

plotjButton**.**addActionListener**(new** java**.**awt**.**event**.**ActionListener**()** **{**

public void actionPerformed**(**java**.**awt**.**event**.**ActionEvent evt**)** **{**

plotjButtonActionPerformed**(**evt**);**

**}**

**});**

deleteAlljButton**.**setText**(**"DELETE ALL"**);**

deleteAlljButton**.**setToolTipText**(**"Διαγραφή όλων των δεδομένων από τη βάση"**);**

deleteAlljButton**.**addMouseListener**(new** java**.**awt**.**event**.**MouseAdapter**()** **{**

public void mouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**

deleteAlljButtonMouseClicked**(**evt**);**

**}**

**});**

deleteAlljButton**.**addActionListener**(new** java**.**awt**.**event**.**ActionListener**()** **{**

public void actionPerformed**(**java**.**awt**.**event**.**ActionEvent evt**)** **{**

deleteAlljButtonActionPerformed**(**evt**);**

**}**

**});**

//ΕΝΕΡΓΟΠΟΙΗΣΗ ΚΟΥΜΠΙΟΥ deleteAlljButton ΜΌΝΟ ΑΝ ΥΠΑΡΧΟΥΝ ΔΕΔΟΜΕΝΑ ΣΤΗ ΒΑΣΗ

**if(**controller**.**EmptyDB**()){**deleteAlljButton**.**setEnabled**(false);**

**}else{**deleteAlljButton**.**setEnabled**(true);}**

alreadySavedjCheckBox**.**setText**(**"Already Saved to Database"**);**

alreadySavedjCheckBox**.**setEnabled**(false);**

alreadySavedjCheckBox**.**addActionListener**(new** java**.**awt**.**event**.**ActionListener**()** **{**

public void actionPerformed**(**java**.**awt**.**event**.**ActionEvent evt**)** **{**

alreadySavedjCheckBoxActionPerformed**(**evt**);**

**}**

**});**

countryjComboBox**.**setToolTipText**(**"Επιλογή χώρας"**);**

//ΦΟΡΤΩΣΗ ΔΕΔΟΜΕΝΩΝ ΣΤΟ JComboBox

countryjComboBox**.**setModel**(new** javax**.**swing**.**DefaultComboBoxModel**<>(**controller**.**getCountries**()));**

countryjComboBox**.**addItemListener**(new** java**.**awt**.**event**.**ItemListener**()** **{**

public void itemStateChanged**(**java**.**awt**.**event**.**ItemEvent evt**)** **{**

countryjComboBoxItemStateChanged**(**evt**);**

**}**

**});**

countryjComboBox**.**addActionListener**(new** java**.**awt**.**event**.**ActionListener**()** **{**

public void actionPerformed**(**java**.**awt**.**event**.**ActionEvent evt**)** **{**

countryjComboBoxActionPerformed**(**evt**);**

**}**

**});**

oilDatajTable**.**setModel**(new** javax**.**swing**.**table**.**DefaultTableModel**(**

**new** Object **[][]** **{**

**},**

**new** String **[]** **{**

"Year"**,** "Value"

**}**

**)** **{**

Class**[]** types **=** **new** Class **[]** **{**

java**.**lang**.**String**.**class**,** java**.**lang**.**String**.**class

**};**

boolean**[]** canEdit **=** **new** boolean **[]** **{**

**false,** **false**

**};**

public Class getColumnClass**(**int columnIndex**)** **{**

**return** types **[**columnIndex**];**

**}**

public boolean isCellEditable**(**int rowIndex**,** int columnIndex**)** **{**

**return** canEdit **[**columnIndex**];**

**}**

**});**

oilDatajTable**.**setEnabled**(false);**

oilDatajTable**.**getTableHeader**().**setReorderingAllowed**(false);**

jScrollPane3**.**setViewportView**(**oilDatajTable**);**

**if** **(**oilDatajTable**.**getColumnModel**().**getColumnCount**()** **>** 0**)** **{**

oilDatajTable**.**getColumnModel**().**getColumn**(**0**).**setResizable**(false);**

oilDatajTable**.**getColumnModel**().**getColumn**(**1**).**setResizable**(false);**

**}**

gdpDatajTable**.**setModel**(new** javax**.**swing**.**table**.**DefaultTableModel**(**

**new** Object **[][]** **{**

**},**

**new** String **[]** **{**

"Year"**,** "Value"

**}**

**)** **{**

Class**[]** types **=** **new** Class **[]** **{**

java**.**lang**.**String**.**class**,** java**.**lang**.**String**.**class

**};**

boolean**[]** canEdit **=** **new** boolean **[]** **{**

**false,** **false**

**};**

public Class getColumnClass**(**int columnIndex**)** **{**

**return** types **[**columnIndex**];**

**}**

public boolean isCellEditable**(**int rowIndex**,** int columnIndex**)** **{**

**return** canEdit **[**columnIndex**];**

**}**

**});**

gdpDatajTable**.**setEnabled**(false);**

gdpDatajTable**.**getTableHeader**().**setReorderingAllowed**(false);**

jScrollPane4**.**setViewportView**(**gdpDatajTable**);**

**if** **(**gdpDatajTable**.**getColumnModel**().**getColumnCount**()** **>** 0**)** **{**

gdpDatajTable**.**getColumnModel**().**getColumn**(**0**).**setResizable**(false);**

gdpDatajTable**.**getColumnModel**().**getColumn**(**1**).**setResizable**(false);**

**}**

//ΑΡΧΙΚΗ ΚΕΝΤΡΙΚΗ ΣΤΟΙΧΙΣΗ JTABLES

centerRenderer**.**setHorizontalAlignment**(** JLabel**.**CENTER **);**

**for(**int i**=**0**;** i**<**2**;** i**++){**

oilDatajTable**.**getColumnModel**().**getColumn**(**i**).**setHeaderRenderer**(**centerRenderer**);**

oilDatajTable**.**getColumnModel**().**getColumn**(**i**).**setCellRenderer**(**centerRenderer**);**

gdpDatajTable**.**getColumnModel**().**getColumn**(**i**).**setHeaderRenderer**(**centerRenderer**);**

gdpDatajTable**.**getColumnModel**().**getColumn**(**i**).**setCellRenderer**(**centerRenderer**);**

**}**

javax**.**swing**.**GroupLayout layout **=** **new** javax**.**swing**.**GroupLayout**(**getContentPane**());**

getContentPane**().**setLayout**(**layout**);**

layout**.**setHorizontalGroup**(**

layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addGroup**(**layout**.**createSequentialGroup**()**

**.**addGap**(**25**,** 25**,** 25**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addComponent**(**countrySelectjLabel**)**

**.**addGroup**(**layout**.**createSequentialGroup**()**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addGroup**(**layout**.**createSequentialGroup**()**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addComponent**(**oilEndDateTitlejLabel**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 70**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**

**.**addComponent**(**oilStartDateTitlejLabel**))**

**.**addGap**(**67**,** 67**,** 67**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**,** **false)**

**.**addComponent**(**oilStartDatejlabel**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** 80**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**oilEndDatejlabel**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**)))**

**.**addComponent**(**avTimespanjLabel**)**

**.**addComponent**(**oilDatasetNamejlabel**)**

**.**addComponent**(**oilDatasetNameTitlejLabel**)**

**.**addComponent**(**oilDatajLabel**))**

**.**addGap**(**160**,** 160**,** 160**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addComponent**(**gdpDatasetNameTitlejLabel**)**

**.**addComponent**(**gdpDatajLabel**)**

**.**addComponent**(**gdpDatasetNamejlabel**)**

**.**addComponent**(**avTimespanjLabel1**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**TRAILING**,** **false)**

**.**addGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**,** layout**.**createSequentialGroup**()**

**.**addComponent**(**gdpEndDateTitlejLabel**)**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**RELATED**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**gdpEndDatejlabel**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 80**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**))**

**.**addGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**,** layout**.**createSequentialGroup**()**

**.**addComponent**(**gdpStartDateTitlejLabel**)**

**.**addGap**(**77**,** 77**,** 77**)**

**.**addComponent**(**gdpStartDatejlabel**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 80**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)))))**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**TRAILING**,** **false)**

**.**addGroup**(**layout**.**createSequentialGroup**()**

**.**addComponent**(**savejButton**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 130**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**

**.**addGap**(**30**,** 30**,** 30**)**

**.**addComponent**(**plotjButton**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 130**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**

**.**addGap**(**30**,** 30**,** 30**)**

**.**addComponent**(**deleteAlljButton**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 140**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**RELATED**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**alreadySavedjCheckBox**))**

**.**addGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**,** layout**.**createSequentialGroup**()**

**.**addComponent**(**jScrollPane3**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 348**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**

**.**addGap**(**27**,** 27**,** 27**)**

**.**addComponent**(**jScrollPane4**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 0**,** Short**.**MAX\_VALUE**))**

**.**addGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**,** layout**.**createSequentialGroup**()**

**.**addComponent**(**countryjComboBox**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 502**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**

**.**addGap**(**31**,** 31**,** 31**)**

**.**addComponent**(**fetchDatajButton**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 190**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**))))**

**.**addContainerGap**(**41**,** Short**.**MAX\_VALUE**))**

**);**

layout**.**setVerticalGroup**(**

layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addGroup**(**layout**.**createSequentialGroup**()**

**.**addGap**(**24**,** 24**,** 24**)**

**.**addComponent**(**countrySelectjLabel**)**

**.**addGap**(**1**,** 1**,** 1**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**fetchDatajButton**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 50**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**

**.**addComponent**(**countryjComboBox**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**))**

**.**addGap**(**22**,** 22**,** 22**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**oilDatajLabel**)**

**.**addComponent**(**gdpDatajLabel**))**

**.**addGap**(**0**,** 0**,** 0**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**oilDatasetNameTitlejLabel**)**

**.**addComponent**(**gdpDatasetNameTitlejLabel**))**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**UNRELATED**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**oilDatasetNamejlabel**)**

**.**addComponent**(**gdpDatasetNamejlabel**))**

**.**addGap**(**46**,** 46**,** 46**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**avTimespanjLabel**)**

**.**addComponent**(**avTimespanjLabel1**))**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**RELATED**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**oilStartDateTitlejLabel**)**

**.**addComponent**(**gdpStartDateTitlejLabel**)**

**.**addComponent**(**oilStartDatejlabel**)**

**.**addComponent**(**gdpStartDatejlabel**))**

**.**addGap**(**1**,** 1**,** 1**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**oilEndDateTitlejLabel**)**

**.**addComponent**(**gdpEndDateTitlejLabel**)**

**.**addComponent**(**oilEndDatejlabel**)**

**.**addComponent**(**gdpEndDatejlabel**))**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**UNRELATED**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**,** **false)**

**.**addComponent**(**jScrollPane4**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** 203**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**jScrollPane3**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 0**,** Short**.**MAX\_VALUE**))**

**.**addGap**(**26**,** 26**,** 26**)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**,** **false)**

**.**addGroup**(**layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**deleteAlljButton**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 50**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**

**.**addComponent**(**alreadySavedjCheckBox**))**

**.**addComponent**(**savejButton**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**plotjButton**,** javax**.**swing**.**GroupLayout**.**Alignment**.**TRAILING**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**))**

**.**addContainerGap**(**22**,** Short**.**MAX\_VALUE**))**

**);**

pack**();**

**}**// </editor-fold>

private void savejButtonActionPerformed**(**java**.**awt**.**event**.**ActionEvent evt**)** **{**

//ΕΜΦΑΝΙΣΗ ΠΑΡΑΘΥΡΟΥ ΕΠΙΒΕΒΑΙΩΣΗΣ ΑΠΟΘΗΚΕΥΣΗΣ

int response **=** JOptionPane**.**showConfirmDialog**(this,** "Θέλετε να αποθηκεύσετε "

**+** "τα δεδομένα για την χώρα " **+** country**.**getName**()** **+**"? \n "**,** "Αποθήκευση"**,**

JOptionPane**.**OK\_CANCEL\_OPTION**,** JOptionPane**.**QUESTION\_MESSAGE**);**

//ΕΝΕΡΓΕΙΕΣ ΜΕ ΒΑΣΗ ΤΗΝ ΕΠΙΛΟΓΗ ΤΟΥ ΧΡΗΣΤΗ

**if(**response **==** JOptionPane**.**OK\_OPTION**){** //ΕΠΙΛΟΓΗ ΟΚ

controller**.**insertDataToCountry**(**country**);**

controller**.**insertDatasetsToCountryDataset**(**cDatasets**);**

controller**.**insertDataToCountryData**(**allCountryData**);**

System**.**out**.**println**(**"<-- Επιτυχής αποθήκευση δεδομένων για την χώρα: " **+** country**.**getName**());**

**}else** **if** **(**response **==** JOptionPane**.**CANCEL\_OPTION**){** //ΕΠΙΛΟΓΗ CANCEL

**}else** **if** **(**response **==** JOptionPane**.**CLOSED\_OPTION**){** //ΕΠΙΛΟΓΗ ΕΞΟΔΟΥ JOPTION

**}**

//ΑΠΕΝΕΡΓΟΠΟΙΕΙ ΤΟ ΚΟΥΜΠΙ savejButton

savejButton**.**setEnabled**(false);**

//ΕΠΙΛΟΓΗ alreadySavedjCheckBox

alreadySavedjCheckBox**.**setSelected**(true);**

//ΕΝΕΡΓΟΠΟΙΗΣΗ ΚΟΥΜΠΙΟΥ deleteAlljButton

deleteAlljButton**.**setEnabled**(true);**

**}**

private void countryjComboBoxActionPerformed**(**java**.**awt**.**event**.**ActionEvent evt**)** **{**

//ΑΝ ΔΕΝ ΕΠΙΛΕΧΘΕΙ Η ΙΔΙΑ ΧΩΡΑ ΞΑΝΑ ΚΑΙ ΑΝ Η ΕΠΙΛΟΓΗ ΧΩΡΑΣ ΔΕΝ ΕΙΝΑΙ ΚΕΝΗ

**if(!**countryName**.**equals**(**countryjComboBox**.**getSelectedItem**().**toString**())** **&&** **!**countryName**.**equals**(**""**)){**

//ΕΝΕΡΓΟΠΟΙΗΣΗ fetchDatajButton

fetchDatajButton**.**setEnabled**(true);**

**}**

**}**

private void fetchDatajButtonMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**

// TODO add your handling code here:

**}**

private void countryjComboBoxItemStateChanged**(**java**.**awt**.**event**.**ItemEvent evt**)** **{**

// TODO add your handling code here:

**}**

private void alreadySavedjCheckBoxActionPerformed**(**java**.**awt**.**event**.**ActionEvent evt**)** **{**

// TODO add your handling code here:

**}**

private void deleteAlljButtonMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**

// TODO add your handling code here:

**}**

private void fetchDatajButtonActionPerformed**(**java**.**awt**.**event**.**ActionEvent evt**)** **{**

//ΚΑΤΑΧΩΡΗΣΗ ΕΠΙΛΟΓΗΣ countryjComboBox ΣΕ ΜΕΤΑΒΛΗΤΗ

countryName **=** countryjComboBox**.**getSelectedItem**().**toString**();**

//ΟΤΑΝ ΕΠΙΛΕΧΘΕΙ ΜΙΑ ΧΩΡΑ

**if(!**countryName**.**equals**(**"Choose a country"**)){**

//ΑΠΕΝΕΡΓΟΠΟΙΗΣΗ fetchDatajButton

fetchDatajButton**.**setEnabled**(false);**

//ΑΡΧΙΚΟΠΟΙΗΣΗ ΤΩΝ JTables - ΑΔΕΙΑΣΜΑ

oilDatajTable**.**setModel**(new** DefaultTableModel**(null,** **new** String**[]{**"Year"**,** "Value"**}));**

gdpDatajTable**.**setModel**(new** DefaultTableModel**(null,** **new** String**[]{**"Year"**,** "Value"**}));**

//ΒΡΙΣΚΟΥΜΕ ΤΟ ΑΝΤΙΣΤΟΙΧΟ Alpha3 isocode

isocode **=** controller**.**getIsocodeOfCountry**(**countryjComboBox**.**getSelectedItem**().**toString**());**

//ΔΗΜΙΟΥΡΓΙΑ ΛΙΣΤΩΝ ΓΙΑ ΧΡΗΣΗ

allCountryData **=** **new** ArrayList**<>();**

cDatasets **=** **new** ArrayList**<>();**

//ΔΗΜΙΟΥΡΓΙΑ ΑΝΤΙΚΕΙΜΕΝΩΝ ΤΥΠΟΥ DefaultTableModel

DefaultTableModel model\_oil **=** **(**DefaultTableModel**)**oilDatajTable**.**getModel**();**

DefaultTableModel model\_gdp **=** **(**DefaultTableModel**)**gdpDatajTable**.**getModel**();**

//ΔΗΜΙΟΥΡΓΙΑ ΑΝΤΙΚΕΙΜΕΝΟΥ Country

country **=** **new** Country**(**isocode**,** countryName**);**

//ΑΝ ΔΕΝ ΥΠΑΡΧΕΙ ΚΑΤΑΧΩΡΗΜΕΝΗ ΕΓΓΡΑΦΗ ΣΤΗ ΒΑΣΗ ΓΙΑ ΤΗΝ ΕΠΙΛΕΓΜΕΝΗ ΧΩΡΑ

**if(!**controller**.**FindInDbByIsocode**(**isocode**)){**

//ΔΗΜΙΟΥΡΓΙΑ RESTEndpoints

QuandlAPI call **=** **new** QuandlAPI**(**isocode**);**

String restEndpoint\_oil **=** call**.**OilConsumption\_quandlCall**();**

String restEndpoint\_gdp **=** call**.**GDP\_quandlCall**();**

String year**,** value**;**

//ΔΗΜΙΥΡΓΙΑ ΑΝΤΙΚΕΙΜΕΝΟΥ JSONREAD ΓΙΑ ΔΙΑΧΕΙΡΙΣΗ json string

JsonRead jsonread **=** **new** JsonRead**();**

int i**=**0**;**

//ΕΜΦΑΝΙΣΗ ΣΤΟΙΧΕΙΩΝ ΣΤΟ GUI ΓΙΑ OIL CONSUMPTION

//ΠΕΡΙΠΤΩΣΗ ΠΟΥ ΥΠΑΡΧΕΙ ΔΙΑΘΕΣΙΜΟ Dataset

**if(**restEndpoint\_oil **!=** **null){**

cDataset **=** jsonread**.**GetDataFromJsonAndSaveToDB**(**restEndpoint\_oil**,** country**);**

oilDatasetNamejlabel**.**setText**(**jsonread**.**datasetInfo**.**get**(**0**));**

oilStartDatejlabel**.**setText**(**jsonread**.**datasetInfo**.**get**(**1**));**

oilEndDatejlabel**.**setText**(**jsonread**.**datasetInfo**.**get**(**2**));**

**for(**i**=**0**;** i**<**jsonread**.**allData**.**size**();** i**++){**

year **=** jsonread**.**allData**.**get**(**i**).**getDataYear**();**

value **=** jsonread**.**allData**.**get**(**i**).**getValue**();**

model\_oil**.**addRow**(new** Object**[]{**year**,** value**});**

**}**

//ΠΡΟΣΘΗΚΗ DATASET ΣΤΗ ΛΙΣΤΑ ΤΩΝ CountryDatasets ΤΗΣ ΧΩΡΑΣ

cDatasets**.**add**(**cDataset**);**

**}** **else** **{** //ΠΕΡΙΠΤΩΣΗ ΠΟΥ ΔΕΝ ΥΠΑΡΧΕΙ ΔΙΑΘΕΣΙΜΟ Dataset

oilDatasetNamejlabel**.**setText**(**"Not available data."**);**

oilStartDatejlabel**.**setText**(**"-"**);**

oilEndDatejlabel**.**setText**(**"-"**);**

**}**

//ΕΜΦΑΝΙΣΗ ΣΤΟΙΧΕΙΩΝ ΣΤΟ GUI ΓΙΑ GDP

//ΠΕΡΙΠΤΩΣΗ ΠΟΥ ΥΠΑΡΧΕΙ ΔΙΑΘΕΣΙΜΟ Dataset

**if(**restEndpoint\_gdp **!=** **null){**

cDataset **=** jsonread**.**GetDataFromJsonAndSaveToDB**(**restEndpoint\_gdp**,** country**);**

gdpDatasetNamejlabel**.**setText**(**jsonread**.**datasetInfo**.**get**(**0**));**

gdpStartDatejlabel**.**setText**(**jsonread**.**datasetInfo**.**get**(**1**));**

gdpEndDatejlabel**.**setText**(**jsonread**.**datasetInfo**.**get**(**2**));**

**for(**int j**=**i**;** j**<**jsonread**.**allData**.**size**();** j**++){**

year **=** jsonread**.**allData**.**get**(**j**).**getDataYear**();**

value **=** jsonread**.**allData**.**get**(**j**).**getValue**();**

model\_gdp**.**addRow**(new** Object**[]{**year**,** value**});**

**}**

//ΠΡΟΣΘΗΚΗ DATASET ΣΤΗ ΛΙΣΤΑ ΤΩΝ CountryDatasets ΤΗΣ ΧΩΡΑΣ

cDatasets**.**add**(**cDataset**);**

**}** **else** **{** //ΠΕΡΙΠΤΩΣΗ ΠΟΥ ΔΕΝ ΥΠΑΡΧΕΙ ΔΙΑΘΕΣΙΜΟ Dataset

gdpDatasetNamejlabel**.**setText**(**"Not available data."**);**

gdpStartDatejlabel**.**setText**(**"-"**);**

gdpEndDatejlabel**.**setText**(**"-"**);**

**}**

//ΚΑΤΑΧΩΡΗΣΗ ΟΛΩΝ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΓΙΑ OIL CONSUMPTIOΝ KAI GDP ΣΕ

//ΤΟΠΙΚΗ ΛΙΣΤΑ ΜΕ ΣΚΟΠΟ ΤΗ ΧΡΗΣΗ ΤΗΣ ΓΙΑ ΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΤΟΥΣ

allCountryData **=** jsonread**.**allData**;**

//ΕΝΕΡΓΟΠΟΙΕΙ ΤΟ ΚΟΥΜΠΙ savejButton ΚΑΙ ΤΟ ΚΟΥΜΠΙ plotjButton

//ΜΟΝΟ ΑΝ ΥΠΑΡΧΕΙ ΤΟΥΛΑΧΙΣΤΟΝ ΕΝΑ ΔΙΑΘΕΣΙΜΟ DATASET

**if(**restEndpoint\_oil **!=** **null** **||** restEndpoint\_gdp **!=** **null){**

savejButton**.**setEnabled**(true);**

plotjButton**.**setEnabled**(true);**

**}else{**

savejButton**.**setEnabled**(false);**

plotjButton**.**setEnabled**(false);**

**}**

//ΑΠΕΠΙΛΟΓΗ alreadySavedjCheckBox

alreadySavedjCheckBox**.**setSelected**(false);**

**}else{** //ΑΝ ΥΠΑΡΧΟΥΝ ΗΔΗ ΔΕΔΟΜΕΝΑ ΣΤΗ ΒΑΣΗ

//ΑΠΕΝΕΡΓΟΠΟΙΕΙ ΤΟ ΚΟΥΜΠΙ savejButton

savejButton**.**setEnabled**(false);**

//ΕΝΕΡΓΟΠΟΙΕΙ ΤΟ ΚΟΥΜΠΙ plotjButton

plotjButton**.**setEnabled**(true);**

//ΣΗΜΕΙΩΣΗ ΕΝΔΕΙΞΗΣ ΓΙΑ ΗΔΗ ΑΠΟΘΗΚΕΥΜΕΝΑ ΔΕΔΟΜΕΝΑ ΣΤΗ ΒΑΣΗ

alreadySavedjCheckBox**.**setSelected**(true);**

//ΑΡΧΙΚΟΠΟΙΗΣΗ ΤΩΝ labels ΤΩΝ datasets

oilDatasetNamejlabel**.**setText**(**"-"**);**

oilStartDatejlabel**.**setText**(**"-"**);**

oilEndDatejlabel**.**setText**(**"-"**);**

System**.**out**.**println**(**"--> Επιτυχής ανάκτηση δεδομένων από τη βάση για την χώρα: "**+** country**.**getName**());**

//ΕΥΡΕΣΗ Datasets ΜΕ ΒΑΣΗ ΤΟ country ΑΠΟ ΤΟΝ ΠΙΝΑΚΑ COUNTRY\_DATASET

cDatasets **=** controller**.**FindCountryDatasets**(**country**);**

//ΛΙΣΤΑ ΓΙΑ ΤΗΡΗΣΗ ΟΛΩΝ ΤΩΝ ΑΝΤΙΚΕΙΜΕΝΩΝ CountryData ΣΥΓΚΕΝΤΡΩΤΙΚΑ

//ΓΙΑ ΟΛΑ ΤΑ ΑΠΟΘΗΚΕΥΜΕΝΑ DATASETS

allCountryData **=** **new** ArrayList**<>();**

//ΓΙΑ ΚΑΘΕ dataset ΕΜΦΑΝΙΖΕΙ ΤΑ ΣΤΟΙΧΕΙΑ ΤΟΥ

**for** **(**int i **=** 0**;** i**<**cDatasets**.**size**();** i**++){**

//ΛΙΣΤΑ ΑΝΤΙΚΕΙΜΕΝΩΝ CountryData ΓΙΑ ΤΟ ΚΑΘΕ DATASET

List**<**CountryData**>** allCountryData1**;**

**if(**cDatasets**.**get**(**i**).**getName**().**startsWith**(**"Oil"**)){**

oilDatasetNamejlabel**.**setText**(**cDatasets**.**get**(**i**).**getName**());**

oilStartDatejlabel**.**setText**(**cDatasets**.**get**(**i**).**getStartYear**()** **+** "-12-31"**);**

oilEndDatejlabel**.**setText**(**cDatasets**.**get**(**i**).**getEndYear**()** **+** "-12-31"**);**

//ΓΙΑ ΚΑΘΕ data record ΤΟΥ oilDataset ΕΜΦΑΝΙΖΕΙ ΤΑ ΣΤΟΙΧΕΙΑ ΤΟΥ

allCountryData1 **=** controller**.**FindCountryData**(**cDatasets**.**get**(**i**));**

**for** **(**int j **=** 0**;** j**<**allCountryData1**.**size**();** j**++){**

model\_oil**.**addRow**(new** Object**[]{**allCountryData1**.**get**(**j**).**getDataYear**(),** allCountryData1**.**get**(**j**).**getValue**()});**

allCountryData**.**add**(**allCountryData1**.**get**(**j**));**

**}**

**}else{**

gdpDatasetNamejlabel**.**setText**(**cDatasets**.**get**(**i**).**getName**());**

gdpStartDatejlabel**.**setText**(**cDatasets**.**get**(**i**).**getStartYear**()** **+** "-12-31"**);**

gdpEndDatejlabel**.**setText**(**cDatasets**.**get**(**i**).**getEndYear**()** **+** "-12-31"**);**

//ΓΙΑ ΚΑΘΕ data record ΤΟΥ gdpDataset ΕΜΦΑΝΙΖΕΙ ΤΑ ΣΤΟΙΧΕΙΑ ΤΟΥ

allCountryData1 **=** controller**.**FindCountryData**(**cDatasets**.**get**(**i**));**

**for** **(**int j **=** 0**;** j**<**allCountryData1**.**size**();** j**++){**

model\_gdp**.**addRow**(new** Object**[]{**allCountryData1**.**get**(**j**).**getDataYear**(),** allCountryData1**.**get**(**j**).**getValue**()});**

allCountryData**.**add**(**allCountryData1**.**get**(**j**));**

**}**

**}**

**}**

//ΑΝ ΔΕΝ ΕΧΟΥΝ ΒΡΕΘΕΙ ΤΑ ΑΝΤΙΣΤΟΙΧΑ datasets

**if(**oilDatasetNamejlabel**.**getText**().**equals**(**"-"**)){**

oilDatasetNamejlabel**.**setText**(**"Not available data."**);**

oilStartDatejlabel**.**setText**(**"-"**);**

oilEndDatejlabel**.**setText**(**"-"**);**

**}**

**if(**gdpDatasetNamejlabel**.**getText**().**equals**(**"-"**)){**

gdpDatasetNamejlabel**.**setText**(**"Not available data."**);**

gdpStartDatejlabel**.**setText**(**"-"**);**

gdpEndDatejlabel**.**setText**(**"-"**);**

**}**

**}**

**}else{** //ΑΝ ΔΕΝ ΕΧΕΙ ΕΠΙΛΕΧΘΕΙ ΧΩΡΑ

JOptionPane**.**showMessageDialog**(this,** "Επιλέξτε πρώτα μία χώρα."**,**

"Σφάλμα"**,** JOptionPane**.**ERROR\_MESSAGE**);**

**}**

//ΚΕΝΤΡΙΚΗ ΣΤΟΙΧΙΣΗ JTABLES

**for(**int i**=**0**;** i**<**2**;** i**++){**

oilDatajTable**.**getColumnModel**().**getColumn**(**i**).**setHeaderRenderer**(**centerRenderer**);**

oilDatajTable**.**getColumnModel**().**getColumn**(**i**).**setCellRenderer**(**centerRenderer**);**

gdpDatajTable**.**getColumnModel**().**getColumn**(**i**).**setHeaderRenderer**(**centerRenderer**);**

gdpDatajTable**.**getColumnModel**().**getColumn**(**i**).**setCellRenderer**(**centerRenderer**);**

**}**

**}**

private void deleteAlljButtonActionPerformed**(**java**.**awt**.**event**.**ActionEvent evt**)** **{**

//ΕΜΦΑΝΙΣΗ ΠΑΡΑΘΥΡΟΥ ΕΠΙΒΕΒΑΙΩΣΗΣ ΔΙΑΓΡΑΦΗΣ ΔΕΔΟΜΕΝΩΝ ΒΑΣΗΣ

int response **=** JOptionPane**.**showConfirmDialog**(this,** "Θέλετε να διαγράψετε όλα "

**+** "τα δεδομένα από τη βάση? \n "**,** "Διαγραφή δεδομένων"**,**

JOptionPane**.**OK\_CANCEL\_OPTION**,** JOptionPane**.**QUESTION\_MESSAGE**);**

//ΕΝΕΡΓΕΙΕΣ ΜΕ ΒΑΣΗ ΤΗΝ ΕΠΙΛΟΓΗ ΤΟΥ ΧΡΗΣΤΗ

**if(**response **==** JOptionPane**.**OK\_OPTION**){** //ΕΠΙΛΟΓΗ ΟΚ

controller**.**clearTables**();**

System**.**out**.**println**(**"------------------------------------------------"

**+** "----------------------------"**);**

System**.**out**.**println**(**"ΔΙΑΓΡΑΦΗ ΟΛΩΝ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΤΟΥΣ ΠΙΝΑΚΕΣ - "

**+** "Η ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ ΕΙΝΑΙ ΑΔΕΙΑ!"**);**

System**.**out**.**println**(**"------------------------------------------------"

**+** "----------------------------"**);**

JOptionPane**.**showMessageDialog**(this,** "Όλα τα δεδομένα διαγράφηκαν από "

**+** "τη βάση.\nΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ ΕΙΝΑΙ ΑΔΕΙΑ!"**,** "Ενημέρωση"**,** JOptionPane**.**INFORMATION\_MESSAGE**);**

//ΑΠΕΝΕΡΓΟΠΟΙΗΣΗ ΚΟΥΜΠΙΟΥ deleteAlljButton

deleteAlljButton**.**setEnabled**(false);**

//ΑΝΑΝΕΩΣΗ ΔΕΔΟΜΕΝΩΝ

**if(!**countryName**.**equals**(**"Choose a country"**)** **&&** **!**countryName**.**equals**(**""**)){**

fetchDatajButtonActionPerformed**(**evt**);**

**}**

**}else** **if** **(**response **==** JOptionPane**.**CANCEL\_OPTION**){** //ΕΠΙΛΟΓΗ CANCEL

**}else** **if** **(**response **==** JOptionPane**.**CLOSED\_OPTION**){** //ΕΠΙΛΟΓΗ ΕΞΟΔΟΥ JOPTION

**}**

**}**

private void plotjButtonActionPerformed**(**java**.**awt**.**event**.**ActionEvent evt**)** **{**

//ΔΗΜΙΟΥΡΓΙΑ ΑΝΤΙΚΕΙΜΕΝΟΥ PlotMaker ΚΑΙ ΕΜΦΑΝΙΣΗ ΤΟΥ ΣΤΟ ΚΕΝΤΡΟ

plot **=** **new** PlotMaker**(**"Economic Data for " **+**

country**.**getName**(),** allCountryData**,** cDatasets**);**

plot**.**pack**();**

RefineryUtilities**.**centerFrameOnScreen**(**plot**);**

plot**.**setVisible**(true);**

**}**

private void formWindowActivated**(**java**.**awt**.**event**.**WindowEvent evt**)** **{**

// TODO add your handling code here:

**}**

private void formWindowGainedFocus**(**java**.**awt**.**event**.**WindowEvent evt**)** **{**

//ΚΛΕΙΣΙΜΟ ΔΙΑΓΡΑΜΜΑΤΟΣ ΟΤΑΝ ΕΣΤΙΑΖΟΥΜΕ ΣΤΗΝ ΚΕΝΤΡΙΚΗ ΦΟΡΜΑ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

**if(**plot**!=null)** **{**plot**.**dispose**();}**

**}**

//ΔΙΑΔΙΚΑΣΙΑ ΕΡΩΤΗΣΗΣ ΤΟΥ ΧΡΗΣΤΗ ΚΑΤΑ ΤΗΝ ΕΠΙΛΟΓΗ ΕΞΟΔΟΥ ΑΠΟ ΤΗΝ ΕΦΑΡΜΟΓΗ

private void ExitOnCloseOption **()** **{**

Object**[]** options **=** **{**"OK"**,** "Άκυρο"**};**

int result **=** JOptionPane**.**showOptionDialog**(this,**

"Θέλετε να γίνει τερματισμός της εφαρμογής;"**,** "Έξοδος"**,**

JOptionPane**.**YES\_NO\_OPTION**,** JOptionPane**.**QUESTION\_MESSAGE**,null,**options**,**options**[**0**]);**

**if** **(**result **==** JOptionPane**.**YES\_OPTION**)**

System**.**exit**(**0**);**

**}**

/\*\*

\* **@param** args the command line arguments

\*/

public static void main**(**String args**[])** **{**

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

**try** **{**

**for** **(**javax**.**swing**.**UIManager**.**LookAndFeelInfo info **:** javax**.**swing**.**UIManager**.**getInstalledLookAndFeels**())** **{**

**if** **(**"Nimbus"**.**equals**(**info**.**getName**()))** **{**

javax**.**swing**.**UIManager**.**setLookAndFeel**(**info**.**getClassName**());**

**break;**

**}**

**}**

**}** **catch** **(**ClassNotFoundException ex**)** **{**

java**.**util**.**logging**.**Logger**.**getLogger**(**EconometricaGUI**.**class**.**getName**()).**log**(**java**.**util**.**logging**.**Level**.**SEVERE**,** **null,** ex**);**

**}** **catch** **(**InstantiationException ex**)** **{**

java**.**util**.**logging**.**Logger**.**getLogger**(**EconometricaGUI**.**class**.**getName**()).**log**(**java**.**util**.**logging**.**Level**.**SEVERE**,** **null,** ex**);**

**}** **catch** **(**IllegalAccessException ex**)** **{**

java**.**util**.**logging**.**Logger**.**getLogger**(**EconometricaGUI**.**class**.**getName**()).**log**(**java**.**util**.**logging**.**Level**.**SEVERE**,** **null,** ex**);**

**}** **catch** **(**javax**.**swing**.**UnsupportedLookAndFeelException ex**)** **{**

java**.**util**.**logging**.**Logger**.**getLogger**(**EconometricaGUI**.**class**.**getName**()).**log**(**java**.**util**.**logging**.**Level**.**SEVERE**,** **null,** ex**);**

**}**

//</editor-fold>

/\* Create and display the form \*/

java**.**awt**.**EventQueue**.**invokeLater**(new** Runnable**()** **{**

@Override

public void run**()** **{**

EconometricaGUI econ **=** **new** EconometricaGUI**();**

//ΕΜΦΑΝΙΣΗ ΠΑΡΑΘΥΡΟΥ ΣΤΟ ΚΕΝΤΡΟ

RefineryUtilities**.**centerFrameOnScreen**(**econ**);**

econ**.**setVisible**(true);**

**}**

**});**

**}**

// Variables declaration - do not modify

private javax**.**swing**.**JCheckBox alreadySavedjCheckBox**;**

private javax**.**swing**.**JLabel avTimespanjLabel**;**

private javax**.**swing**.**JLabel avTimespanjLabel1**;**

private javax**.**swing**.**JLabel countrySelectjLabel**;**

private javax**.**swing**.**JComboBox**<**String**>** countryjComboBox**;**

private javax**.**swing**.**JButton deleteAlljButton**;**

private javax**.**swing**.**JButton fetchDatajButton**;**

private javax**.**swing**.**JLabel gdpDatajLabel**;**

private javax**.**swing**.**JTable gdpDatajTable**;**

private javax**.**swing**.**JLabel gdpDatasetNameTitlejLabel**;**

private javax**.**swing**.**JLabel gdpDatasetNamejlabel**;**

private javax**.**swing**.**JLabel gdpEndDateTitlejLabel**;**

private javax**.**swing**.**JLabel gdpEndDatejlabel**;**

private javax**.**swing**.**JLabel gdpStartDateTitlejLabel**;**

private javax**.**swing**.**JLabel gdpStartDatejlabel**;**

private javax**.**swing**.**JScrollPane jScrollPane3**;**

private javax**.**swing**.**JScrollPane jScrollPane4**;**

private javax**.**swing**.**JLabel oilDatajLabel**;**

private javax**.**swing**.**JTable oilDatajTable**;**

private javax**.**swing**.**JLabel oilDatasetNameTitlejLabel**;**

private javax**.**swing**.**JLabel oilDatasetNamejlabel**;**

private javax**.**swing**.**JLabel oilEndDateTitlejLabel**;**

private javax**.**swing**.**JLabel oilEndDatejlabel**;**

private javax**.**swing**.**JLabel oilStartDateTitlejLabel**;**

private javax**.**swing**.**JLabel oilStartDatejlabel**;**

private javax**.**swing**.**JButton plotjButton**;**

private javax**.**swing**.**JButton savejButton**;**

// End of variables declaration

**}**

## Παρουσίαση Αποτελεσμάτων / Στατιστικών Στοιχείων

PlotMaker

Η κλάση PlotMaker, δημιουργεί τα 2 dataset για το oil consumption και το GDP και εμφανίζει το διάγραμμα. Η κλάση αναφέρεται και στο υποερώτημα 3.1 β όπου ζητούνται όλες οι κλάσεις

package econometrica**;**

**import** databaseClasses**.\*;**

**import** org**.**jfree**.**ui**.**ApplicationFrame**;**

**import** java**.**awt**.**Color**;**

**import** java**.**text**.**NumberFormat**;**

**import** java**.**text**.**ParseException**;**

**import** java**.**text**.**SimpleDateFormat**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Date**;**

**import** java**.**util**.**List**;**

**import** java**.**util**.**Locale**;**

**import** java**.**util**.**logging**.**Level**;**

**import** java**.**util**.**logging**.**Logger**;**

**import** org**.**jfree**.**chart**.**ChartFactory**;**

**import** org**.**jfree**.**chart**.**ChartPanel**;**

**import** org**.**jfree**.**chart**.**JFreeChart**;**

**import** org**.**jfree**.**chart**.**axis**.**DateAxis**;**

**import** org**.**jfree**.**chart**.**axis**.**NumberAxis**;**

**import** org**.**jfree**.**chart**.**labels**.**StandardXYToolTipGenerator**;**

**import** org**.**jfree**.**chart**.**plot**.**XYPlot**;**

**import** org**.**jfree**.**chart**.**renderer**.**xy**.**StandardXYItemRenderer**;**

**import** org**.**jfree**.**chart**.**renderer**.**xy**.**XYItemRenderer**;**

**import** org**.**jfree**.**data**.**time**.**TimeSeries**;**

**import** org**.**jfree**.**data**.**time**.**TimeSeriesCollection**;**

**import** org**.**jfree**.**data**.**time**.**Year**;**

**import** org**.**jfree**.**data**.**xy**.**XYDataset**;**

public class PlotMaker **extends** ApplicationFrame**{**

//ΛΙΣΤΕΣ ΓΙΑ ΔΙΑΧΩΡΙΣΜΟ ΤΩΝ ΑΝΤΙΚΕΙΜΕΝΩΝ ΤΩΝ 2 DATASETS

private static List**<**CountryData**>** datalist1**,** datalist2**;**

static SimpleDateFormat dateformat **=** **new** SimpleDateFormat**(**"yyyy"**);**

final ChartPanel graphPanel**;**

//CONSTRUCTOR

public PlotMaker**(**String title**,** List**<**CountryData**>** alldata**,** List**<**CountryDataset**>** datasetList**){**

**super(**title**);**

final String chartTitle **=** title**;**

datalist1 **=** **new** ArrayList**<>();**

datalist2 **=** **new** ArrayList**<>();**

//ΔΙΑΧΩΡΙΣΜΟΣ ΑΝΤΙΚΕΙΜΕΝΩΝ Country\_Data ΓΙΑ ΤΟ ΚΑΘΕ DATASET ΑΠΟ ΤΗ ΛΙΣΤΑ

//alldata Η ΟΠΟΙΑ ΠΕΡΙΕΧΕΙ ΟΛΑ ΤΑ Country\_Data ΤΗΣ ΧΩΡΑΣ

**for** **(**int i **=** 0**;** i**<**alldata**.**size**();** i**++){**

**if(**alldata**.**get**(**i**).**getDataset**().**getName**().**startsWith**(**"Oil"**)){**

datalist1**.**add**(**alldata**.**get**(**i**));**

**}else{**

datalist2**.**add**(**alldata**.**get**(**i**));**

**}**

**}**

final XYDataset dataset **=** createDataset1**();**

final JFreeChart graph **=** ChartFactory**.**createTimeSeriesChart**(**

chartTitle**,**

"Year"**,**

"GDP"**,**

dataset**,**

**true,**

**true,**

**false);**

final XYPlot plot **=** graph**.**getXYPlot**();**

final NumberAxis axis2 **=** **new** NumberAxis**(**"Oil Consumption"**);**

axis2**.**setAutoRangeIncludesZero**(false);**

plot**.**setRangeAxis**(**1**,** axis2**);**

plot**.**setDataset**(**1**,** createDataset2**());**

plot**.**mapDatasetToRangeAxis**(**1**,** 1**);**

final XYItemRenderer renderer **=** plot**.**getRenderer**();**

renderer**.**setToolTipGenerator**(**StandardXYToolTipGenerator**.**getTimeSeriesInstance**());**

**if** **(**renderer **instanceof** StandardXYItemRenderer**)** **{**

final StandardXYItemRenderer rr **=** **(**StandardXYItemRenderer**)** renderer**;**

rr**.**setShapesFilled**(true);**

**}**

final StandardXYItemRenderer renderer2 **=** **new** StandardXYItemRenderer**();**

renderer2**.**setSeriesPaint**(**0**,** Color**.**BLUE**);**

renderer**.**setToolTipGenerator**(**StandardXYToolTipGenerator**.**getTimeSeriesInstance**());**

plot**.**setRenderer**(**1**,** renderer2**);**

final DateAxis axis **=** **(**DateAxis**)**plot**.**getDomainAxis**();**

axis**.**setDateFormatOverride**(**dateformat**);**

graphPanel **=** **new** ChartPanel**(**graph**);**

graphPanel**.**setPreferredSize**(new** java**.**awt**.**Dimension**(**1500**,** 800**));**

setContentPane**(**graphPanel**);**

**}**

//ΜΕΘΟΔΟΣ ΔΗΜΙΟΥΡΓΙΑΣ XYDataset

private static XYDataset createDataset1**(){**

Date year **=** **null;**

Number value **=** **null;**

final TimeSeries s1 **=** **new** TimeSeries**(**"Gross Domestic Product (GDP)"**,** Year**.**class**);**

//NumberFormat ΓΙΑ ΑΝΤΙΜΕΤΩΠΙΣΗ ΤΟΥ ΚΟΜΜΑ ΣΤΟΝ ΑΡΙΘΜΟ

NumberFormat nf **=** NumberFormat**.**getInstance**(**Locale**.**ENGLISH**);**

//ΓΙΑ ΚΑΘΕ ΑΝΤΙΚΕΙΜΕΝΟ Country\_Data ΤΟΥ dataset1 ΧΡΗΣΙΜΟΠΟΙΟΥΜΕ ΤΙΜΗ ΚΑΙ ΕΤΟΣ

**for** **(**int i **=** 0**;** i**<**datalist2**.**size**();** i**++){**

**try** **{**

value **=** nf**.**parse**(**datalist2**.**get**(**i**).**getValue**());**

year **=** dateformat**.**parse**(**datalist2**.**get**(**i**).**getDataYear**());**

**}** **catch** **(**ParseException ex**)** **{**

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

**}**

s1**.**add**(new** Year**(**year**),** value**);**

**}**

final TimeSeriesCollection dataset **=** **new** TimeSeriesCollection**();**

dataset**.**addSeries**(**s1**);**

**return** dataset**;**

**}**

//ΜΕΘΟΔΟΣ ΔΗΜΙΟΥΡΓΙΑΣ XYDataset

private XYDataset createDataset2**()** **{**

Date year **=** **null;**

Number value **=** **null;**

final TimeSeries s2 **=** **new** TimeSeries**(**"Oil Consumption"**,** Year**.**class**);**

//NumberFormat ΓΙΑ ΑΝΤΙΜΕΤΩΠΙΣΗ ΤΟΥ ΚΟΜΜΑ ΣΤΟΝ ΑΡΙΘΜΟ

NumberFormat nf **=** NumberFormat**.**getInstance**(**Locale**.**ENGLISH**);**

**for** **(**int i **=** 0**;** i**<**datalist1**.**size**();** i**++){**

**try** **{**

value **=** nf**.**parse**(**datalist1**.**get**(**i**).**getValue**());**

year **=** dateformat**.**parse**(**datalist1**.**get**(**i**).**getDataYear**());**

**}** **catch** **(**ParseException ex**)** **{**

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

**}**

s2**.**add**(new** Year**(**year**),** value**);**

**}**

final TimeSeriesCollection dataset **=** **new** TimeSeriesCollection**();**

dataset**.**addSeries**(**s2**);**

**return** dataset**;**

**}**

**}**

**Plotjbutton**

Μέθοδοι που μπήκαν στην κλάση EcometricaGui**.** Οι μέθοδοι παρακάτω είναι διάσπαρτες. Υπάρχουν στο σύνολο τους στην υλοποίηση του GUI.

plotjButton**.**setText**(**"Plot"**);**

plotjButton**.**setToolTipText**(**"Προβολή διαγράμματος"**);**

plotjButton**.**setEnabled**(false);**

plotjButton**.**addActionListener**(new** java**.**awt**.**event**.**ActionListener**()** **{**

public void actionPerformed**(**java**.**awt**.**event**.**ActionEvent evt**)** **{**

plotjButtonActionPerformed**(**evt**);**

//ΚΑΤΑΧΩΡΗΣΗ ΟΛΩΝ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΓΙΑ OIL CONSUMPTIOΝ KAI GDP ΣΕ

//ΤΟΠΙΚΗ ΛΙΣΤΑ ΜΕ ΣΚΟΠΟ ΤΗ ΧΡΗΣΗ ΤΗΣ ΓΙΑ ΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΤΟΥΣ

allCountryData **=** jsonread**.**allData**;**

//ΕΝΕΡΓΟΠΟΙΕΙ ΤΟ ΚΟΥΜΠΙ savejButton ΚΑΙ ΤΟ ΚΟΥΜΠΙ plotjButton

//ΜΟΝΟ ΑΝ ΥΠΑΡΧΕΙ ΤΟΥΛΑΧΙΣΤΟΝ ΕΝΑ ΔΙΑΘΕΣΙΜΟ DATASET

**if(**restEndpoint\_oil **!=** **null** **||** restEndpoint\_gdp **!=** **null){**

savejButton**.**setEnabled**(true);**

plotjButton**.**setEnabled**(true);**

**}else{**

savejButton**.**setEnabled**(false);**

plotjButton**.**setEnabled**(false);**

**}**

//ΑΠΕΠΙΛΟΓΗ alreadySavedjCheckBox

alreadySavedjCheckBox**.**setSelected**(false);**

**}else{** //ΑΝ ΥΠΑΡΧΟΥΝ ΗΔΗ ΔΕΔΟΜΕΝΑ ΣΤΗ ΒΑΣΗ

//ΑΠΕΝΕΡΓΟΠΟΙΕΙ ΤΟ ΚΟΥΜΠΙ savejButton

savejButton**.**setEnabled**(false);**

//ΕΝΕΡΓΟΠΟΙΕΙ ΤΟ ΚΟΥΜΠΙ plotjButton

plotjButton**.**setEnabled**(true);**

//ΣΗΜΕΙΩΣΗ ΕΝΔΕΙΞΗΣ ΓΙΑ ΗΔΗ ΑΠΟΘΗΚΕΥΜΕΝΑ ΔΕΔΟΜΕΝΑ ΣΤΗ ΒΑΣΗ

alreadySavedjCheckBox**.**setSelected**(true);**

//ΑΡΧΙΚΟΠΟΙΗΣΗ ΤΩΝ labels ΤΩΝ datasets

oilDatasetNamejlabel**.**setText**(**"-"**);**

oilStartDatejlabel**.**setText**(**"-"**);**

oilEndDatejlabel**.**setText**(**"-"**);**

System**.**out**.**println**(**"--> Επιτυχής ανάκτηση δεδομένων από τη βάση για την χώρα: "**+** country**.**getName**());**

//ΚΑΤΑΧΩΡΗΣΗ ΟΛΩΝ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΓΙΑ OIL CONSUMPTIOΝ KAI GDP ΣΕ

//ΤΟΠΙΚΗ ΛΙΣΤΑ ΜΕ ΣΚΟΠΟ ΤΗ ΧΡΗΣΗ ΤΗΣ ΓΙΑ ΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΤΟΥΣ

allCountryData **=** jsonread**.**allData**;**

//ΕΝΕΡΓΟΠΟΙΕΙ ΤΟ ΚΟΥΜΠΙ savejButton ΚΑΙ ΤΟ ΚΟΥΜΠΙ plotjButton

//ΜΟΝΟ ΑΝ ΥΠΑΡΧΕΙ ΤΟΥΛΑΧΙΣΤΟΝ ΕΝΑ ΔΙΑΘΕΣΙΜΟ DATASET

**if(**restEndpoint\_oil **!=** **null** **||** restEndpoint\_gdp **!=** **null){**

savejButton**.**setEnabled**(true);**

plotjButton**.**setEnabled**(true);**

**}else{**

savejButton**.**setEnabled**(false);**

plotjButton**.**setEnabled**(false);**

**}**

//ΑΠΕΠΙΛΟΓΗ alreadySavedjCheckBox

alreadySavedjCheckBox**.**setSelected**(false);**

**}else{** //ΑΝ ΥΠΑΡΧΟΥΝ ΗΔΗ ΔΕΔΟΜΕΝΑ ΣΤΗ ΒΑΣΗ

//ΑΠΕΝΕΡΓΟΠΟΙΕΙ ΤΟ ΚΟΥΜΠΙ savejButton

savejButton**.**setEnabled**(false);**

//ΕΝΕΡΓΟΠΟΙΕΙ ΤΟ ΚΟΥΜΠΙ plotjButton

plotjButton**.**setEnabled**(true);**

//ΣΗΜΕΙΩΣΗ ΕΝΔΕΙΞΗΣ ΓΙΑ ΗΔΗ ΑΠΟΘΗΚΕΥΜΕΝΑ ΔΕΔΟΜΕΝΑ ΣΤΗ ΒΑΣΗ

alreadySavedjCheckBox**.**setSelected**(true);**

//ΑΡΧΙΚΟΠΟΙΗΣΗ ΤΩΝ labels ΤΩΝ datasets

oilDatasetNamejlabel**.**setText**(**"-"**);**

oilStartDatejlabel**.**setText**(**"-"**);**

oilEndDatejlabel**.**setText**(**"-"**);**

System**.**out**.**println**(**"--> Επιτυχής ανάκτηση δεδομένων από τη βάση για την χώρα: "**+** country**.**getName**());**

## Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής

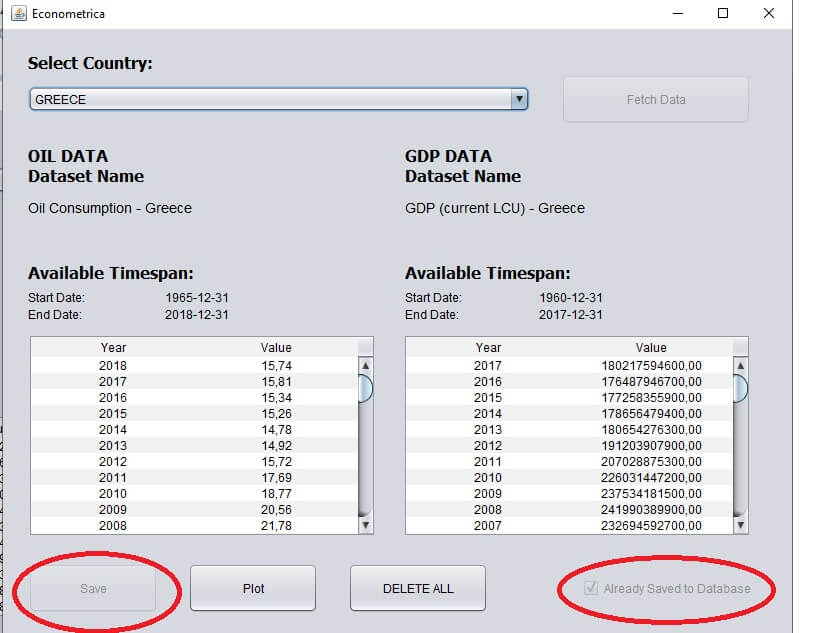
|  |  |
| --- | --- |
| **Απαίτηση** | **Εικόνα** |
| R1 | Στην πρώτη εικόνα βλέπουμε μια γενική άποψη του συστήματος  1  Επίσης την λειτουργία της απαίτησης R1 για την Drop Down Menu  2 |
| R2 | Στις εικόνες που ακολουθούν παρουσιάζεται η λειτουργικότητα της απαίτησης R2 που αφορά την λειτουργία του κουμπιού **«FETCH DATA»** . Με την επιλογή του πλήκτρου **«FETCH DATA»**  το σύστημα ελέγχει αν για την επιλεγμένη χώρα υπάρχουν διαθέσιμα τα δεδομένα στην ΒΔ. Εμφανίζει τα δεδομένα στον αριστερό και δεξιό πίνακα. Αν τα δεδομένα ήταν ήδη αποθηκευμένα, το σύστημα τα διαβάζει από την ΒΔ και σημειώνει τη σχετική ένδειξη  **Already Saved to Database.** Επίσης παρατηρούμε ότι όταν πλέον οι πίνακες έχουν στοιχεία το πλήκτρο **«FETCH DATA»**  απενεργοποιείται.  3  4 |
| R3 | Παρουσίαση της απαίτησης R3, αναφέρεται στην λειτουργία του πλήκτρου **“SAVE”** . Με την επιλογή του πλήκτρου **“SAVE”**  το σύστημα σώζει στην ΒΔ και σημειώνει την σχετική ένδειξη:  **Already Saved to Database.** Όταν η ένδειξη Already Saved to Database είναι ενεργοποιημένη τότε το κουμπί SAVE είναι απενεργοποιημένο.  3  4 |
| R4 | Παρουσίαση της απαίτησης R4, για την λειτουργία του πλήκτρου **“DELETE ALL”**. Με την επιλογή του πλήκτρου DELETE ALL, το σύστημα διαγράφει όλα τα δεδομένα από όλους τους πίνακες της βάσης, αφού προηγηθεί ερώτημα επαλήθευσης με επιβεβαίωση από την χρήστη.  5  6 |
| R5 | Παρουσίαση της απαίτησης R5, αφορά την λειτουργία του πλήκτρου **“PLOT”.** Με την επιλογή του πλήκτρου Plot το σύστημα εμφανίζει σε αντιπαραβολή (στο ίδιο διάγραμμα δηλαδή για να είναι δυνατή η σύγκριση) σε ξεχωριστό παράθυρο τις 2 χρονοσειρές. Στις εικόνες παρακάτω παρουσιάζονται 4 ξεχωριστές περιπτώσεις. Η πρώτη περίπτωση αφορά χώρα που εμφανίζει και τα δύο δεδομένα, oil data και GDP. Στην δεύτερη περίπτωση αφορά χώρα που έχει δεδομένα μόνο στον πίνακα oil data, στην Τρίτη περίπτωση αφορά χώρα που έχει δεδομένα μόνο στον πίνακα GDP και τέλος στην τελευταία περίπτωση χώρα που δεν έχει καθόλου δεδομένα.  13  14  9  10  11  12  8 |

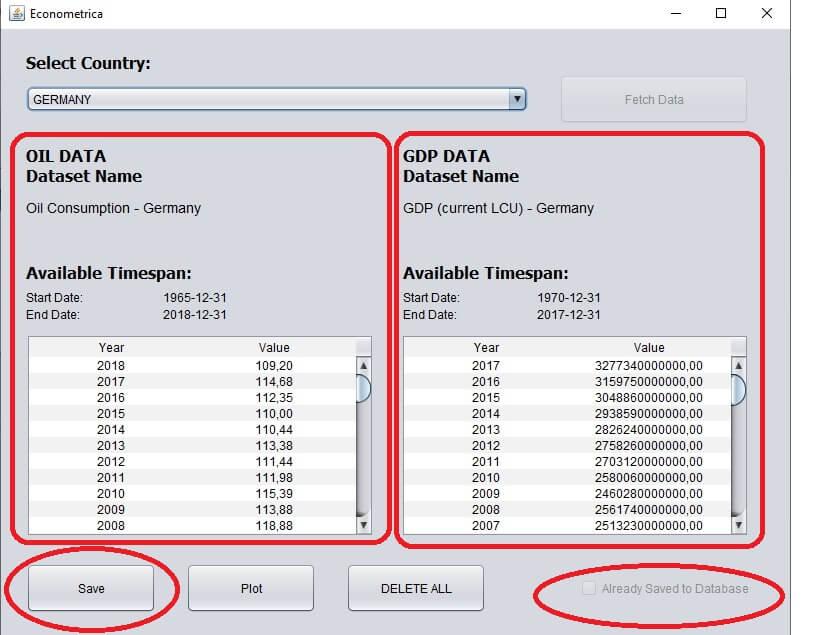
# Τέλος θα ήθελα να αναφερθώ σε μερικές ακόμα λειτουργικότητες που έχουμε προσθέσει στην εφαρμογή για καλύτερη λειτουργικότητα και για να είναι περισσότερο φιλική προς τον τελικό χρήστη.

# Το πλήκτρο “Delete All” είναι απενεργοποιημένο όταν η βάση δεδομένων είναι άδεια.

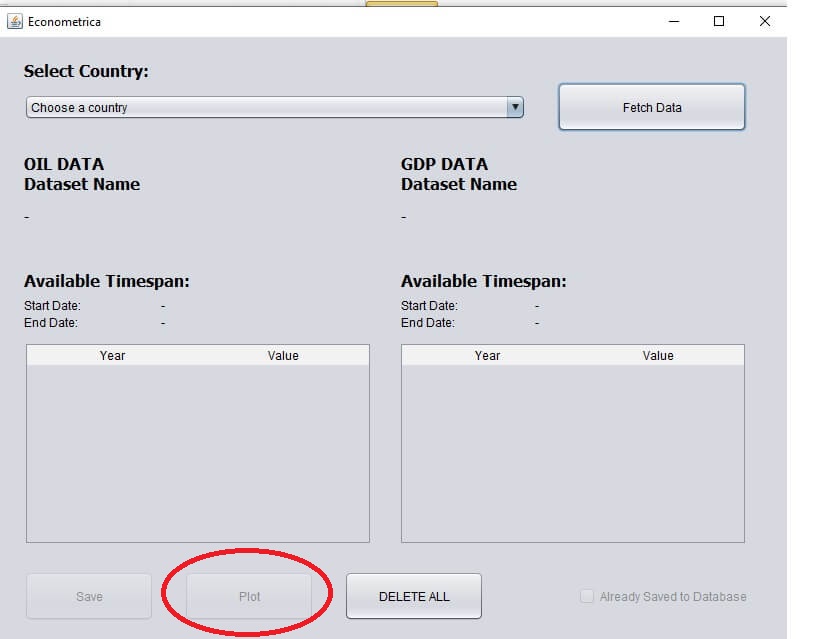
# 13

Το πλήκτρο **“SAVE”** ενεργοποιείται μόνο αν επιλεχθεί χώρα και δεν υπάρχει στην ΒΔ.

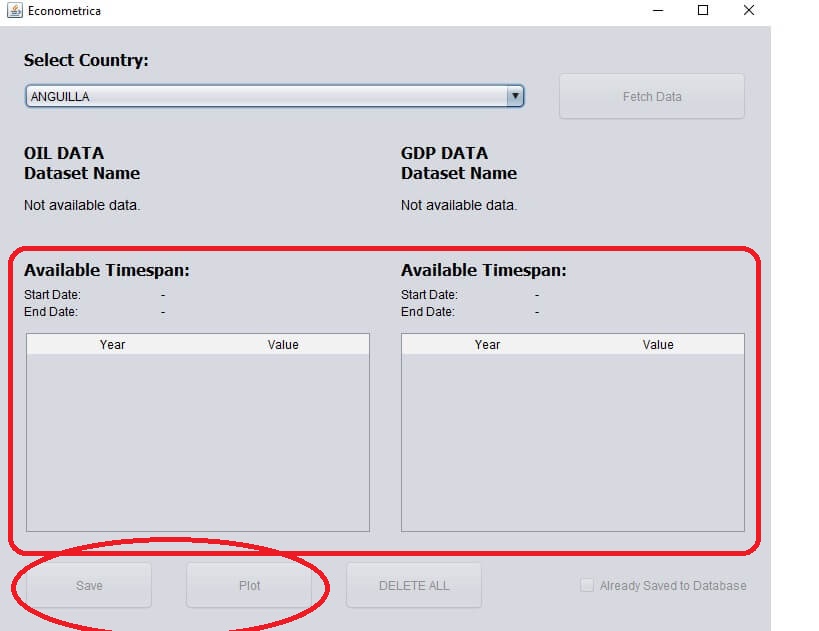




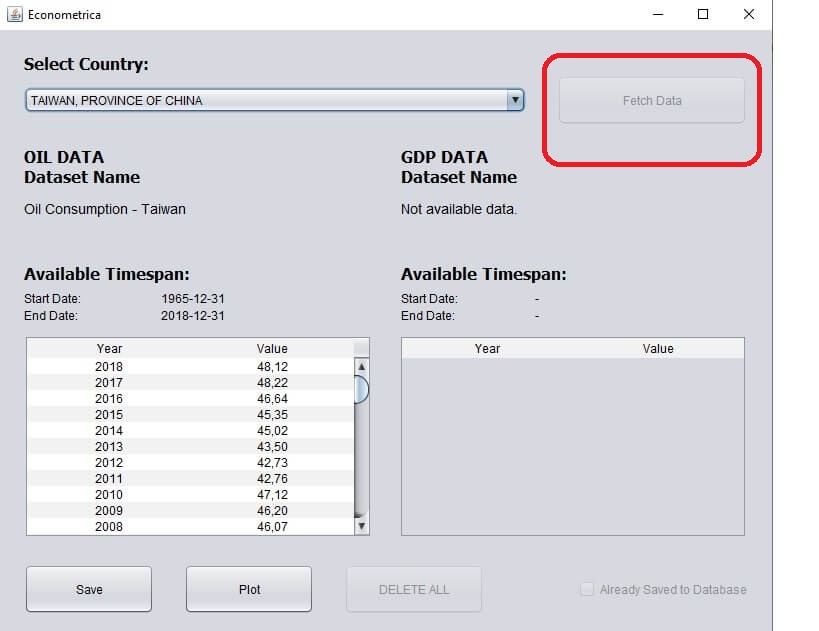
Το πλήκτρο **“PLOT”** ενεργοποιείται μόνο αν επιλεχθεί πρώτα χώρα.



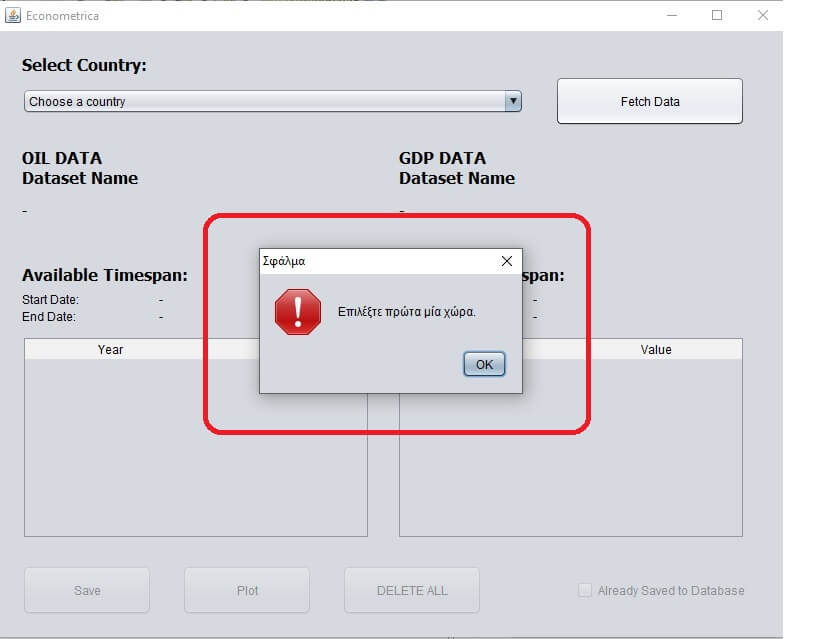
Για τα πλήκτρα “**SAVE” και “PLOT”** είναι ενεργά μόνο αν για την χώρα υπάρχουν δεδομένα.



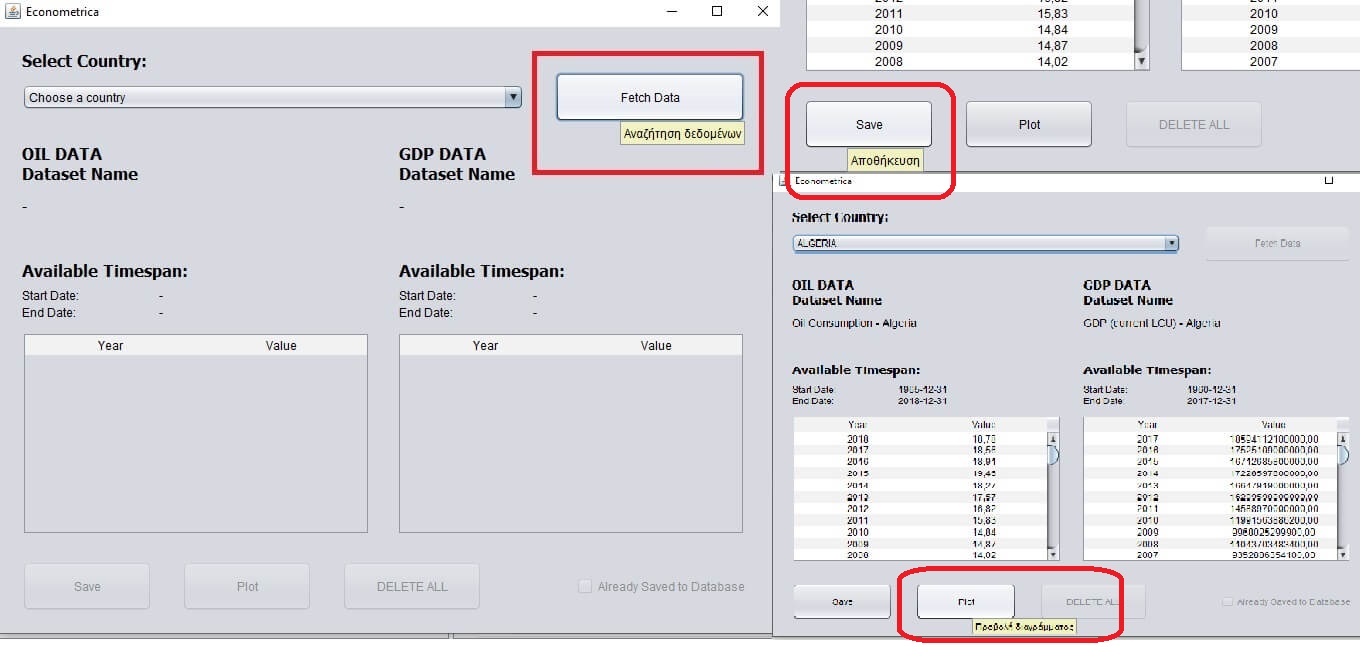
Αναφορικά με το πλήκτρο **“Fetch Data”**, Αν προηγούμενος έχουν έρθει στους πίνακες δεδομένα μια φορά για να είναι λειτουργικό ξανά θα πρέπει να γίνει αλλαγή χώρας.



Επίσης αν δεν έχει γίνει πρώτα επιλογή χώρας και επιλέξουμε από λάθος το πλήκτρο **“Fetch Data”** το σύστημα βγάζει μήνυμα ότι δεν έχει γίνει επιλογή χώρας.



Αν ο pointer του ποντικιού πάει πάνω σε κάποιο πλήκτρο βγαίνει παράθυρο με σχόλιο σε σχέση με την λειτουργία του πλήκτρου.



Τέλος υπάρχει μήνυμα από το σύστημα που ζητάει επιβεβαίωση σε περίπτωση που ο χρήστης τερματίσει την εφαρμογή.

