**Αναφορά Έργου**

**Εφαρμογή Προβολής Πληροφοριών Γευμάτων**

**ΕΑΠ – ΠΛΗ 24**

**3η ΓΡΑΠΤΗ ΕΡΓΑΣΙΑ 2022-2023**

|  |  |  |  |
| --- | --- | --- | --- |
| **Νο** | **Ονοματεπώνυμο** | | **Α.Μ.** |
| 1 | Μιχάλης Νικόλτσιος | | 080550 |
| 2 | Αλέξανδρος - Μάριος Ρεμουντάκης | | 108112 |
| 3 | Νικόλαος Στεφανάτος | | 150547 |
| **ΣΧΟΛΙΑ ΠΡΟΣ ΤΟΝ ΚΑΘΗΓΗΤΗ** | |  | |

***Υπεύθυνη Δήλωση Φοιτητή****: Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία αυτής της εργασίας, είναι πλήρως αναγνωρισμένη και αναφέρεται, είτε στο σημείο «Σχόλια προς καθηγητή», είτε μέσα στην εργασία. Επίσης, έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς, είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη Θεματική Ενότητα.*

Συμφωνώ και αποδέχομαι την ανωτέρω δήλωση

Δε συμφωνώ και δεν αποδέχομαι την ανωτέρω δήλωση (στην περίπτωση αυτή, ο Κ-Σ έχει δικαίωμα να μην αξιολογήσει την εργασία του φοιτητή)

|  |
| --- |
| Ημερομηνία ανακοίνωσης εργασίας:, 24/01/2023  Ημερομηνία παράδοσης εργασίας: 08/03/2023  ΔΕΝ ΘΑ ΔOΘΟΥΝ ΠΑΡΑΤΑΣΕΙΣ |

ΣΤΟΙΧΕΙΑ ΠΟΥ ΣΥΜΠΛΗΡΩΝΕΙ Ο ΚΑΘΗΓΗΤΗΣ

|  |  |
| --- | --- |
| **ΗΜΕΡΟΜΗΝΙΑ ΑΞΙΟΛΟΓΗΣΗΣ** |  |
| **ΒΑΘΜΟΣ** |  |

**ΣΧΟΛΙΑ ΠΡΟΣ ΦΟΙΤΗΤΗ / ΦΟΙΤΗΤΡΙΑ**

##### Αναλυτική Αξιολόγηση

|  |  |  |
| --- | --- | --- |
| **Άσκηση** | **Περιγραφή** | **Ποσοστό** |
| ΔΡΑΣΤΗΡΙΟΤΗΤΑ 1 | Διαχείριση του Έργου με έμφαση στη χρήση του εργαλείου συνεργασίας | 20 |
| ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2.Α | Διάγραμμα ER, Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java (περιλαμβάνει χρήση API και επεξεργασία json) | 20 |
| ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2.Β | Δημιουργία GUI Εφαρμογής | 15 |
| ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2.Γ | Παρουσίαση Στατιστικών Στοιχείων | 15 |
| ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2.Δ | Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής | 15 |
| ΔΡΑΣΤΗΡΙΟΤΗΤΑ 3 | Κριτικός Απολογισμός του Έργου | 5 |
| ΔΡΑΣΤΗΡΙΟΤΗΤΑ 4 | Δημιουργία video | 10 |
|  | **Σύνολο** | 100 |

ΠΕΡΙΕΧΟΜΕΝΑ

[o Αναλυτική Αξιολόγηση 2](#_Toc129176149)

[1 ΕΙΣΑΓΩΓΗ 4](#_Toc129176150)

[2 ΔΙΑΧΕΙΡΙΣΗ ΕΡΓΟΥ 10](#_Toc129176151)

[2.1 Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση 10](#_Toc129176152)

[2.2 Υπολογισμός των προτεραιοτήτων 13](#_Toc129176153)

[2.3 Χρονοδιάγραμμα του έργου 14](#_Toc129176154)

[2.4 To product backlog 16](#_Toc129176155)

[2.5 Οργάνωση ομάδος και αναθέσεις αρμοδιοτήτων 18](#_Toc129176156)

[2.6 Παρακολούθηση της προσπάθειας κατά τη διάρκεια του έργου 20](#_Toc129176157)

[2.7 Χρήση εργαλείου trello 22](#_Toc129176158)

[3 ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2 - ΥΛΟΠΟΙΗΣΗ ΕΡΓΟΥ 24](#_Toc129176159)

[*3.1* Ερώτημα Α– Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java 24](#_Toc129176160)

[3.2 Ερώτημα Β – Δημιουργία GUI Εφαρμογής 68](#_Toc129176161)

[3.3 Ερώτημα Γ – Παρουσίαση στατιστικών στοιχείων γευμάτων 102](#_Toc129176162)

[3.4 Ερώτημα Δ – Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής 111](#_Toc129176163)

[4 ΚΡΙΤΙΚΟΣ ΑΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΕΡΓΟΥ 120](#_Toc129176164)

[5 VIDEO 121](#_Toc129176165)

[6 ΑΝΑΦΟΡΕΣ 122](#_Toc129176166)

# ΕΙΣΑΓΩΓΗ

* 1. **Συνοδευτικά αρχεία**
* Φάκελο Libraries
* Φάκελο MealsDB
* Meal App Product Backlog.xlsx
* Meals App Burn Down Chart.xlsx
* Meals App Class Diagram.pdf
* Meals App Class Diagram.vpp
* Meals App Gantt Diagram.pdf
* Meals App Gantt Diagram.pod
* PLH24\_3ERG\_ΡΕΜΟΥΝΤΑΚΗΣ.docx
* Εξέλιξη του GUI.docx
* Οδηγίες Εγκατάστασης & Χρήσης.docx

**1.2 Περιγραφή της εφαρμογής**

Στα πλαίσια της 3ης εργασίας του μαθήματος ΠΛΗ24 Σχεδιασμός Λογισμικού, ζητήθηκε από τους φοιτητές η ανάπτυξη μίας εφαρμογής με την ονομασία **Meals App**.

Η εφαρμογή αντλεί, επεξεργάζεται και παρουσιάζει δεδομένα από την υπηρεσία ιστού (web service) που περιγράφεται στην ιστοσελίδα TheMealDB: [**https://themealdb.com/api.php**](https://themealdb.com/api.php)

Η **Meals App** παρέχει τις εν λόγω διαδικτυακές υπηρεσίες μέσω web services τύπου REST (Representational State Transfer), προκειμένου να χρησιμοποιηθούν από οποιοδήποτε ενδιαφερόμενο μέρος. Τα σχετικά δεδομένα επιστρέφονται σε μορφή JSON (JavaScript Object Notation).

Μερικές από τις πληροφορίες που μπορούν να αντληθούν είναι:

* Αναζήτηση γεύματος με το όνομα.
* Αναζήτηση ενός τυχαίου γεύματος.
* Αναζήτηση γευμάτων ανά κατηγοριών γευμάτων.
* Φιλτράρισμα ανά κατηγορία γεύματος.
* Αναζήτηση συστατικών κ.α.

Αφού αντληθούν τα δεδομένα η εφαρμογή μπορεί να τα αποθηκεύσει / επεξεργαστεί και σε **τοπική βάση δεδομένων**.

**1.3 Στόχος του έργου.**

Ο στόχος του έργου όπως ζητείται από την εργασία είναι η άντληση των δεδομένων από την API και η παρουσίαση αυτών στον χρήστη με απλό και σταθερό τρόπο.

Συγκεκριμένα η [**themealdb.com**](https://themealdb.com/api.php) είναι μία διαδικτυακή βάση δεδομένων με συνταγές και γεύματα από όλο τον κόσμο. Διαθέτει πάνω από 280 συνταγές, συμπεριλαμβανομένων παραδοσιακών και σύγχρονων πιάτων από διάφορες κουζίνες. Με τη φιλική προς το χρήστη διεπαφή της, ο ιστότοπος είναι εύκολος στην πλοήγηση, επιτρέποντας στους χρήστες να βρουν συνταγές για κάθε περίσταση.

Ένα από τα καλύτερα χαρακτηριστικά του themealdb.com είναι η λειτουργία αναζήτησης. Οι χρήστες μπορούν να αναζητήσουν συνταγές με βάση την κουζίνα, τον τύπο του γεύματος, το συστατικό. Ο ιστότοπος παρέχει επίσης βήμα προς βήμα οδηγίες για κάθε συνταγή.

**Ο βασικός στόχος του έργου** είναι η εφαρμογή **Meals App** να αντλεί δεδομένα από την [**themealdb.com**](https://themealdb.com/api.php) και να τα παρουσιάζει.

Ο χρήστης μπορεί πολύ εύκολα να αναζητήσει ένα γεύμα με διάφορους τρόπους, να αντλήσει τα δεδομένα τους και στη συνέχεια,  να τα επεξεργαστεί ή αποθηκεύσει με τη βοήθεια ενός απλού και εύχρηστου γραφικού περιβάλλοντος (GUI).

Παράλληλα μπορεί να εξάγει πίνακες με στατιστικά δεδομένα και εκτύπωσή τους σε αρχείο .pdf.  
Ο σχεδιασμός της εφαρμογή έγινε με τον γνώμονα να μπορεί να προστεθούν εύκολα και σύντομα νέες λειτουργίες χωρίς να επιβαρύνεται με σύνθετες μεθόδους ή υπολογισμούς.

**1.4 Χαρακτηριστικά και απαιτούμενες λειτουργίες της εφαρμογής.**

**Προδιαγραφή R1: Επιλογές εφαρμογής:**

**1. Προβολή δεδομένων γευμάτων:**

Κατά την έναρξη της MealsApp ο χρήστης μπορεί να αναζητήσει τα γεύματα που τον ενδιαφέρουν με διάφορους τρόπους από την API (που θα μελετηθούν παρακάτω) και να γίνεται παρουσίαση των αντίστοιχων δεδομένων μέσα από ένα εύχρηστο GUI.

**2. Προβολή λίστας γευμάτων ανά κατηγορία γεύματος:**

Ο χρήστης μπορεί να επιλέξει να του παρουσιαστούν οι κατηγορίες των γευμάτων και τα γεύματα που περιέχονται ανά κατηγορία και να μεταβεί στο επιθυμητό γεύμα, ακολουθώντας ένα απλό μονοπάτι σχεδιασμού και όχι περίπλοκες κινήσεις.

**3. Προβολή στατιστικών δεδομένων γευμάτων και εκτύπωση σε αρχείο pdf.**

Δίνεται η δυνατότητα να παρουσιαστούν στο GUI τρεις τρόποι στατιστικών δεδομένων, 3D Columns, 3D Pie και Table και αντίστοιχα να τυπωθούν σε pdf αρχείο.

**4. Έξοδος.**

Η εφαρμογή τερματίζει.

**Προδιαγραφή R2: Προβολή των δεδομένων:**

**Α)**

Τα δεδομένα που είναι υποχρεωτικό να προβληθούν είναι τα: strMeal, strCategory, strArea , strInstructions, αλλά η βιβλιοθήκη GSON μας έδινε τη δυνατότητα μέσω του Serialization της POJO κλάσης να αντλήσουμε και άλλες πληροφορίες με την προσθήκη των αντίστοιχων variables / getters / setters και να τις παρουσιάσουμε.

Ο χρήστης μπορεί να με διάφορους τρόπους να αναζητήσει δεδομένα από την API ή την db, και

συγκεκριμένα μπορεί να αναζητήσει γεύματα:

* Βάση της ονομασίας.
* Βάση της κατηγορίας.
* Βάση της περιοχή προέλευσης.
* Βάση των συστατικών.
* Να αντλήσει ένα τυχαίο γεύμα.
* Να αντλήσει το πρώτο διαθέσιμο γεύμα τις API
* Παράλληλα όπου ήταν εφικτό προβάλλονται πληροφορίες και thumbnail:
  + Κατηγοριών γεύματος.
  + Συστατικών γεύματος.

Αφού ο χρήστης επιλέξει το γεύμα που τον ενδιαφέρει μεταβαίνει στην οθόνη Display Meal από μπορεί να δει τις παρακάτω πληροφορίες:

* strMeal (Το όνομα του γεύματος).
* strCategory (Η κατηγορία στην οποία υπάγεται το γεύμα).
* strArea (Η θεματική περιοχή του εν λόγω γεύματος).
* strInstructions (Οι οδηγίες παρασκευής του γεύματος).

Παράλληλα προστέθηκε η δυνατότητα να εμφανίζονται (όπου είναι διαθέσιμα):

* strMealThumb: URL ενός thumbnail του εκάστοτε γεύματος.
* strYoutube: URL με τον τρόπο παρασκευής του εκάστοτε γεύματος.
* strIngredient1-20: τα συστατικά του εκάστοτε γεύματος.
* strMeasure1-20: την ποσότητα των συστατικών  του εκάστοτε γεύματος.

**Β)**

Μόλις βρεθεί το γεύμα γίνεται μία απλή έγγραφη στη βάση δεδομένων με τα απολύτως απαραίτητα

στοιχεία που εξυπηρετούν την εμφάνιση των στατιστικών δεδομένων, συγκεκριμένα αποθηκεύονται:

* idcategory: Το μοναδικό id του πίνακα Mealcategory (τοπική βάση δεδομένων).
* dtype: (Δημιουργείται αυτόματα) Που χρησιμοποιείται από την JPA και υποδεικνύει τι τύπος είναι το συγκεκριμένο object.
* strMeal: Η ονομασία του γεύματος
* timesOfVisit: Πόσες φορές επισκέφτηκε ο χρήστης το συγκεκριμένο γεύμα τοπικά ή μέσω της API.
* Παράλληλα προβάλλονται στο GUI όλες οι προαναφερόμενες πληροφορίες.

**Γ)**

Υπάρχει η πρόβλεψη και η κατάλληλη επιλογή όπου ο χρήστης μπορεί να “ξεκλειδώσει” και επεξεργαστεί

τα δεδομένα. Εν συνεχεία ο χρήστης έχει τη δυνατότητα να αποθηκεύσει πλήρως το γεύμα, προστέθηκε η δυνατότητα να ελέγχεται αν είναι **ήδη αποθηκευμένο το γεύμα** με ερώτηση προς το χρήστη και με την κατάλληλη επιλογή να κάνει overwrite σε περίπτωση που γίνει κάποια επεξεργασία δεδομένων.

**Δ)**

Με την αντίστοιχη επιλογή μπορούν να διαγράφουν τα δεδομένα από τη βάση δεδομένων. Μόλις

επιλεχθεί το **Delete** ελέγχεται αν έχει αποθηκευτεί το γεύμα και ζητείται επαλήθευση από τον χρήστη, όπου δίνεται η δυνατότητα για ολική διαγραφή του γεύματος ή να κρατηθούν τα απαραίτητα στοιχεία για τις στατιστικές απεικονίσεις. Σε κάθε περίπτωσηστο τέλος της διαγραφής “καθαρίζει” το GUI.

**Προδιαγραφή R3: Προβολή λίστας γευμάτων ανά κατηγορία γεύματος:**

Με την επιλογή **Advanced Search** και συνδυαστικά με το ComboBox γίνεται προβολή της ζητούμενης προδιαγραφής, συγκεκριμένα, αντλούνται από την API οι κατηγορίες των γευμάτων (δε ζητήθηκε αλλά παρατηρήθηκε ότι με μία κλήση API δίνονται thumbnails και Description για όλες τις κατηγορίες οπότε προστέθηκαν στο GUI).

Στη συνέχεια προβάλλονται οι κατηγορίες και ο χρήστης μπορεί να επιλέξει την κατηγορία που τον ενδιαφέρει και γίνεται άντληση των διαθέσιμων γευμάτων, παράλληλα προβάλλονται η φωτογραφία της κατηγορίας και περιγραφή της.

Ακόμη**,** μόλις επιλέξει ο χρήστης το γεύμα που τον ενδιαφέρει, το GUI καλεί τα δεδομένα του γεύματος και μεταβαίνει στην οθόνη **Meals Display** και του παρουσιάζονται τα δεδομένα**.**

Επιπλέον, ο χρήστης με την κατάλληλη επιλογή στο ComboBox μπορεί να επιλέξει και άλλους τρόπους αναζήτησης γευμάτων όπως:

* Βάση της περιοχή προέλευσης.
* Βάση των συστατικών.

Τέλος, στην οθόνη **Meals Display** με bouble click ο χρήστης μπορεί να μεταβεί στη λίστα των **συστατικών** και να προσδιορίσει γεύματα με το επιλεγμένο συστατικό. Όπου είναι διαθέσιμα προβάλλονται τα αντίστοιχα thumbnails, descriptions.

**Προδιαγραφή R4: Προβολή στατιστικών δεδομένων και εκτύπωση σε pdf:**

Δίνεται η δυνατότητα να προβάλλονται στο GUI στατιστικά δεδομένα σε **3D Columns**, **3D Pie** και **Table**, απεικονίσεις.

Για λόγους αισθητικούς στη 3D Columns απεικόνιση προβάλουμε τα **15** ποιο δημοφιλή σε φθίνουσα σειρά, στη 3D Pie προβάλουμε τα **10** ποιο δημοφιλή (με **clockwise** φθίνουσα σειρά) και στην Table **δεν υπάρχει όριο**.

Δημιουργήθηκαν τα αντίστοιχα **Queries** όπου αντλούν από τη βάση με φθίνουσα σειρά τα ποιο δημοφιλή και σε περίπτωση που υπάρχει ισοψηφία στη δημοφιλία προχωράει σε αλφαβητική σειρά.

Τέλος, ο χρήστης μπορεί να κάνει εκτύπωση σε αρχείο pdf με τη δυνατότητα να ορίσει το **path** και το **file name** του αρχείου.

**1.5 Παραδοχές**

Η ευελιξία του κώδικα δίνει τη γενικευμένη παραδοχή ότι η Meals App μπορεί να επεκταθεί με απλές προσθήκες στο backend / frontend.

Ο χρήστης μπορεί να δει την οντότητα ενός γεύματος ανά πάσα στιγμή στην τοπική βάση και αν επιθυμεί μπορεί να την αντλήσει ή να την αφαιρέσει. Η λογική της αναζήτησης σε πολλά σημεία είναι ότι πρώτα ανιχνεύει αν υπάρχει ένα γεύμα στην API και μετά ελέγχει αν υπάρχει στη βάση και ο χρήστης μπορεί ανάλογα να επιλέξει τα δεδομένα που θα του προβληθούν.

Χρησιμοποιήθηκε ένας πίνακας για όλη την λειτουργία της εφαρμογής, εκμεταλλευόμενοι τις δυνατότητες της JPA (ο τρόπος περιγράφεται πιο κάτω).

Σε κάθε αναζήτηση/επίσκεψη/προβολή ενός γεύματος είτε από την API ή την db, δημιουργείται μία απλά εγγραφή με τα βασικά στοιχεία του γεύματος και προστίθεται +1 στο αντίστοιχο πεδίο της εν λόγω εγγραφής.

Ο χρήστης έχει της δυνατότητα να διαγράψει όλα τα δεδομένα ή να κρατήσει τα στατιστικά του γεύματος.

Επίσης, θεωρήσαμε ότι και για την random αναζήτηση να ακολουθηθεί η προαναφερόμενη λειτουργία.

Όσον αφορά την εφαρμογή υλοποιήθηκε σε **Java** και συγκεκριμένα χρησιμοποιήθηκε το **JDK 17.0.2**. Ενώ για βάση δεδομένων χρησιμοποιήθηκε η **Apache Derby** (έκδοση 10.14.2.0). Το GUI δημιουργήθηκε με **Swing**.

Για την συνεργασία των μελών της ομάδας, στην συγγραφή του κώδικα, χρησιμοποιήθηκε το **github** και για IDE χρησιμοποιήθηκε το **Netbeans IDE 16**.

Για τις διαδικασίες του **Scrum** χρησιμοποιήθηκαν διάφορα εργαλεία, όπως:

* **Planningpoker.com** (για τον υπολογισμό του effort και της προτεραιοποίησης)
* **Trello** (για διαχείριση του backlog)
* **Toggl** (για καταγραφή του χρόνου που ξοδεύει το κάθε μέλος στα User Stories)
* **Midjourney** (για εικαστικά)

Χρησιμοποιήθηκε επίσης το **Figma** για σχεδίαση των mockups του GUI και το **Visual Paradigm** για την σχεδίαση διαγραμμάτων, όπως και το **ProjectLibre** για σχεδίαση διαγράμματος **Gantt**

Κοινή χρήση αρχείων έγινε με το **Dropbox**, ενώ χρησιμοποιήθηκαν και το **Google Docs** και **Online MS Office**, για κοινή συγγραφή των απαιτούμενων εγγράφων.

Για να γίνει compile η εφαρμογή και να τρέξει, πρέπει να γίνει η απαραίτητη εισαγωγή βιβλιοθηκών. Σχετικές οδηγίες υπάρχουν στο έγγραφο “**Οδηγίες Εγκατάστασης & Χρήσης.docx**”

# ΔΙΑΧΕΙΡΙΣΗ ΕΡΓΟΥ

## Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση

Η ομάδα ορίστηκε και συναντήθηκε πρώτη φορά στις 27/01, αφού έγινε μία σύντομη συζήτηση για το πως αντιλαμβάνονται την απαιτούμενη προσπάθεια, την επόμενη μέρα έγινε η συνάντηση και χρησιμοποιήθηκε η μέθοδο του [**Planning Poker**](https://www.planningpoker.com/about/). Με τη μέθοδο αυτή, κάθε μέλος της ομάδας ψηφίζει και αξιολογεί τις απαιτήσεις ώστε να υπολογιστεί η απαιτούμενη προσπάθεια (effort).

Αρχίζοντας τη διαδικασία υλοποίησης στις 28/01/2023 είχαμε στη διάθεσή μας, ακριβώς 40 μέρες για την ανάπτυξη και εκπόνηση της εργασίας με τελικό deadline στις 08/03/2023.

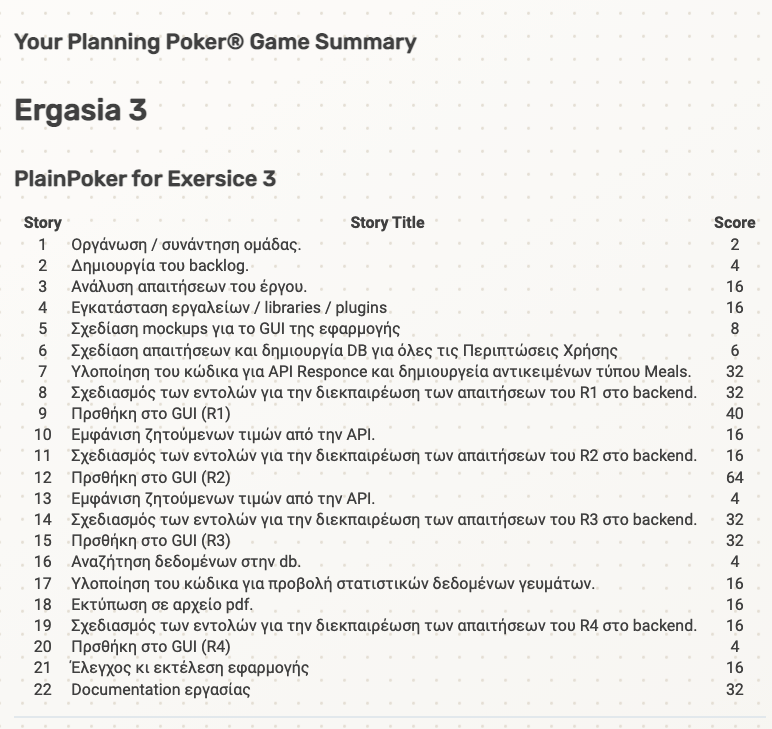
Πάρα πολύ σύντομα έπρεπε να αναλογιστούμε τους παράγοντες που θα μας βοηθούσαν ή εμπόδιζαν για το project, σύμφωνα με το πρότυπο Project Management Body of Knowledge (PMBOK) προσδιορίσαμε τα παρακάτω ερωτήματα.

* Πόσο εξοικειωμένοι είμαστε με τη Java;
* Τι γνώση χρειαζόμαστε για να φτάσουμε στο επιθυμητό επίπεδο;
* Υπάρχει αντίστοιχη προϋπηρεσία;
* Μπορούμε να δουλέψουμε σαν ομάδα ή είμαστε ατομιστές;
* Έχει υλοποιηθεί ο κώδικας για τα σενάρια που καλύπτει το Backlog;
* Ο κώδικας παράγει τα αναμενόμενα αποτελέσματα και άρα καλύπτει τα κριτήρια αποδοχής (acceptance criteria);
* Ο κώδικας έχει δοκιμαστεί και εγκριθεί από όλα τα μέλη της ομάδας;

Με γνώμονα τα παραπάνω και το backlog,  για τον υπολογισμό της απαιτούμενης προσπάθειας ανά απαίτηση χρησιμοποιήθηκε η μέθοδος του [Planning Poker](http://www.planningpoker.com). Με τη μέθοδο αυτή μπορεί η ομάδα να υπολογίσει άμεσα και διασκεδαστικά την προσπάθεια που αναλογεί για κάθε story.

Ο Scrum Master έφτιαξε ένα παιχνίδι με τα **22 User Stories** και ορίστηκε **κάρτες των δυνάμεων του 2** (0, 1, 2, 4, 8, 16, 32, 64, ?, Pass). Με κάθε νούμερο κάρτας να αντιστοιχεί σε **story point**.  
Θεωρήσαμε ότι ένα **story point** αντιστοιχεί σε μία **ανθρωποώρα.**

Ο Scrum Master έκανε ένα μικρό πρόλογο για το κάθε User Story και μετά γινόταν η ψηφοφορία.

  
*Πίνακας 1 - Effort*

Αρχικά στην ψηφοφορία υπήρχαν μεγάλες αποκλίσεις λόγω άγνοιας ή φοβίας λ.χ. Υπολογίστηκε ότι για την εγκατάσταση των βιβλιοθηκών θα θέλαμε 64 **ανθρωποώρες** το οποίο δεν ήταν καθόλου ρεαλιστικό και έπρεπε να αναθεωρήσουμε σε μία πιο λογική τιμή.

Αρχικά υπολογίστηκαν **424 ανθρωποώρες** το οποίο έμοιαζε παράλογο διότι με **8 ανθρωποώρες/ημέρα** θα χρειαζόμασταν **53 ανθρωποημέρες** με αποτέλεσμα να μην προλάβουμε την προθεσμία, οπότε πιέσαμε τους χρόνους και κρατήσαμε σταθερές τις συνολικές **ανθρωποώρες** αλλά ανεβάσαμε την καθημερινή μας ενασχόληση στις **12 ανθρωποώρες/ημέρα** ή 4 **ανθρωποώρες/ημέρα/φοιτητή** με αποτέλεσμα να υπολογίσουμε ότι η ομάδα θα έχει συνολικά **35.33** **ανθρωποημέρες** για να εργαστεί και **4.6 μέρες** για να καθυστερήσει.

Δίνεται ο πίνακας που έγινε ο υπολογισμός:



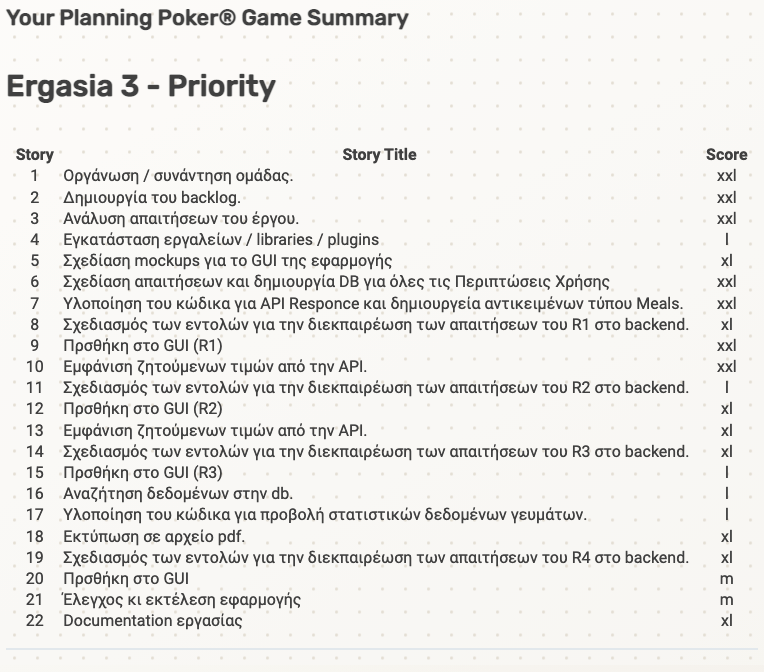
*Πίνακας 2 – Effort ανθρωποημέρες*

## Υπολογισμός των προτεραιοτήτων

Με τον ίδιο τρόπο υλοποιήθηκε η μέθοδος του [**Priority Poker**](https://www.uxforthemasses.com/priority-poker/). Με τη μέθοδο αυτή, κάθε μέλος της ομάδας

ψηφίσε και αξιολόγησε τις απαιτήσεις ώστε να προσδιοριστεί η προτεραιοποίηση των στόχων μας.

Σε αυτή την περίπτωση, αντί για κάρτες με **story points,** χρησιμοποιήθηκαν κάρτες με μεγέθη **T-Shirts** (XXS, XS, S, M, L, XL, XXL, ?, Pass).

  
*Πίνακας 3 - Priority*

Αν και το κάθε μέλος της ομάδας είχε διαφορετική αντίληψη των προτεραιοτήτων, κατά την πορεία της υλοποίησης υπήρχε η σύμφωνη αποδοχή για τις προτεραιότητες και η αντίστοιχη υποστήριξη, αλληλοβοήθεια.

## Χρονοδιάγραμμα του έργου

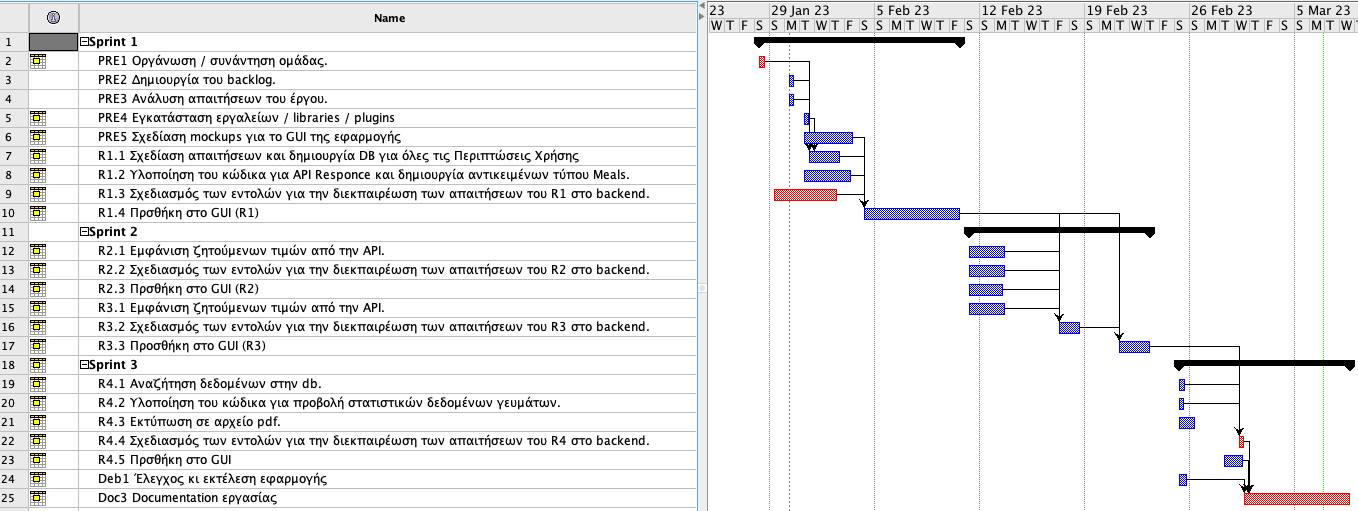
Οι οδηγίες που είχαμε ήταν ότι έπρεπε να μοιράσουμε το χρόνο μας σε τρεις επαναλήψεις (sprints), οι δύο πρώτες επαναλήψεις θα ήταν διάρκειας δύο εβδομάδων, ενώ η τελευταία  θα ήταν διάρκειας δέκα ημερών.

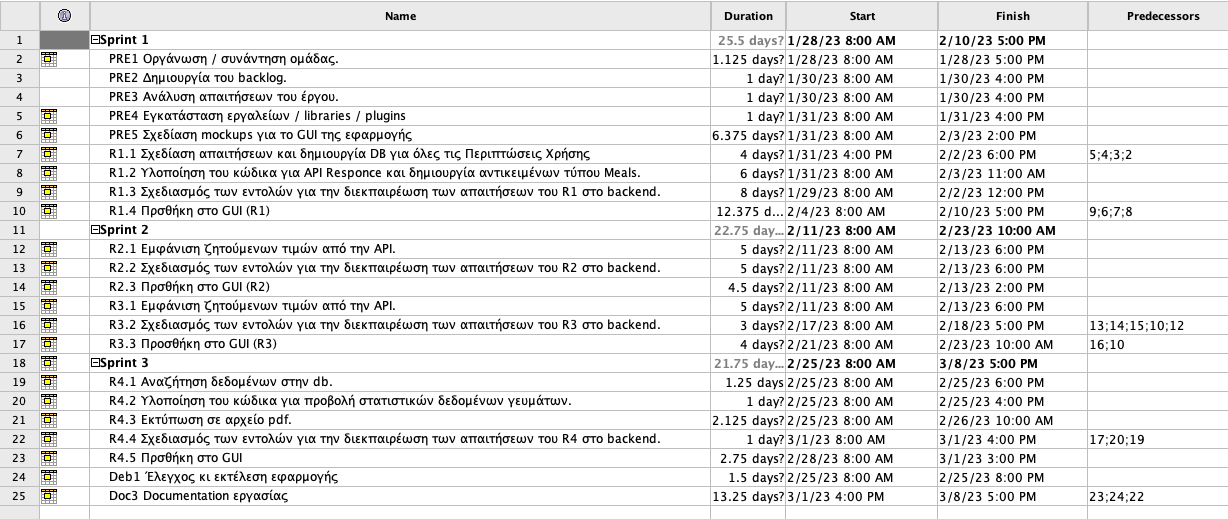
Οπότε το schedule μας ήταν:

* 1ο Sprint: 28/01/2023 - 11/02/2023
* 2ο Sprint: 11/02/2023 - 25/02/2023
* 3ο Sprint: 25/02/2023 - 08/03/2023

Βάλαμε στόχο η κάθε εργάσιμη ημέρα (χωρίς αργίες κτλ) να ορίζεται από 08.00 έως τις 23.00 και προκύπτει το παρακάτω διάγραμμα Gantt (δημιουργήθηκε με το **Project Libre**).

***Σημείωση: στον φάκελο της εργασία υπάρχει το συνοδευτικό αρχείο .pod.***





*Πίνακες 4-5 - Gantt Diagram - Project Libre*

Τα stories που είχε χωριστεί το συγκεκριμένο project ήταν:

|  |  |
| --- | --- |
| **Κωδικός** | **Story** |
| Sprint 1 | |
| PRE1 | Οργάνωση / συνάντηση ομάδας. |
| PRE2 | Δημιουργία του backlog. |
| PRE3 | Ανάλυση απαιτήσεων του έργου. |
| PRE4 | Εγκατάσταση εργαλείων / libraries / plugins |
| PRE5 | Σχεδίαση mockups για το GUI της εφαρμογής |
| R1.1 | Σχεδίαση απαιτήσεων και δημιουργία DB για όλες τις Περιπτώσεις Χρήσης |
| R1.2 | Υλοποίηση του κώδικα για API Responce και δημιουργία αντικειμένων τύπου Meals. |
| R1.3 | Σχεδιασμός των εντολών για την διεκπεραίωση των απαιτήσεων του R1 στο backend. |
| R1.4 | Προσθήκη στο GUI (R1) |
| Sprint 2 | |
| R2.1 | Εμφάνιση ζητούμενων τιμών από την API. |
| R2.2 | Σχεδιασμός των εντολών για την διεκπεραίωση των απαιτήσεων του R2 στο backend. |
| R2.3 | Προσθήκη στο GUI (R2) |
| R3.1 | Εμφάνιση ζητούμενων τιμών από την API. |
| R3.2 | Σχεδιασμός των εντολών για την διεκπεραίωση των απαιτήσεων του R3 στο backend. |
| R3.3 | Προσθήκη στο GUI (R3) |
| Sprint 3 | |
| R4.1 | Αναζήτηση δεδομένων στην db. |
| R4.2 | Υλοποίηση του κώδικα για προβολή στατιστικών δεδομένων γευμάτων. |
| R4.3 | Εκτύπωση σε αρχείο pdf. |
| R4.4 | Σχεδιασμός των εντολών για την διεκπεραίωση των απαιτήσεων του R4 στο backend. |
| R4.5 | Προσθήκη στο GUI |
| Deb1 | Έλεγχος κι εκτέλεση εφαρμογής |
| Doc3 | Documentation εργασίας |

*Πίνακας 6 - Stories*

## To product backlog

***Σημείωση: στον φάκελο της εργασία υπάρχει το συνοδευτικό αρχείο .xlsx χωρισμένο και ανά sprint αναλυτικά όπως ζητείται.***

Συνοπτικά η περιγραφή του backlog με τα κριτήρια αποδοχής όπως αποτυπώνονται στο διάγραμμα **Gantt** παρουσιάζεται στον παρακάτω πίνακα:

|  |  |  |  |
| --- | --- | --- | --- |
| **A/A** | **Κωδικός** | **Story** | **Κριτήρια Αποδοχής** |
| Sprint 1 | | | |
| 1 | PRE1 | Οργάνωση / συνάντηση ομάδας. |  |
| 2 | PRE2 | Δημιουργία του backlog. |  |
| 3 | PRE3 | Ανάλυση απαιτήσεων του έργου. |  |
| 4 | PRE4 | Εγκατάσταση εργαλείων / libraries / plugins |  |
| 5 | PRE5 | Σχεδίαση mockups για το GUI της εφαρμογής | PRE5, PRE4, PRE3,PRE2 |
| 6 | R1.1 | Σχεδίαση απαιτήσεων και δημιουργία DB για όλες τις Περιπτώσεις Χρήσης |  |
| 7 | R1.2 | Υλοποίηση του κώδικα για API Responce και δημιουργία αντικειμένων τύπου Meals. |  |
| 8 | R1.3 | Σχεδιασμός των εντολών για την διεκπεραίωση των απαιτήσεων του R1 στο backend. | R1.3, PRE5, R1.1, R1.2 |
| 9 | R1.4 | Προσθήκη στο GUI (R1) |  |
| Sprint 2 | | | |
| 10 | R2.1 | Εμφάνιση ζητούμενων τιμών από την API. |  |
| 11 | R2.2 | Σχεδιασμός των εντολών για την διεκπεραίωση των απαιτήσεων του R2 στο backend. |  |
| 12 | R2.3 | Προσθήκη στο GUI (R2) |  |
| 13 | R3.1 | Εμφάνιση ζητούμενων τιμών από την API. |  |
| 14 | R3.2 | Σχεδιασμός των εντολών για την διεκπεραίωση των απαιτήσεων του R3 στο backend. | R2.2, R2.3, R3.1, R1.4, R2.1 |
| 15 | R3.3 | Προσθήκη στο GUI (R3) | R3.2, R1.4 |
| Sprint 3 | | | |
| 16 | R4.1 | Αναζήτηση δεδομένων στην db. |  |
| 17 | R4.2 | Υλοποίηση του κώδικα για προβολή στατιστικών δεδομένων γευμάτων. |  |
| 18 | R4.3 | Εκτύπωση σε αρχείο pdf. |  |
| 19 | R4.4 | Σχεδιασμός των εντολών για την διεκπεραίωση των απαιτήσεων του R4 στο backend. | R3.3, R4.2, R4.1 |
| 20 | R4.5 | Προσθήκη στο GUI |  |
| 21 | Deb4.1 | Έλεγχος κι εκτέλεση εφαρμογής |  |
| 22 | Doc4.2 | Documentation εργασίας | R4.5, Deb4.1, R4.4 |

*Πίνακας 7 – Product Backlog*

Στο παραδοτέο αρχείο περιλαμβάνεται η συνολική επισκόπηση του backlog συνολικά αλλά και διαχωρισμένο ανά sprint. Επίσης περιλαμβάνονται η **αρχή** και **λήξη** κάθε story, τα **κριτήρια παραδοχής και** η **προτεραιότητά** όπως περιγράφονται στο διάγραμμα Gantt.

## Οργάνωση ομάδος και αναθέσεις αρμοδιοτήτων

Σύμφωνα με την εκφώνηση της εργασίας η οργάνωση της ομάδας έπρεπε να υλοποιηθεί με τη μεθοδολογία Scrum, η οποία έχει κάποιους βασικούς ρόλους και ανατέθηκαν όπως παρακάτω:

**Ιδιοκτήτηs/διαχειριστήs του Προϊόντος (Product Owner):**

Ιδιοκτήτης του έργου ήταν ο ΣΕΠ του τμήματος, Μανής Γεώργιος.

**Ομάδα ανάπτυξης (Scrum Τeam):**

Η ομάδα ανάπτυξης αποτελούταν από τους:

* Νικόλτσιος Μιχαήλ
* Ρεμουντάκης Αλέξανδρος -  Μάριος
* Στεφανάτος Νικόλαος

**Διαχειριστής του έργου (ScrumMaster):**

Ως διαχειριστή του έργου ορίστηκε ο φοιτητής Ρεμουντάκης Αλέξανδρος - Μάριος.

Η ομάδα δημιουργήθηκε τυχαία κατά τη διάρκεια της ΟΣΣ χωρίς να υπάρχει οποιαδήποτε προηγούμενη γνωριμία των μελών της. Με τη λήξη της ΟΣΣ πραγματοποιήθηκε η πρώτη τηλεδιάσκεψη για να γνωριστούν τα μέλη μεταξύ τους και ορίστηκε ως **Scrum Master** (Διαχειριστής ή ειδικός της Scrum) ο Ρεμουντάκης Αλέξανδρος Μάριος, έπειτα από επιθυμία του γι’ αυτόν τον ρόλο και μην έχοντας κάποια προηγούμενη εμπειρία στο τομέα της Διαχείρισης Έργου.

Λαμβάνοντας υπόψη ότι όλα τα μέλη είχαν ελάχιστο διαθέσιμο χρόνο καθημερινά και μη καθορισμένο ωράριο εργασίας,  συμφωνήθηκε από κοινού η χρήση της εφαρμογής Viber για την επικοινωνία των μελών καθ’ όλο το 24ωρο το οποίο αποτέλεσε εξαιρετική επιλογή καθώς τα μέλη απαντούσαν άμεσα όπου και εάν βρίσκονταν βοηθώντας έτσι ο ένας τον άλλο και αποφεύγοντας πολύτιμες καθυστερήσεις.

Ως ομάδα όλες οι αποφάσεις παίρνονταν από κοινού και σε περίπτωση διαφωνίας κάποιου, με βάση την πλειοψηφία.

Κατά κύριο λόγο, ο σκοπός των μελών της ομάδας ήταν να ασχοληθούν με όλα τα στάδια της δημιουργίας του προγράμματος, προκειμένου να αποκτήσουν πολύτιμη γνώση και εμπειρία. Παρόλα αυτά, έχοντας αναθέσει ρόλους σε κάθε μέλος, ο καθένας ήταν υπεύθυνος για συγκεκριμένα κομμάτια της ανάπτυξης του έργου.

Οι αρμοδιότητες που ανατέθηκαν ανά μέλος ήταν:

|  |  |
| --- | --- |
| Στεφανάτος Νικόλαος | **Front-End Develope**r και ήταν υπεύθυνος για την ανάπτυξη και τη συντήρηση της δομής και του **interface** της εφαρμογής. |
| Ρεμουντάκης Αλέξανδρος - Μάριος | **Scrum Master** και **Back-End Developer**, υπεύθυνος για την ανάπτυξη και συντήρηση της **db** και των διασυνδέσεων **ΑΡΙ** και του **Scrum**. |
| Νικόλτσιος Μιχαήλ | **Full Stack Developer** και **Debugger** βοηθώντας σε όλα τα επίπεδα της ανάπτυξης του λογισμικού και εξετάζοντας τον κώδικα για **σφάλματα** που μπορεί να παρουσιαστούν καθ όλη τη διάρκεια του έργου. |

Σε αυτό το σημείο πρέπει να αναφερθούν μερικά προβλήματα που αντιμετώπισε η ομάδα.  
Αρχικά, τα 2 μέλη είχαν εγκατεστημένο το IntelliJ και 1 μέλος το NetBeans, οπότε αποφασίστηκε με πλειοψηφία να εγκατασταθεί το **IntelliJ** σε όλα τα μέλη και να εργαστούν πάνω σε αυτό για την εκπόνηση της εργασίας. Στη συνέχεια, καθώς οι ημέρες περνούσαν, διαπιστώθηκε από την ομάδα ότι το IntelliJ **δεν παρείχε αρκετή ευκολία στον χρήστη για τη δημιουργία του GUI** σε αντίθεση με το NetBeans. Έτσι, αφού αφιέρωσαν αρκετές εργατοώρες πάνω στο IntelliJ, αποφασίστηκε να χρησιμοποιηθεί το NetBeans. Αυτό είχε σαν αποτέλεσμα την περαιτέρω **σπατάλη χρόνου** μέχρι να εξοικειωθούν τα μέλη με την πλατφόρμα και τις ιδιαιτερότητές της, καθώς και με προβλήματα που προέκυπταν με τις εκδόσεις των JDK, Derby και NetBeans.

Στη συνέχεια, παρουσιάστηκε ένα ακόμα πρόβλημα όταν ο κώδικας άρχισε να μεγαλώνει ακόμη περισσότερο. Η ομάδα δυσκολευόταν να διαχειριστεί τις πολλές διαφορετικές εκδόσεις του κώδικα που είχαν δημιουργηθεί και να τις συγχρονίσει μεταξύ τους. Για να αντιμετωπιστεί αυτό το πρόβλημα, η ομάδα αποφάσισε να χρησιμοποιήσει τη διαδικασία του **version control** με το **Github**. Ετσι, ο **Full Stack Developer** δημιούργησε ένα **repository** στο GitHub και στη συνέχεια όλα τα μέλη συνδέθηκαν σε αυτό. Πλέον κάθε μέλος μπορούσε να δουλεύει πάνω στον κώδικα του σε ξεχωριστό **branch** και ο κώδικας θα συγχωνεύονταν αυτόματα στο κύριο branch όταν επιβεβαιώνονταν οι αλλαγές του.

Η χρήση του Git έκανε τη διαχείριση του κώδικα πολύ πιο εύκολη και αποτελεσματική.

Οι πρακτικές του Scrum που εφαρμόστηκαν στην ομάδα ήταν η **Daily Scrum**, η **Sprint Planning**, η **Sprint Review** και η **Sprint Retrospective**, καθώς και η χρήση του **Product Backlog** για την καταγραφή των χαρακτηριστικών του προϊόντος. Η πρακτική του Sprint και Sprint Backlog δε χρησιμοποιήθηκε, καθώς οι προηγούμενες πρακτικές κάλυπταν επαρκώς τα αποτελέσματα που επιδιώκονταν. Η εφαρμογή αυτών των πρακτικών βοήθησε την ομάδα να έχει μια καλύτερη οργάνωση, ενημερωμένη πρόοδο και βελτιωμένη διαδικασία ανάπτυξης του προϊόντος.

Τέλος, για την οργάνωση μας και έλεγχο προόδου είχαμε ορίσει δύο συναντήσεις κάθε εβδομάδα.

## Παρακολούθηση της προσπάθειας κατά τη διάρκεια του έργου

To **burndown chart** είναι ένα γράφημα με δύο άξονες y και x. Ο κάθετος άξονας (y) μετρά το συνολικό ποσό εργασίας που η ομάδα εκτιμά ότι θα ολοκληρώσει κατά τη διάρκεια του τρέχοντος sprint. Ο οριζόντιος άξονας (x) δείχνει τον αριθμό των ημερών που απομένουν μέχρι το τέλος του sprint.

Στο διάγραμμα υπάρχουν **συνήθως** δύο γραμμές: H **πραγματική γραμμή εργασίας** (που αντιπροσωπεύει την πρόοδο της ομάδας) Την **ιδανική γραμμή εργασίας** και μερικές φορές την **ταχύτητα υλοποίησης** (velocity).

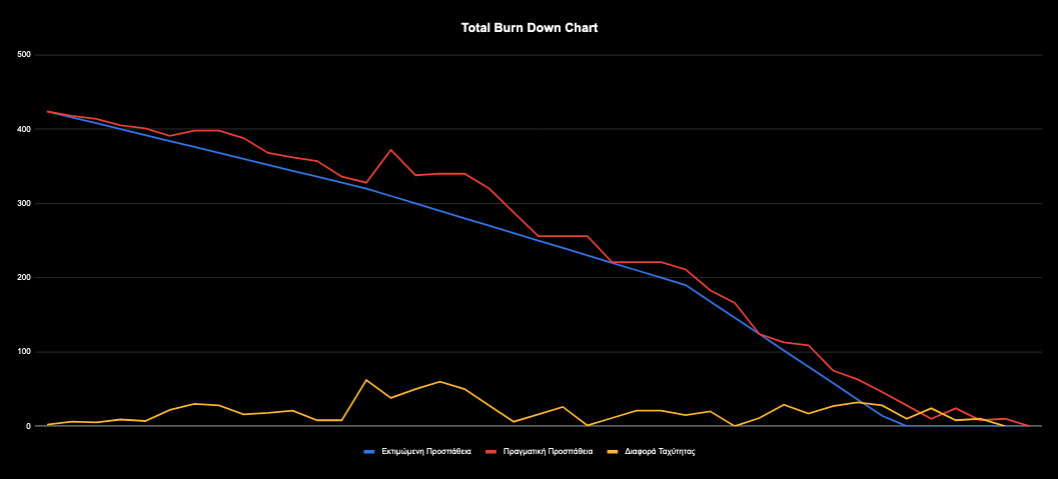
Γενικά η **πραγματική γραμμή εργασίας** πρέπει να ακολουθεί όσο το δυνατόν πιο πιστά την **ιδανική γραμμή** εργασίας. Διότι αυτό θα έχει σαν αποτέλεσμα ότι η εργασία ολοκληρώνεται σταδιακά και με τέτοιο ρυθμό ώστε να μπορεί να ολοκληρωθεί μέχρι το τέλος του κάθε sprint.

Μια καλή πρακτική για τον scrum master μιας ομάδας είναι να επανεξετάζει το burndown chart σε καθημερινή βάση (όπως και έγινε). Με αυτόν τον τρόπο εντοπίζονται δυνητικά προβλήματα προόδου που συμβαίνουν στο sprint.

**(Σημείωση: στα συνοδευτικά αρχεία υπάρχει το αντίστοιχο .xlsx με αναλυτικούς πίνακες ανά sprint)**

Τα Sprint Backlogs αποτυπώνουν τα στοιχεία του Product Backlog που επιλέγονται για υλοποίηση στο Sprint μαζί με το πλάνο για την επίτευξη του στόχου. Η ομάδα συμφώνησε ότι το "Έτοιμο" σημαίνει ότι έχει υλοποιηθεί ο κώδικας, είναι σχολιασμένος, παράγει τα αναμενόμενα αποτελέσματα, έχει δοκιμαστεί από όλα τα μέλη της.

Παρακάτω παρουσιάζεται το συνολικό burndown chart:



*Πίνακας 8 – Burn Down Chart*

Τα γενικό συμπέρασμα που μπορούμε να αντλήσουμε είναι ότι δεν ήμασταν τόσοι καλοί στις προβλέψεις, ενώ στην αρχή νομίζαμε ότι είχαμε τον έλεγχο και η εφαρμογή υλοποιείται σταδιακά και ομαλά, προέκυψαν μεγάλα προβλήματα με διαφορετικές συνιστώσες όπως:

* Έλλειψη γνώσεων Java.
* Τρόπος λειτουργίας των library
* Κακός σχεδιασμός.
* Διάφορα bugs
* JPA Queries
* Δημιουργία βάσης δεδομένων.

Οπότε στα σημεία που δημιουργούνται τα “spikes” ή οι μεγάλες αποκλίσεις στην **Πραγματική Προσπάθεια** υποδεικνύεται ότι η ομάδα είχε πάρει φωτιά και δούλευε ώστε να ξεπεραστούν τα προβλήματα.

Ενώ, αρχικά ήμασταν πολύ αισιόδοξοι ότι θα τελειώσουμε περίπου 4 μέρες πριν το deadline υπήρξαν μέρες που έπρεπε να δουλέψουμε έως και 80% πιο πολύ για να προλάβουμε την παράδοση.

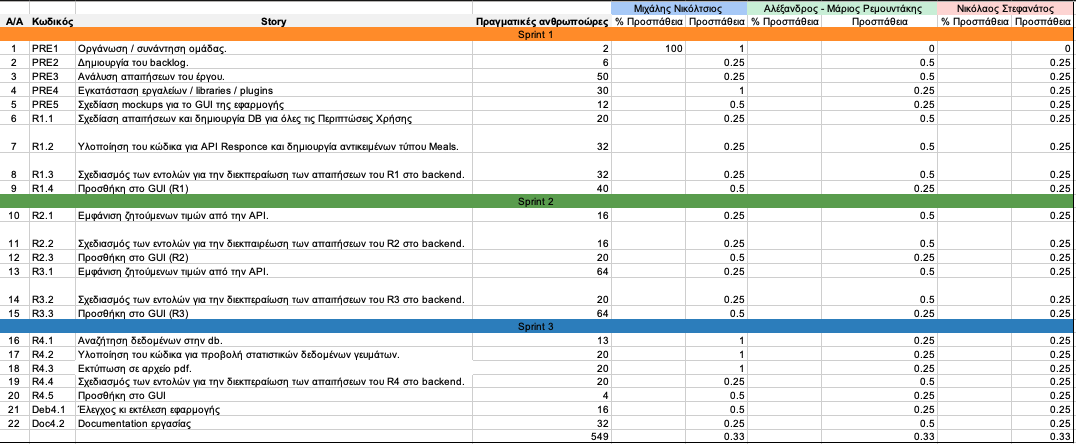
Πιο συγκεκριμένα:

Ενώ στην αρχή του πρώτου 1ου sprint είχαμε κάνει όλα τα προ απαιτούμενα (συνάντηση, επιλογή scrum master κτλ) και είχαμε αρχίσει την υλοποίηση κάποιου κώδικα, είδαμε ότι μεγάλο **τροχοπέδη** ήταν το GUI.  
Κάναμε πολλούς πειραματισμούς ώστε να βρούμε τη βέλτιστη λύση και όπως διακρίνουμε όταν βρέθηκε η λύση για τα κριτήρια αποδοχής PRE5, PRE4, PRE3,PRE2, επανήλθαμε πιο κοντά στην Εκτιμώμενη Προσπάθεια.

Στο 2ο sprint προσπαθήσαμε να πλησιάσουμε όσο γίνεται την Εκτιμώμενη Προσπάθεια με μεγάλο κόπο και προσωρινά το επιτύχαμε.

Στο 3ο sprint βρήκαμε ένα λογικό σφάλμα στην εκτέλεση του κώδικα με αποτέλεσμα να καθυστερήσουμε κατά **100 περίπου ανθρωποώρες** τα κριτήρια αποδοχής R3.3, R4.2, R4.1 και να δυσκολευτούμε αρκετά για να πλησιάσουμε την **Εκτιμώμενη Προσπάθεια** λίγες μέρες πριν το deadline.

Ενδεικτικά η προσπάθεια ανά μέλος της ομάδας ανά story.

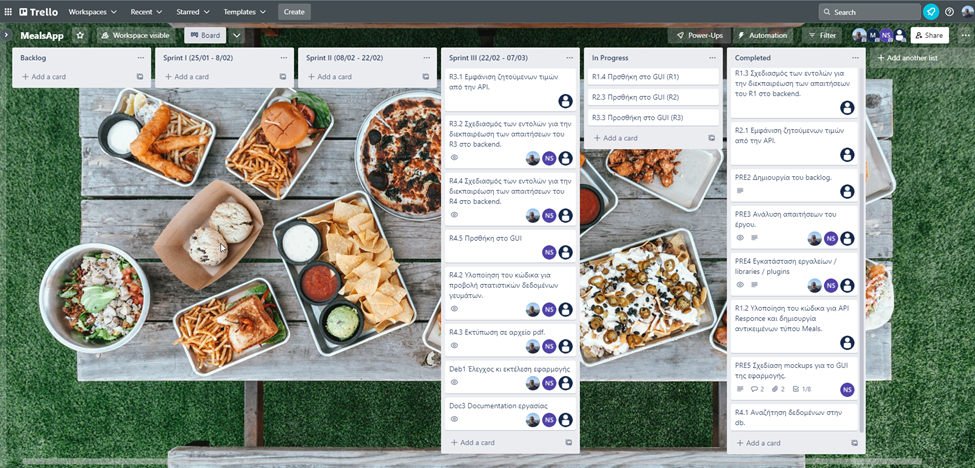
**

*Πίνακας 9 – Προσπάθεια ανά χρήστη.*

## Χρήση εργαλείου trello

Το Agile Board διαμορφώθηκε έχοντας 6 λίστες :

* Backlog
* Sprint I  (25/01-08/02)
* Sprint II (08/02-22/02)
* SprintIII (22/02 – 07/03)
* In Progress
* Completed



*Πίνακας 10 – Trello*

Για να ορίσουμε τις κατηγορίες δραστηριοτήτων στο **Trello**, αναλύσαμε τα διάφορα στάδια του έργου μας και τις απαιτούμενες δραστηριότητες για την ολοκλήρωσή του. Στη συνέχεια, δημιουργήσαμε λίστες για κάθε στάδιο και στη συνέχεια διαχωρίσαμε τις δραστηριότητες σε κατηγορίες όπως προγραμματισμός, σχεδίαση και τεκμηρίωση.

Για την καταγραφή του χρόνου που χρειάστηκε για κάθε δραστηριότητα δεν χρησιμοποιήθηκε το Trello (π.χ. Trello Time Tracking by Apploye), καθώς το πρώτο καιρό δεν υπήρχε μεγάλη εξοικείωση με αυτό και καταλήξαμε στη χρήση άλλων εργαλείων για την καταγραφή των χρόνων (Excel).

Όπως προαναφέραμε η ομάδα συνεργάστηκε στο Agile Board του Trello, δημιουργώντας λίστες για κάθε στάδιο του έργου και κάρτες για κάθε δραστηριότητα που χρειαζόταν να ολοκληρωθεί. Δεν χρησιμοποίησε εκτενώς τα διαφορετικά χρώματα για  την πρόοδο και το επίπεδο προτεραιότητας των διαφόρων δραστηριοτήτων. Επίσης, χρησιμοποιήθηκε η λειτουργία σχολιασμού του Trello για να επικοινωνούν και να συζητούν για τις διάφορες δραστηριότητες και τα προβλήματα που προέκυπταν.

Ορισμένες από τις χρήσιμες λειτουργίες του Trello που χρησιμοποίησε η ομάδα  ήταν οι παρακάτω:

**Λίστες**: Δημιουργήσαμε λίστες για κάθε στάδιο του έργου μας και για να οργανώσουμε τις διάφορες δραστηριότητες που χρειαζόμασταν να ολοκληρώσουμε.

**Κάρτες**: Δημιουργήσαμε κάρτες για κάθε δραστηριότητα που χρειαζόταν να ολοκληρωθεί και τις τοποθετήσαμε στην κατάλληλη λίστα.

**Χρώματα**: Χρησιμοποιήσαμε διαφορετικά χρώματα για να δείξουμε την πρόοδο και το επίπεδο προτεραιότητας των διαφόρων δραστηριοτήτων.

**Σχολιασμός**: Χρησιμοποιήσαμε τη λειτουργία σχολιασμού του Trello για να επικοινωνούμε και να συζητάμε  για τις διάφορες δραστηριότητες.

Εύκολες στη χρήση ήταν οι λειτουργίες που μας επέτρεπαν να μετακινήσουμε **εύκολα και γρήγορα τις κάρτες** από μια λίστα σε μια άλλη, να αντιγράψουμε και να επικολλήσουμε κάρτες και να προσθέσουμε σχόλια.

Η λειτουργία αλληλεπίδρασης (**ή σύνδεση με άλλες εφαρμογές**) με άλλα εργαλεία της αγοράς μας φάνηκε λιγότερο χρήσιμη για το συγκεκριμένο έργο, καθώς δεν απαιτούνταν κάτι εξειδικευμένο και δεν είχαμε αρκετό χρόνο για να κάνουμε δοκιμές.

# ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2 - ΥΛΟΠΟΙΗΣΗ ΕΡΓΟΥ

## Ερώτημα Α– Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java

**(Σημείωση: στα συνοδευτικά αρχεία υπάρχει το αντίστοιχο .vpp)**

Ακολουθήθηκε μία πολύ απλή σκέψη για την συνολική υλοποίηση της εφαρμογής.

Μπορούμε να οραματιστούμε ότι υπάρχει μία συνολική συσχέτιση με τα δεδομένα που παρέχονται στην εφαρμογή, οπότε μπορούμε να υποθέσουμε ότι τα δεδομένα από την API έχουν σχέσεις αντικειμενοστρέφειας, δηλαδή οι Meals Categories είναι ο «γονέας»  και τα Meals τα «παιδιά» από άποψη κληρονομικότητας, άρα:

* Meal Category parent class.
* Meals subclass.

Έτσι, με δύο μόνο classes και χρησιμοποιώντας τις δυνατότητες των GSON και JPA μπορούμε να ελέγχουμε τα αντικείμενα με μόνο δύο «οντότητες» χωρίς να χρειάζεται να γράψουμε ή ορίσουμε δεκάδες μεταβλητές κτλ.

Οι GSON και JPA με απλό τρόπο μας έδωσαν την δυνατότητα από 4 ζητούμενες μεταβλητές να χειριζόμαστε πάνω από **50** χρησιμοποιώντας την κληρονομικότητα και τις δυνατότητες του αντικειμενοστρεφή προγραμματισμού (object-oriented programming).

Εδώ εξηγείται η ύπαρξη και χρήση της υπερκλάσης **MealCategory** διότι μας δίνει την ευελιξία να μην **ντετερμινίσουμε** την εφαρμογή μας. Σε περίπτωση που υπάρξει **προσθήκη ή αφαίρεση κατηγορίας ή πεδίου** από την **API** η εφαρμογή θα λειτουργεί ομαλά και όποια αλλαγή απαιτείται μπορεί να προστεθεί χωρίς να χρειάζονται **σύνθετες αλλαγές**.

Επίσης, για λόγους καλύτερης διαχείρισης αποφασίστηκε να αποθηκεύονται και τα δεδομένα των κατηγοριών διότι σε μελλοντική ανάπτυξη η **MealsApp** θα μπορούσε να λειτουργεί και **χωρίς σύνδεση στο Internet** αντλώντας τα δεδομένα από την τοπική βάση, χρησιμοποιώντας **ένα πίνακα** με την ιδιότητα **@Inheritance(strategy = InheritanceType.SINGLE\_TABLE)**.

Παρά ταύτα έγινε μελέτη και προστέθηκαν όπου ήταν απαραίτητο τα αντίστοιχα προειδοποιητικά μηνύματα σε περίπτωση αδυναμίας της σύνδεσης ή αποτυχία λήψης δεδομένων.

Επίσης για τις περιπτώσεις χρήσης που ζητούνταν σύνθετες αναζητήσεις, με την δημιουργία **απλών Queries** η JPA έφερνε τα δεδομένα που επιθυμούσαμε.

Προσπαθήσαμε να ακολουθήσουμε το πρότυπο **Model-View-Controller (MVC)**.

Το Model-View-Controller (MVC) είναι ένα αρχιτεκτονικό πρότυπο λογισμικού που χρησιμοποιείται για την ανάπτυξη διεπαφών χρήστη για εφαρμογές. Το πρότυπο διαχωρίζει μια εφαρμογή σε τρία αλληλοσυνδεόμενα στοιχεία: το Μοντέλο, την Προβολή και τον Ελεγκτή.

Το Μοντέλο(Model) αντιπροσωπεύει τα δεδομένα και τη λογική της εφαρμογής. Χειρίζεται την αποθήκευση, την ανάκτηση και τον χειρισμό δεδομένων και περιέχει τους επιχειρηματικούς κανόνες και τους αλγορίθμους που καθορίζουν τη συμπεριφορά της εφαρμογής.

Η προβολή (View) αντιπροσωπεύει τη διεπαφή χρήστη της εφαρμογής. Είναι υπεύθυνο για την απόδοση των δεδομένων από το Μοντέλο και την εμφάνισή τους στο χρήστη. Οι προβολές μπορούν να περιλαμβάνουν πράγματα όπως οθόνες, παράθυρα διαλόγου, φόρμες και αναφορές.

Ο Ελεγκτής (Controller) ενεργεί ως ενδιάμεσος μεταξύ του Μοντέλου και της Προβολής. Λαμβάνει την είσοδο του χρήστη από την Προβολή και ενημερώνει ανάλογα το Μοντέλο. Στη συνέχεια ενημερώνει την Προβολή με τα νέα δεδομένα από το Μοντέλο. Οι ελεγκτές χειρίζονται επίσης συμβάντα, όπως αλληλεπιδράσεις του χρήστη ή ειδοποιήσεις του συστήματος.

Το πρότυπο **MVC** προωθεί το διαχωρισμό των προβλημάτων και επιτρέπει την ανάπτυξη αρθρωτού και επαναχρησιμοποιήσιμου κώδικα. Διευκολύνει επίσης τη συνεργασία μεταξύ διαφορετικών ομάδων, καθώς οι προγραμματιστές μπορούν να εργάζονται ανεξάρτητα σε διαφορετικά στοιχεία της εφαρμογής. Το μοτίβο χρησιμοποιείται συνήθως σε πλαίσια ανάπτυξης ιστού, όπως το **Ruby on Rails**, το **Django** και το **ASP.NET**.

Οπότε συνολικά έχουμε τα παρακάτω πακέτα με τις αντίστοιχες κλάσεις που περιέχουν:

Πακέτο API, εκτελούνται τα request από την API και με την βοήθεια της GSON και της MealClasses δημιουργεί αντικείμενα Meal, MealsClasses και κάποιες άλλες βοηθητικές λειτουργίες.

* API  
  APIRequest.java  
  JsonParsers.java

Πακέτο Charts, εκτελούνται οι εντολές για να παρουσιάζονται τα διάφορα γραφήματα και η εκτύπωση τους όπου με την βοήθεια του πακέτου DB αντλεί τα δεδομένα που απαιτούνται.

* Charts  
  ColumnChart.java  
  PieChart.java  
  PrintList.java

Πακέτο DB, περιέχει τις απαραίτητες ενέργειες ώστε να δημιουργείται αυτόματα db και αν δεν υπάρχει κατά την έναρξη της εφαρμογής, επίσης αντλούνται με τα κατάλληλα queries τα απαραίτητα δεδομένα.

* DB  
  DatabaseCreation.java  
  mealsCategoryQueryOperations.java  
  mealsQueryOperations.java

Πακέτο GUI, περιέχει όλες οι απαραίτητες λειτουργίες για την ύπαρξη του GUI.

* GUI  
  commandList.java  
  HelpCommands.java  
  MainForm.form  
  MainForm.java

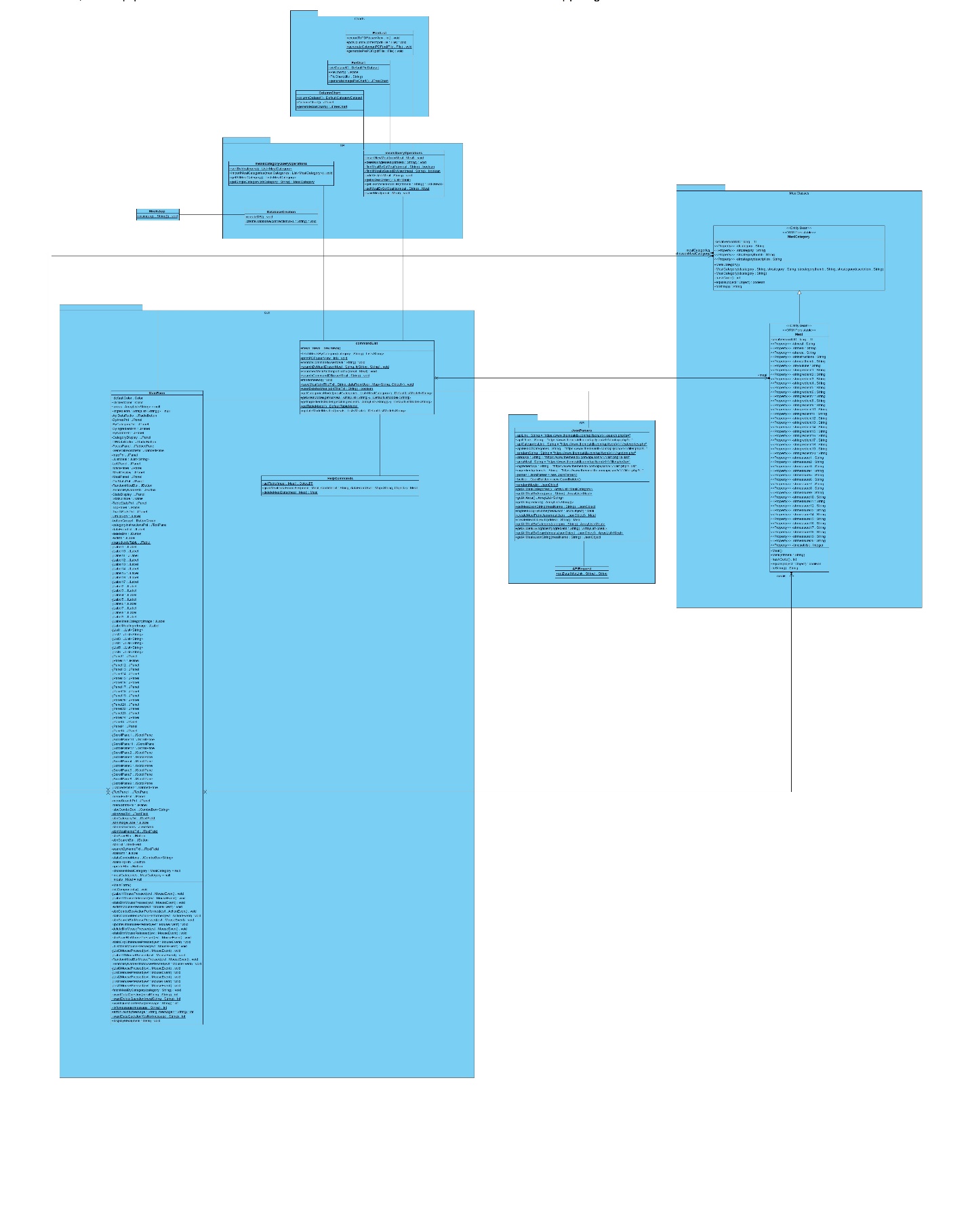
Πακέτο MealClassesν περιέχει τις **POJO** **classes** που απαιτούνται, περιέχουν [**GSON** **@SerializedName**](https://makeinjava.com/serialization-deserialization-object-pojo-java-example/) για να μην υπάρχει conflict μεταξύ GSON και JPA.

* MealClasses  
  Meal.java  
  MealCategory.java

Η Main κλάσση.

MealsApp.java

Οπότε παρακάτω παρουσιάζεται το διάγραμμα κλάσεων όπως περιγράφηκε:



B) Classes εφαρμογής:

package mealsapp. MealsApp.java

package mealsapp;  
  
// Εισαγωγή των απαραίτητων βιβλιοθηκών και κλάσεων για την εκτέλεση του προγράμματος  
import mealsapp.API.JsonParsers;  
import mealsapp.GUI.MainForm;  
import javax.swing.UIManager;  
import javax.swing.UnsupportedLookAndFeelException;  
import mealsapp.DB.DatabaseCreation;  
*/\*\*  
 \*   
 \** ***@author*** *\* ΡΕΜΟΥΝΤΑΚΗΣ ΑΛΕΞΑΝΔΡΟΣ ΜΑΡΙΟΣ  
 \* ΣΤΕΦΑΝΑΤΟΣ ΝΙΚΟΛΑΟΣ  
 \* ΝΙΚΟΛΤΣΙΟΣ ΜΙΧΑΗΛ   
 \*   
 \*/*// Κλάση για την εκτέλεση του προγράμματος  
public class MealsApp {  
  
  
 public static void main(String[] args) throws UnsupportedLookAndFeelException {  
  
 /\*  
 Για την λειτουργία του προγράμματος δημιουργείται αυτόματα μια DB με τα στοιχεία, όνομα MealsDB user=mealsdb password=mealsdb  
 Σε περίπτωση που δεν δημιουργηθεί αυτόματα, δημιουργείστε μία βάση με τα κάτωθι στοιχεία:  
 Για driver Java DB (Embedded):  
 jdbc:derby:MealsDB;create=true;user=mealsdb;password=mealsdb  
 Για driver Java DB (Network):  
 jdbc:derby://localhost:1527/MealsDB;create=true;user=mealsdb;password=mealsdb  
 \*/  
 try {  
 // Ορισμός του συστήματος εμφάνισης του προγράμματος στην εμφάνιση των κουμπιών και των παραθύρων  
 UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());  
 } catch (ClassNotFoundException | InstantiationException | IllegalAccessException | UnsupportedLookAndFeelException e) {  
 e.printStackTrace();  
 }  
 // Δημιουργία της βάσης δεδομένων, αν δεν υπάρχει ήδη  
 DatabaseCreation.createDB();  
 System.out.println("Current working directory: " + System.getProperty("user.dir"));  
  
 // Εκκίνηση της κεντρικής φόρμας της εφαρμογής  
 MainForm mf = new MainForm();  
 mf.pack();  
 mf.setVisible(true);  
  
 }  
  
}

package mealsapp.API. APIRequest.java

// Πακέτο που περιέχει την κλάση APIRequest  
package mealsapp.API;  
  
// Εισαγωγή απαραίτητων βιβλιοθηκών  
import java.io.IOException;  
import okhttp3.OkHttpClient;  
import okhttp3.Request;  
import okhttp3.Response;  
  
// Δημιουργία κλάσης APIRequest  
public class APIRequest {  
  
*/\*\*  
 \* Μέθοδος για κάλεσμα API και λήψη απόκρισης  
 \** ***@param*** *httpLink : μια συμβολοσειρά που χρησιμοποιείται ως ερώτημα για την API  
 \** ***@return*** *String : η απόκριση από την API σε μορφή συμβολοσειράς.   
 \*/* public static String getData(String httpLink) {  
 // Δημιουργία αντικειμένου OkHttpClient  
 OkHttpClient client = new OkHttpClient();  
  
 // Δημιουργία αντικειμένου Request με το URL που περιέχεται στη μεταβλητή httpLink  
 Request request = new Request.Builder().url(httpLink).build();  
  
 // Προσπάθεια εκτέλεσης του κλήσης API και λήψη απόκρισης  
 try (Response response = client.newCall(request).execute()) {  
 // Εάν η απόκριση είναι επιτυχής και περιέχει σώμα  
 if (response.isSuccessful() && response.body() != null) {  
  
 // Επιστροφή του σώματος απόκρισης ως συμβολοσειρά  
 return response.body().string();  
 } else {  
 // Επιστροφή null εάν η απόκριση δεν είναι επιτυχής ή δεν περιέχει σώμα  
 return "null";  
 }  
 } catch (IOException e) {  
   
 return "null";  
 }  
 }  
}

package mealsapp.API. JsonParsers.java

package mealsapp.API;  
  
import mealsapp.MealClasses.MealCategory;  
import mealsapp.MealClasses.Meal;  
import com.google.gson.JsonParser;  
import com.google.gson.GsonBuilder;  
import com.google.gson.JsonObject;  
import com.google.gson.JsonElement;  
import com.google.gson.JsonArray;  
import com.google.gson.Gson;  
import java.util.ArrayList;  
  
public class JsonParsers {  
  
 // Ορισμός σταθερών για τα links της API  
 static String apiLink = "https://www.themealdb.com/api/json/v1/1/search.php?s=";  
 static String apiIDLink = "https://www.themealdb.com/api/json/v1/1/lookup.php?i=";  
 static String apiCategoriesLink = "https://www.themealdb.com/api/json/v1/1/categories.php";  
 static String apiMealOfCategories = "https://www.themealdb.com/api/json/v1/1/filter.php?c=";  
 static String randomString = "https://www.themealdb.com/api/json/v1/1/random.php";  
 static String areaApi = "https://www.themealdb.com/api/json/v1/1/list.php?a=list";  
 static String areaMeal = "https://www.themealdb.com/api/json/v1/1/filter.php?a=";  
 static String ingredientApi = "https://www.themealdb.com/api/json/v1/1/list.php?i=list";  
 static String ingredientgetmeals = "https://www.themealdb.com/api/json/v1/1/filter.php?i=";  
  
 // Δημιουργία αντικειμένων για τον parser και τον builder του Gson  
 static JsonParser parser = new JsonParser();  
 static GsonBuilder builder = new GsonBuilder();  
  
 //Μέθοδος που επιστρέφει ένα τυχαίο γεύμα από την API  
 public static JsonObject randomMeal() {  
 JsonObject jsonObject = null;  
 try {  
 // Λαμβάνουμε τα δεδομένα μέσω της μεθόδου getData από την API   
 String jsonData = APIRequest.getData(randomString);  
 JsonParser parser = new JsonParser();  
  
 // Μετατρέπουμε τα δεδομένα σε JsonObject  
 jsonObject = parser.parse(jsonData).getAsJsonObject();  
 // Ελέγχουμε αν το JsonObject είναι κενό ή null  
 if (jsonObject == null || jsonObject.size() == 0 || jsonObject.isJsonNull()) {  
  
 return null;  
 }  
  
 // Αντλούμε το στοιχείο με όνομα "meals" από το JsonObject  
 JsonElement mealsElement = jsonObject.get("meals");  
 // Ελέγχουμε εάν το JsonElement είναι null  
 if (mealsElement == null || mealsElement.isJsonNull()) {  
 // Επιστρέφουμε null εάν το JsonElement είναι null  
 return null;  
 }  
 // Μετατρέπουμε το JsonElement σε έναν πίνακα JsonArray  
 JsonArray mealsArray = mealsElement.getAsJsonArray();  
  
 // Ελέγχουμε εάν ο πίνακας JsonArray είναι null ή κενός  
 if (mealsArray == null || mealsArray.size() == 0 || mealsArray.isJsonNull()) {  
 // Επιστρέφουμε null εάν ο πίνακας JsonArray είναι null ή κενός  
 return null;  
 }  
  
 // Επιλέγουμε το πρώτο γεύμα από τον πίνακα JsonArray  
 jsonObject = mealsArray.get(0).getAsJsonObject();  
 } catch (Exception e) {  
  
 }  
 // Επιστρέφουμε το αντικείμενο JsonObject που περιέχει τα δεδομένα  
 return jsonObject;  
 }  
  
 //Μέθοδος που επιστρέφει μια λίστα με όλες τις διαθέσιμες κατηγορίες γευμάτων από την API.  
 public static ArrayList<MealCategory> getAllMealCategories() {  
 // Δημιουργία μιας νέας λίστας MealCategory  
 ArrayList<MealCategory> mealsArray = new ArrayList<MealCategory>();  
 try {  
 // Δημιουργία μιας νέας σύνδεσης με το API και λήψη των δεδομένων για τις κατηγορίες  
 String jsonData = APIRequest.getData(apiCategoriesLink);  
 JsonParser parser = new JsonParser();  
  
 // Μετατροπή των δεδομένων σε JsonObject  
 JsonObject jsonObject = parser.parse(jsonData).getAsJsonObject();  
 // Έλεγχος για το αν υπάρχουν δεδομένα στο πεδίο "categories"  
 if (!jsonObject.get("categories").isJsonNull()) {  
 // Λήψη του πίνακα "categories" από το JsonObject  
 JsonArray categoryArray = jsonObject.get("categories").getAsJsonArray();  
  
 // Επιστροφή όλων των κατηγοριών στη λίστα mealsArray  
 for (int i = 0; i < categoryArray.size(); i++) {  
 // Δημιουργία μιας νέας κατηγορίας φαγητού από το JsonObject  
 builder.setPrettyPrinting();  
 Gson gson = builder.create();  
 JsonObject category = categoryArray.get(i).getAsJsonObject();  
 mealsArray.add(gson.fromJson(category, MealCategory.class));  
 }  
 return mealsArray;  
 } else {  
  
 }  
 } catch (Exception e) {  
  
 }  
 //Επιστρέφει μια λίστα από MealCategory αντικείμενα που περιέχουν όλες τις κατηγορίες γευμάτων  
 return mealsArray;  
 }  
  
 // Μέθοδος που επιστρέφει μια λίστα με όλα τα γεύματα που ανήκουν σε μια συγκεκριμένη περιοχή  
 public static ArrayList<Meal> getAllMealByArea(String area) {  
 ArrayList<Meal> mealsArray = new ArrayList<>();  
  
 try {  
 // Λαμβάνουμε τα δεδομένα των γευμάτων από το API με βάση την περιοχή  
 String jsonData = APIRequest.getData(areaMeal + area);  
  
 // Δημιουργούμε έναν JsonParser για να αναλύσουμε τα δεδομένα JSON  
 JsonParser parser = new JsonParser();  
 JsonObject jsonObject = parser.parse(jsonData).getAsJsonObject();  
  
 if (jsonObject.has("meals")) {  
 // Λαμβάνουμε τον πίνακα με τα γεύματα από το JsonObject  
 JsonArray mealsJsonArray = jsonObject.getAsJsonArray("meals");  
 // Δημιουργούμε ένα Gson object για να μετατρέψουμε το JSON σε αντικείμενα Meal  
 Gson gson = new GsonBuilder().setPrettyPrinting().create();  
 // Προσθέτουμε τα αντικείμενα Meal στη λίστα mealsArray  
 for (int i = 0; i < mealsJsonArray.size(); i++) {  
 JsonObject mealJson = mealsJsonArray.get(i).getAsJsonObject();  
 Meal meal = gson.fromJson(mealJson, Meal.class);  
 mealsArray.add(meal);  
 }  
 }  
 } catch (Exception e) {  
  
 }  
 // Επιστρέφουμε τη λίστα με τα αντικείμενα Meal  
 return mealsArray;  
 }  
  
 //Μέθοδος που επιστρέφει μια λίστα με όλες τις περιοχές που υποστηρίζονται από το API  
 public static ArrayList<String> getAllArea() {  
 // Δημιουργία λίστας για τις περιοχές  
 ArrayList<String> areaArray = new ArrayList<String>();  
 try {  
 // Ανάκτηση δεδομένων από το API  
 String jsonData = APIRequest.getData(areaApi);  
 JsonParser parser = new JsonParser();  
 JsonObject jsonObject = parser.parse(jsonData).getAsJsonObject();  
  
 // Έλεγχος για το αν υπάρχει το πεδίο "meals" στο αντικείμενο JSON  
 if (!jsonObject.get("meals").isJsonNull()) {  
 // Ανάκτηση του πίνακα "meals" από το αντικείμενο JSON  
 JsonArray categoryArray = jsonObject.get("meals").getAsJsonArray();  
  
 // Προσθήκη κάθε περιοχής στην λίστα  
 for (int i = 0; i < categoryArray.size(); i++) {  
 areaArray.add(categoryArray.get(i).getAsJsonObject().get("strArea").getAsString());  
  
 }  
 return areaArray;  
 } else {  
  
 }  
 } catch (Exception e) {  
  
 }  
 // Επιστρέφει τη λίστα με τις περιοχές  
 return areaArray;  
 }  
  
 //Μέθοδος που επιστρέφει απο την ΑΡΙ μιά λίστα με όλα τα διαθέσιμα συστατικά και τις περιγραφές τους.  
 public static ArrayList<String[]> getAllIngredient() {  
 ArrayList<String[]> ingredientArray = new ArrayList<String[]>();  
 try {  
 // Ανάκτηση του JSON από το API  
 String jsonData = APIRequest.getData(ingredientApi);  
 JsonParser parser = new JsonParser();  
  
 // Μετατροπή του JSON σε JsonObject  
 JsonObject jsonObject = parser.parse(jsonData).getAsJsonObject();  
 // Έλεγχος εάν το πεδίο "meals" υπάρχει και δεν είναι κενό  
 if (!jsonObject.get("meals").isJsonNull()) {  
 // Ανάκτηση του πίνακα με τα συστατικά από το JsonObject  
 JsonArray ingreJson = jsonObject.get("meals").getAsJsonArray();  
  
 // Διατρέχουμε τον πίνακα με τα συστατικά  
 for (int i = 0; i < ingreJson.size(); i++) {  
 // Έλεγχος εάν το πεδίο "strIngredient" και "strDescription" υπάρχουν και δεν είναι κενά  
 if ((!ingreJson.get(i).getAsJsonObject().get("strIngredient").isJsonNull())) {  
 // Ανάκτηση του ονόματος και της περιγραφής του συστατικού  
 String strIngredient = ingreJson.get(i).getAsJsonObject().get("strIngredient").getAsString();  
 String strDescription = "";  
 if (ingreJson.get(i).getAsJsonObject().get("strDescription").isJsonNull()) {  
 strDescription = "";  
 } else {  
 strDescription = ingreJson.get(i).getAsJsonObject().get("strDescription").getAsString();  
 }  
  
 // Δημιουργία πίνακα με τα δεδομένα του συστατικού  
 String[] row = {strIngredient, strDescription};  
 // Προσθήκη του πίνακα στη λίστα με τα συστατικά  
 ingredientArray.add(row);  
  
 }  
  
 }  
 // Επιστροφή της λίστας με τα συστατικά  
 return ingredientArray;  
 } else {  
  
 }  
 } catch (Exception e) {  
 System.out.println("Error: " + e.getMessage());  
 }  
 //Επιστρέφει λίστα με όλα τα διαθέσιμα συστατικά και τις περιγραφές τους  
 return ingredientArray;  
 }  
  
 //Μέθοδος που επιστρέφει ένα JsonObject που περιέχει τα στοιχεία μιας συνταγής με βάση το όνομα της συνταγής.  
 public static JsonObject getMealJsonString(String mealName) {  
 JsonObject jsonObject = null;  
 try {  
 // Λήψη δεδομένων μέσω APIRequest  
 String jsonData = APIRequest.getData(apiLink + mealName);  
 // Δημιουργία ενός JsonParser  
 JsonParser parser = new JsonParser();  
  
 // Μετατροπή των δεδομένων σε JsonObject  
 jsonObject = parser.parse(jsonData).getAsJsonObject();  
 // Έλεγχος εάν το JsonObject είναι null ή έχει μέγεθος 0 ή είναι null  
 if (jsonObject == null || jsonObject.size() == 0 || jsonObject.isJsonNull()) {  
 // Επιστροφή null εάν το JsonObject είναι null ή έχει μέγεθος 0  
 return null;  
 }  
  
 // Λήψη του JsonElement με τo κλειδί "meals"  
 JsonElement mealsElement = jsonObject.get("meals");  
 if (mealsElement == null || mealsElement.isJsonNull()) {  
 // Επιστροφή null εάν το JsonElement είναι null  
 return null;  
 }  
 // Μετατροπή του JsonElement σε JsonArray  
 JsonArray mealsArray = mealsElement.getAsJsonArray();  
  
 // Έλεγχος εάν το JsonArray είναι null ή έχει μέγεθος 0  
 if (mealsArray == null || mealsArray.size() == 0 || mealsArray.isJsonNull()) {  
 // Επιστροφή null εάν το JsonArray είναι null ή έχει μέγεθος 0  
 return null;  
 }  
  
 // Λήψη του πρώτου JsonObject από το JsonArray  
 jsonObject = mealsArray.get(0).getAsJsonObject();  
 } catch (Exception e) {  
  
 }  
 // Επιστροφή του JsonObject  
 return jsonObject;  
 }  
  
 //Μέθοδος που επιστρέφει ένα αντικείμενο Meal με το όνομα του γεύματος  
 public static Meal logMealObjByName(JsonObject mealJson) {  
 // Δημιουργία νέου αντικειμένου Meal με το όνομα του γεύματος από το JsonObject που δόθηκε ως παράμετρος  
 Meal meal = new Meal(mealJson.get("strMeal").getAsString());  
 // Επιστροφή του αντικειμένου Meal  
 return meal;  
 }  
  
 //Μέθοδος που επιστρέφει ενα αντικείμενο Meal που δημιουργήθηκε από το JsonObject  
 public static Meal createMealFromJson(JsonObject mealJson) {  
 // Ορισμός του builder για τη μετατροπή του JSON σε String με μορφοποίηση.  
 builder.setPrettyPrinting();  
 // Δημιουργία του gson αντικειμένου με τη χρήση του builder για τη μετατροπή του JSON σε αντικείμενο Meal.  
 Gson gson = builder.create();  
 // Επιστρέφει το αντικείμενο Meal που δημιουργήθηκε από το JsonObject  
 return gson.fromJson(mealJson, Meal.class);  
 }  
  
 //Μέθοδος που δημιουργεί ένα αντικείμενο γεύματος Meal βάσει ενός μοναδικού αριθμού id  
 public static Meal createMealObjByID(String idMeal) {  
 // Παίρνουμε τα δεδομένα από το API  
 String jsonData = APIRequest.getData(apiIDLink + idMeal);  
  
 try {  
 // Μετατρέπουμε τα δεδομένα σε JsonObject  
 JsonObject jsonObject = parser.parse(jsonData).getAsJsonObject();  
  
 // Έλεγχος εάν το JsonObject είναι κενό  
 if (jsonObject.get("meals").isJsonNull()) {  
 return null;  
 }  
  
 // Παίρνουμε το πρώτο γεύμα από τον πίνακα με τα γεύματα  
 JsonArray mealsArray = jsonObject.get("meals").getAsJsonArray();  
 JsonObject firstMeal = mealsArray.get(0).getAsJsonObject();  
  
 // Δημιουργούμε ένα νέο Gson object και καλούμε την μέθοδο fromJson για να μετατρέψουμε τα δεδομένα σε ένα αντικείμενο Meal  
 builder.setPrettyPrinting();  
 Gson gson = builder.create();  
 return gson.fromJson(firstMeal, Meal.class);  
 } catch (Exception e) {  
  
 return null;  
 }  
 }  
  
 //Mέθοδος που λαμβάνει μια κατηγορία γευμάτων ως είσοδο και επιστρέφει μια λίστα με όλα τα γεύματα που ανήκουν σε αυτήν την κατηγορία.   
 public static ArrayList<Meal> getAllMealByCategory(String category) {  
 // Δημιουργία μιας κενής λίστας που θα περιέχει τα γεύματα της κατηγορίας  
 ArrayList<Meal> mealsArray = new ArrayList<>();  
  
 try {  
 // Λαμβάνουμε τα δεδομένα του αιτήματος από το API  
 String jsonData = APIRequest.getData(apiMealOfCategories + category);  
  
 // Χρησιμοποιούμε τη βιβλιοθήκη Gson για να μετατρέψουμε τα δεδομένα από JSON σε Java objects   
 JsonParser parser = new JsonParser();  
 JsonObject jsonObject = parser.parse(jsonData).getAsJsonObject();  
  
 // Έλεγχος εάν υπάρχουν γεύματα για την επιλεγμένη κατηγορία  
 if (jsonObject.has("meals")) {  
 // Παίρνουμε τον πίνακα με τα γεύματα από το JSON  
 JsonArray mealsJsonArray = jsonObject.getAsJsonArray("meals");  
  
 // Χρησιμοποιούμε τη βιβλιοθήκη Gson για να μετατρέψουμε τα δεδομένα του κάθε γεύματος σε ένα αντικείμενο Meal και να τα προσθέσουμε στη λίστα με τα γεύματα  
 Gson gson = new GsonBuilder().setPrettyPrinting().create();  
 for (int i = 0; i < mealsJsonArray.size(); i++) {  
 JsonObject mealJson = mealsJsonArray.get(i).getAsJsonObject();  
 Meal meal = gson.fromJson(mealJson, Meal.class);  
 mealsArray.add(meal);  
 }  
 }  
 } catch (Exception e) {  
  
 }  
 //Επιστρέφει ένα ArrayList από Meal αντικείμενα, που αντιστοιχούν σε γεύματα που ανήκουν στην κατηγορία που δόθηκε ως παράμετρος.  
 return mealsArray;  
 }  
  
 //Μέθοδος που αναζητά όλα τα γεύματα που περιέχουν ένα συγκεκριμένο συστατικό (ingredient)  
 public static ArrayList<Meal> getAllMealByIngridient(String ingredient) {  
 ArrayList<Meal> mealsArray = new ArrayList<>();  
  
 try {  
 //Ανάκτηση δεδομένων από την εξωτερική API με βάση το όνομα του υλικού  
 String jsonData = APIRequest.getData(ingredientgetmeals + ingredient);  
  
 JsonParser parser = new JsonParser();  
 JsonObject jsonObject = parser.parse(jsonData).getAsJsonObject();  
  
 //Έλεγχος για την ύπαρξη του πεδίου "meals" στο JSON αποτέλεσμα  
 if (jsonObject.has("meals")) {  
 JsonArray mealsJsonArray = jsonObject.getAsJsonArray("meals");  
 Gson gson = new GsonBuilder().setPrettyPrinting().create();  
  
 //Δημιουργία αντικειμένων Meal από τα JSON αντικείμενα της λίστας "meals"  
 for (int i = 0; i < mealsJsonArray.size(); i++) {  
 JsonObject mealJson = mealsJsonArray.get(i).getAsJsonObject();  
 Meal meal = gson.fromJson(mealJson, Meal.class);  
 mealsArray.add(meal);  
 }  
 }  
 } catch (Exception e) {  
  
 }  
 //Επιστροφή της λίστας με τα αντικείμενα Meal  
 return mealsArray;  
 }  
  
 //Μέθοδος που δέχεται ένα JsonObject που περιέχει πληροφορίες για γεύματα που αναζητούνται και επιστρέφει   
 //μια λίστα από αντικείμενα Meal που αντιστοιχούν σε αυτά τα γεύματα.  
 public static ArrayList<Meal> getAllMealBySearch(JsonObject mealsJsonObject) {  
 ArrayList<Meal> mealsArray = new ArrayList<>();  
  
 try {  
 // Έλεγχος αν το JsonObject περιέχει το πεδίο "meals"  
 if (mealsJsonObject.has("meals")) {  
 // Ανάκτηση του JsonArray με τα γεύματα  
 JsonArray mealsJsonArray = mealsJsonObject.getAsJsonArray("meals");  
 // Δημιουργία ενός Gson object με την μέθοδο setPrettyPrinting για ένα ορατό JSON format  
 Gson gson = new GsonBuilder().setPrettyPrinting().create();  
 // Προσπέλαση του JsonArray και δημιουργία ενός αντικειμένου Meal για κάθε JsonObject  
 for (int i = 0; i < mealsJsonArray.size(); i++) {  
 JsonObject mealJson = mealsJsonArray.get(i).getAsJsonObject();  
 Meal meal = gson.fromJson(mealJson, Meal.class);  
 mealsArray.add(meal);  
 }  
 }  
 } catch (Exception e) {  
  
 }  
 // Επιστροφή της λίστας με τα γεύματα  
 return mealsArray;  
 }  
  
 //Μέθοδος που επιστρέφει το JsonObject με όλες τις πληροφορίες για ένα γεύμα με βάση το όνομά του  
 public static JsonObject getAllMealsJsonString(String mealName) {  
 JsonObject jsonObject = null;  
 try {  
 // Ανακτά το δεδομένα σε μορφή JSON με βάση το όνομα του γεύματος  
 String jsonData = APIRequest.getData(apiLink + mealName);  
 JsonParser parser = new JsonParser();  
  
 // Μετατροπή του JSON σε JsonObject  
 jsonObject = parser.parse(jsonData).getAsJsonObject();  
 // Έλεγχος για τυχόν κενό JsonObject  
 if (jsonObject == null || jsonObject.size() == 0 || jsonObject.isJsonNull()) {  
  
 return null;  
 }  
  
 // Ανάκτηση του JsonElement με τα γεύματα  
 JsonElement mealsElement = jsonObject.get("meals");  
 // Έλεγχος για τυχόν κενό JsonElement  
 if (mealsElement == null || mealsElement.isJsonNull()) {  
  
 return null;  
 }  
 // Μετατροπή του JsonElement με τα γεύματα σε JsonArray  
 JsonArray mealsArray = mealsElement.getAsJsonArray();  
  
 // Έλεγχος για τυχόν κενό JsonArray  
 if (mealsArray == null || mealsArray.size() == 0 || mealsArray.isJsonNull()) {  
  
 return null;  
 }  
  
 } catch (Exception e) {  
  
 }  
 // Επιστροφή του JsonObject με τις πληροφορίες για το γεύμα  
 return jsonObject;  
 }  
  
}

package mealsapp.DB. DatabaseCreation.java

package mealsapp.DB;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
  
public class DatabaseCreation {  
 // Μέθοδος για δημιουργία της βάσης δεδομένων  
 public static void createDB() {  
 // Ορισμός του ονόματος της βάσης δεδομένων και του URL σύνδεσης  
 String dbName = "MealsDB";  
 String connectionURL = "jdbc:derby:MealsDB;create=true;user=mealsdb;password=mealsdb";  
  
 try (Connection conn = DriverManager.getConnection(connectionURL)) {  
 // Επιτυχής σύνδεση με τη βάση δεδομένων  
 System.out.println("Connected to database " + dbName);  
 } catch (SQLException e) {  
 if (e.getSQLState().equals("XJ004")) {  
 // Η βάση δεδομένων δεν υπάρχει και πρέπει να δημιουργηθεί  
 System.out.println("Database " + dbName + " does not exist, creating...");  
 createDatabase(connectionURL);  
 } else {  
 // Σφάλμα κατά τη σύνδεση με τη βάση δεδομένων  
 e.printStackTrace();  
 }  
 }  
 }  
 // Μέθοδος για δημιουργία της βάσης δεδομένων  
 private static void createDatabase(String connectionURL) {  
 try (Connection conn = DriverManager.getConnection(connectionURL)) {  
 // Επιτυχής δημιουργία της βάσης δεδομένων  
 System.out.println("Database created successfully");  
 } catch (SQLException e) {  
 // Σφάλμα κατά τη δημιουργία της βάσης δεδομένων  
 e.printStackTrace();  
 }  
 }  
  
}

package mealsapp.DB.mealsCategoryQueryOperations.java

package mealsapp.DB;  
  
import mealsapp.MealClasses.MealCategory;  
import java.util.List;  
import javax.persistence.EntityManager;  
import javax.persistence.EntityManagerFactory;  
import javax.persistence.Persistence;  
import javax.persistence.NoResultException;  
import javax.persistence.TypedQuery;  
  
public class mealsCategoryQueryOperations {  
  
 // Μέθοδος για την επιστροφή μιας ταξινομημένης λίστας MealCategory βάση του id  
 public static List<MealCategory> sortByIdcategory() {  
 // Δημιουργία entity manager factory και entity manager για την επικοινωνία με τη βάση δεδομένων  
 EntityManagerFactory emf = Persistence.createEntityManagerFactory("MealsAppPU");  
 EntityManager em = emf.createEntityManager();  
  
 // Δημιουργία του ερωτήματος για την ταξινόμηση των MealCategory αντικειμένων βάσει του idcategory  
 TypedQuery<MealCategory> query = em.createNamedQuery("MealCategory.sortByIdcategory", MealCategory.class);  
 List<MealCategory> resultList = query.getResultList();  
  
 // Κλείσιμο του entity manager και του entity manager factory  
 em.close();  
 emf.close();  
  
 // Επιστροφή της ταξινομημένης λίστας  
 return resultList;  
 }  
  
 // Μέθοδος για την εισαγωγή της λίστας MealCategory στη βάση δεδομένων  
 public static void importMealCategories(List<MealCategory> mealCategories) {  
 // Δημιουργία entity manager factory και entity manager για την επικοινωνία με τη βάση δεδομένων  
 EntityManagerFactory emf = Persistence.createEntityManagerFactory("MealsAppPU");  
 EntityManager em = emf.createEntityManager();  
  
 // Εισαγωγή κάθε MealCategory αντικειμένου στη βάση δεδομένων (αν δεν υπάρχει ήδη)  
 for (int i = 0; i < mealCategories.size(); i++) {  
  
 try {  
 // Έλεγχος αν υπάρχει ήδη στη βάση δεδομένων κάποιο MealCategory με το ίδιο category και dtype  
 List<MealCategory> existingMealCategories = em.createNamedQuery("MealCategory.findBycategoryAndDtype", MealCategory.class)  
 .setParameter("strcategory", mealCategories.get(i).getStrcategory())  
 .getResultList();  
 // Αν υπάρχει ήδη, συνεχίζεται η επανάληψη χωρίς να γίνεται εισαγωγή του στη βάση  
 if (!existingMealCategories.isEmpty()) {  
  
 continue;  
 }  
 } catch (NoResultException e) {  
  
 }  
  
 // Ξεκινά η διαδικασία για την προσθήκη του MealCategory στη βάση δεδομένων  
 em.getTransaction().begin();  
 // Εισαγωγή του MealCategory στη βάση δεδομένων  
 em.persist(mealCategories.get(i));  
 // Ολοκλήρωση της διαδικασίας προσθήκης  
 em.getTransaction().commit();  
  
 }  
  
 // Κλείσιμο του entity manager και του entity manager factory  
 em.close();  
 emf.close();  
 }  
  
 // Μέθοδος για την επιστροφή μιας λίστας όλων των MealCategory αντικειμένων από τη βάση δεδομένων  
 public static List<MealCategory> getAllMealCategory() {  
 // Δημιουργία entity manager factory και entity manager για την επικοινωνία με τη βάση δεδομένων  
 EntityManagerFactory emf = Persistence.createEntityManagerFactory("MealsAppPU");  
 EntityManager em = emf.createEntityManager();  
  
 // Δημιουργία ερωτήματος για την επιστροφή όλων των MealCategory αντικειμένων από τη βάση δεδομένων  
 TypedQuery<MealCategory> query = em.createNamedQuery("MealCategory.findAll", MealCategory.class);  
 List<MealCategory> resultList = query.getResultList();  
  
 // Κλείσιμο του entity manager και του entity manager factory  
 em.close();  
 emf.close();  
  
 // Επιστροφή της λίστας όλων των MealCategory αντικειμένων από τη βάση δεδομένων  
 return resultList;  
 }  
  
 // Μέθοδος για την επιστροφή ενός μόνο MealCategory αντικειμένου  
 public static MealCategory getSingleCategory(String strCategory) {  
  
 // Δημιουργία entity manager factory και entity manager για την επικοινωνία με τη βάση δεδομένων  
 EntityManagerFactory emf = Persistence.createEntityManagerFactory("MealsAppPU");  
 EntityManager em = emf.createEntityManager();  
  
 MealCategory mealCategory = null;  
  
 try {  
 // Χρήση του named query για την εύρεση του αντικειμένου MealCategory βάσει του strCategory του  
 mealCategory = em.createNamedQuery("MealCategory.findByStrcategory", MealCategory.class)  
 .setParameter("strcategory", strCategory)  
 .getSingleResult();  
 em.getTransaction().begin();  
 em.getTransaction().commit();  
 System.out.println("Meal Found Successfully");  
 } catch (NoResultException e) {  
  
 } finally {  
 // Κλείσιμο του entity manager και του entity manager factory  
 em.close();  
 emf.close();  
  
 }  
 // Επιστροφή του αντικειμένου MealCategory που βρέθηκε ή null αν δεν βρέθηκε κάποιο αντικείμενο  
 return mealCategory;  
  
 }  
}

package mealsapp.DB. mealsQueryOperations.java

package mealsapp.DB;  
  
import mealsapp.MealClasses.Meal;  
import java.util.List;  
import javax.persistence.EntityManager;  
import javax.persistence.EntityManagerFactory;  
import javax.persistence.NoResultException;  
import javax.persistence.Persistence;  
  
public class mealsQueryOperations {  
  
 //Μέθοδος για την εισαγωγή ενός νέου αντικειμένου Meal στη βάση δεδομένων  
 public static void insertNewMeal(Meal newMeal) {  
 // Δημιουργία entity manager factory και entity manager για την επικοινωνία με τη βάση δεδομένων  
 EntityManagerFactory emf = Persistence.createEntityManagerFactory("MealsAppPU");   
 EntityManager em = emf.createEntityManager();   
  
 try {  
 // Χρήση του named query για την εύρεση ενός αντικειμένου Meal βάσει του strmeal του  
 newMeal = em.createNamedQuery("Meal.findByStrmeal", Meal.class)  
 .setParameter("strmeal", newMeal.getStrmeal())  
 .getSingleResult();   
  
 if (newMeal != null) {  
 // Αν υπάρχει ήδη αντικείμενο Meal με το ίδιο strmeal, δε γίνεται εισαγωγή στη βάση και επιστρέφεται η συνάρτηση  
 return;  
 }  
 } catch (NoResultException e) {  
 // Αν δε βρεθεί κάποιο αντικείμενο Meal με το συγκεκριμένο strmeal, δε γίνεται τίποτα και συνεχίζεται η εισαγωγή στη βάση  
 }  
  
 try {  
 // Χρήση του named query "Meal.findBygetIdcategory" για την εύρεση όλων των Meal αντικειμένων που έχουν το ίδιο idcategory με το νέο αντικείμενο  
 List<Meal> newMea2Meal = em.createNamedQuery("Meal.findBygetIdcategory", Meal.class)  
 .setParameter("idcategory", newMeal.getIdcategory())  
 .getResultList();   
  
 if (newMea2Meal != null) {  
 // Αν υπάρχουν ήδη αντικείμενα Meal με το ίδιο idcategory με το νέο αντικείμενο, αυτό σημαίνει ότι το νέο αντικείμενο έχει εισαχθεί από το χρήστη  
 // οπότε θέτουμε τα απαραίτητα πεδία (timesofvisit και idmeal) και κάνουμε merge  
 em.getTransaction().begin();  
 newMeal.setTimesofvisit(0);  
 newMeal.setIdmeal("SavedByUser");  
 em.merge(newMeal);  
 em.getTransaction().commit();  
 em.close();  
 return;  
 }  
 } catch (NoResultException e) {  
 // Αν δε βρεθεί κάποιο αντικείμενο Meal με το συγκεκριμένο idcategory, δε γίνεται τίποτα και συνεχίζεται η εισαγωγή στη βάση  
 }  
 // Αν δε βρεθεί ούτε αντικείμενο Meal με το ίδιο strmeal ούτε αντικείμενο Meal με το ίδιο idcategory, εισάγουμε το νέο αντικείμενο Meal στη βάση  
 em.getTransaction().begin();   
 em.persist(newMeal);   
 em.getTransaction().commit();   
   
 em.close();   
 emf.close();   
 }  
  
 //Μέθοδος για τη διαγραφή ενός αντικειμένου Meal   
 public static void deleteSingleMeal(String strMeal) {  
 // Δημιουργία entity manager factory και entity manager για την επικοινωνία με τη βάση δεδομένων  
 EntityManagerFactory emf = Persistence.createEntityManagerFactory("MealsAppPU");   
 EntityManager em = emf.createEntityManager();   
 try {  
 // Χρήση του named query για την εύρεση του αντικειμένου Meal βάσει του συγκεκριμένου strmeal  
 Meal meal = em.createNamedQuery("Meal.findByStrmeal", Meal.class).setParameter("strmeal", strMeal).getSingleResult();   
   
 em.getTransaction().begin();   
   
 // Διαγραφή του αντικειμένου Meal από τη βάση  
 em.remove(meal);   
 em.getTransaction().commit();   
   
 } catch (NoResultException e) {  
 // Αν δε βρεθεί κάποιο αντικείμενο Meal με το συγκεκριμένο strmeal, δε γίνεται τίποτα και συνεχίζεται η συνάρτηση  
 } finally {  
 // Κλείσιμο του entity manager και του entity manager factory  
 em.close();   
 emf.close();   
 }  
 }  
  
 //Μέθοδος για τον έλεγχο ύπαρξης στη βάση δεδομένων ενός αντικειμένου Meal  
 public static boolean findMealByStrMeal(String strmeal) {  
 // Αρχικά θεωρούμε ότι το αντικείμενο Meal με το συγκεκριμένο strmeal δεν υπάρχει στη βάση  
 boolean result = false;   
   
 // Δημιουργία entity manager factory και entity manager για την επικοινωνία με τη βάση δεδομένων  
 EntityManagerFactory emf = Persistence.createEntityManagerFactory("MealsAppPU");  
 EntityManager em = emf.createEntityManager();  
  
 try {  
 // Χρήση του named query για την εύρεση του αντικειμένου Meal βάσει του συγκεκριμένου strmeal  
 Meal meal = em.createNamedQuery("Meal.findByStrmeal", Meal.class)  
 .setParameter("strmeal", strmeal)  
 .getSingleResult();  
 if (meal.getStrmeal().equals(strmeal)) {  
 // Αν βρεθεί αντικείμενο Meal με το συγκεκριμένο strmeal, τότε το result γίνεται true  
 result = true;   
 }  
 em.getTransaction().begin();  
 em.getTransaction().commit();  
   
 } catch (NoResultException e) {  
 // Αν δε βρεθεί κάποιο αντικείμενο Meal με το συγκεκριμένο strmeal, δε γίνεται τίποτα και συνεχίζεται η συνάρτηση  
 } finally {  
 // Κλείσιμο του entity manager και του entity manager factory και επιστροφή του result  
 em.close();  
 emf.close();  
 return result;   
 }  
 }  
  
 //Μέθοδος για τον έλεγχο εαν το αντικείμενο Meal έχει αποθηκευτεί από τον χρήστη  
 public static boolean findIfMealIsSavedByUser(String meal) {  
 // Αρχικά θεωρούμε ότι το αντικείμενο Meal δεν έχει αποθηκευτεί από τον χρήστη  
 boolean result = false;  
   
 // Δημιουργία entity manager factory και entity manager για την επικοινωνία με τη βάση δεδομένων  
 EntityManagerFactory emf = Persistence.createEntityManagerFactory("MealsAppPU");  
 EntityManager em = emf.createEntityManager();  
  
 try {  
 // Χρήση του named query για την εύρεση του αντικειμένου Meal βάσει του συγκεκριμένου strmeal  
 Meal searchMeal = em.createNamedQuery("Meal.findByStrmeal", Meal.class)  
 .setParameter("strmeal", meal)  
 .getSingleResult();  
 if (searchMeal.getIdmeal() == null) {  
 // Αν το idmeal είναι null, τότε το αντικείμενο Meal δεν έχει αποθηκευτεί από τον χρήστη  
 result = false;   
   
 } else {  
 // Αν το idmeal δεν είναι null, τότε το αντικείμενο Meal έχει αποθηκευτεί από τον χρήστη  
 result = true;  
   
 }  
 } catch (NoResultException e) {  
 // Αν δε βρεθεί κάποιο αντικείμενο Meal με το συγκεκριμένο strmeal, δε γίνεται τίποτα και συνεχίζεται η συνάρτηση  
 } finally {  
 // Κλείσιμο του entity manager και του entity manager factory και επιστροφή του result  
 em.close();  
 emf.close();  
 return result;   
 }  
 }  
  
 //Μέθοδος για την αύξηση του αριθμού επισκέψεων ενός γεύματος  
 public static void addVisit(String strMeal) {  
   
 // Δημιουργία entity manager factory και entity manager για την επικοινωνία με τη βάση δεδομένων  
 EntityManagerFactory emf = Persistence.createEntityManagerFactory("MealsAppPU");  
 EntityManager em = emf.createEntityManager();  
 em.getTransaction().begin();  
  
 try {  
 // Χρήση του named query για την εύρεση του αντικειμένου Meal βάσει του συγκεκριμένου strmeal  
 Meal meal = em.createNamedQuery("Meal.findByStrmeal", Meal.class)  
 .setParameter("strmeal", strMeal)  
 .getSingleResult();  
  
 // Αύξηση του αριθμού επισκέψεων του αντικειμένου Meal κατά 1  
 meal.setTimesofvisit(meal.getTimesofvisit() + 1);  
  
 em.persist(meal);  
 em.getTransaction().commit();  
   
  
 } catch (NoResultException e) {  
 // Αν δε βρεθεί κάποιο αντικείμενο Meal με το συγκεκριμένο strmeal, δε γίνεται τίποτα και συνεχίζεται η συνάρτηση  
 } finally {  
 // Κλείσιμο του entity manager και του entity manager factory  
 em.close();  
 emf.close();  
 }  
  
 }  
  
 //Μέθοδος για την επιστροφή μιας λίστας με τα αντικείμενα Meal ταξινομημένα κατά φθίνουσα σειρά βάσει του αριθμού επισκέψεων  
 public static List<Meal> getbyDecOrder() {  
  
 // Δημιουργία entity manager factory και entity manager για την επικοινωνία με τη βάση δεδομένων  
 EntityManagerFactory emf = Persistence.createEntityManagerFactory("MealsAppPU");  
 EntityManager em = emf.createEntityManager();  
 em.getTransaction().begin();  
  
 // Δήλωση μιας μεταβλητής που θα περιέχει τα αντικείμενα Meal που θα επιστραφούν από το query  
 List<Meal> meals = null;  
  
 try {  
 // Χρήση του named query για την επιστροφή των αντικειμένων Meal ταξινομημένων κατά φθίνουσα σειρά βάσει του αριθμού επισκέψεων  
 meals = em.createNamedQuery("Meal.sortByTimesofVisit", Meal.class).getResultList();  
 } catch (NoResultException e) {  
 // Αν δε βρεθεί κανένα αντικείμενο Meal, δε γίνεται τίποτα και συνεχίζει  
 } finally {  
 // Commit και κλείσιμο του entity manager και του entity manager factory  
 em.getTransaction().commit();  
 em.close();  
 emf.close();  
 }  
  
 // Επιστροφή της λίστας με τα αντικείμενα Meal  
 return meals;  
 }  
  
 //Μέθοδος για την επιστροφή μιας λίστας παρόμοιων γευμάτων από τη βάση δεδομένων  
 public static List<Meal> getLikeMealsFromDB(String strmeal) {  
   
 // Δημιουργία entity manager factory και entity manager για την επικοινωνία με τη βάση δεδομένων  
 EntityManagerFactory emf = Persistence.createEntityManagerFactory("MealsAppPU");  
 EntityManager em = emf.createEntityManager();  
 em.getTransaction().begin();  
  
   
 List<Meal> meals = null;  
  
 try {  
 // Αναζήτηση παρόμοιων γευμάτων με βάση το κείμενο αναζήτησης strmeal, είτε με μικρά γράμματα είτε με πρώτο γράμμα κεφαλαίο  
 if (!strmeal.isEmpty()) {  
 meals = em.createNamedQuery("Meal.getLikeStrMeal", Meal.class)  
 .setParameter("strmeal", "%" + strmeal.toLowerCase() + "%")  
 .setParameter("strmeal", "%" + strmeal.substring(0, 1).toUpperCase() + strmeal.substring(1).toLowerCase() + "%")  
 .getResultList();  
 } else {  
 // Αν το κείμενο αναζήτησης είναι κενό, επιστρέφονται όλα τα γεύματα από τη βάση δεδομένων  
 meals = em.createNamedQuery("Meal.getLikeStrMeal", Meal.class)  
 .setParameter("strmeal", "%" + strmeal + "%").getResultList();  
 }  
 } catch (NoResultException e) {  
 // Σε περίπτωση που δεν βρεθεί κανένα παρόμοιο γεύμα, δεν γίνεται τίποτα  
 } finally {  
 // Commit και κλείσιμο του entity manager και του entity manager factory  
 em.getTransaction().commit();  
 em.close();  
 emf.close();  
 }  
  
 // Επιστροφή λίστας με τα παρόμοια γεύματα που βρέθηκαν στη βάση δεδομένων  
 return meals;  
  
 }  
  
 //Μέθοδος για την επιστροφή ενός γεύματος από τη βάση δεδομένων βάσει του ονόματός του  
 public static Meal getMealByStrMeal(String strmeal) {  
   
 // Δημιουργία entity manager factory και entity manager για την επικοινωνία με τη βάση δεδομένων  
 EntityManagerFactory emf = Persistence.createEntityManagerFactory("MealsAppPU");  
 EntityManager em = emf.createEntityManager();  
  
 Meal meal = null;  
  
 try {  
 // Αναζήτηση του γεύματος με βάση το όνομα του γεύματος   
 meal = em.createNamedQuery("Meal.findByStrmeal", Meal.class)  
 .setParameter("strmeal", strmeal)  
 .getSingleResult();  
  
 em.getTransaction().begin();  
 em.getTransaction().commit();  
   
 } catch (NoResultException e) {  
 // Σε περίπτωση που δεν βρεθεί κανένα γεύμα, δεν γίνεται τίποτα  
 } finally {  
 // Κλείσιμο entity manager και entity manager factory  
 em.close();  
 emf.close();  
   
 }  
 // Επιστροφή του γεύματος που βρέθηκε στη βάση δεδομένων  
 return meal;  
 }  
  
 //Μέθοδος για την αποθήκευση ενός γεύματος στη βάση δεδομένων  
 public static void saveMeal(Meal meal) {  
   
 // Δημιουργία entity manager factory και entity manager για την επικοινωνία με τη βάση δεδομένων  
 EntityManagerFactory emf = Persistence.createEntityManagerFactory("MealsAppPU");  
 EntityManager em = emf.createEntityManager();  
 em.getTransaction().begin();  
  
 try {  
 // Αναζήτηση του γεύματος στη βάση δεδομένων με βάση το όνομα του γεύματος  
 Meal oldmeal = em.createNamedQuery("Meal.findByStrmeal", Meal.class)  
 .setParameter("strmeal", meal.getStrmeal())  
 .getSingleResult();  
  
 // Ενημέρωση του πλήθους επισκέψεων στο γεύμα  
 meal.setTimesofvisit(oldmeal.getTimesofvisit());  
   
 // Ενημέρωση των πεδίων του παλιού γεύματος με τα νέα δεδομένα από το νέο γεύμα  
 oldmeal.setIdmeal(meal.getIdmeal());  
 oldmeal.setStrcategory(meal.getStrcategory());  
 oldmeal.setStrmeal(meal.getStrmeal());  
 oldmeal.setStrarea(meal.getStrarea());  
 oldmeal.setStrinstructions(meal.getStrinstructions());  
 oldmeal.setStrmealthumb(meal.getStrmealthumb());  
 oldmeal.setStryoutube(meal.getStryoutube());  
 oldmeal.setStringredient1(meal.getStringredient1());  
 oldmeal.setStringredient2(meal.getStringredient2());  
 oldmeal.setStringredient3(meal.getStringredient3());  
 oldmeal.setStringredient4(meal.getStringredient4());  
 oldmeal.setStringredient5(meal.getStringredient5());  
 oldmeal.setStringredient6(meal.getStringredient6());  
 oldmeal.setStringredient7(meal.getStringredient7());  
 oldmeal.setStringredient8(meal.getStringredient8());  
 oldmeal.setStringredient9(meal.getStringredient9());  
 oldmeal.setStringredient10(meal.getStringredient10());  
 oldmeal.setStringredient11(meal.getStringredient11());  
 oldmeal.setStringredient12(meal.getStringredient12());  
 oldmeal.setStringredient13(meal.getStringredient13());  
 oldmeal.setStringredient14(meal.getStringredient14());  
 oldmeal.setStringredient15(meal.getStringredient15());  
 oldmeal.setStringredient16(meal.getStringredient16());  
 oldmeal.setStringredient17(meal.getStringredient17());  
 oldmeal.setStringredient18(meal.getStringredient18());  
 oldmeal.setStringredient19(meal.getStringredient19());  
 oldmeal.setStringredient20(meal.getStringredient20());  
 oldmeal.setStrmeasure1(meal.getStrmeasure1());  
 oldmeal.setStrmeasure2(meal.getStrmeasure2());  
 oldmeal.setStrmeasure3(meal.getStrmeasure3());  
 oldmeal.setStrmeasure4(meal.getStrmeasure4());  
 oldmeal.setStrmeasure5(meal.getStrmeasure5());  
 oldmeal.setStrmeasure6(meal.getStrmeasure6());  
 oldmeal.setStrmeasure7(meal.getStrmeasure7());  
 oldmeal.setStrmeasure8(meal.getStrmeasure8());  
 oldmeal.setStrmeasure9(meal.getStrmeasure9());  
 oldmeal.setStrmeasure10(meal.getStrmeasure10());  
 oldmeal.setStrmeasure11(meal.getStrmeasure11());  
 oldmeal.setStrmeasure12(meal.getStrmeasure12());  
 oldmeal.setStrmeasure13(meal.getStrmeasure13());  
 oldmeal.setStrmeasure14(meal.getStrmeasure14());  
 oldmeal.setStrmeasure15(meal.getStrmeasure15());  
 oldmeal.setStrmeasure16(meal.getStrmeasure16());  
 oldmeal.setStrmeasure17(meal.getStrmeasure17());  
 oldmeal.setStrmeasure18(meal.getStrmeasure18());  
 oldmeal.setStrmeasure19(meal.getStrmeasure19());  
 oldmeal.setStrmeasure20(meal.getStrmeasure20());  
   
 // Αποθήκευση των αλλαγών στη βάση δεδομένων  
 em.merge(oldmeal);  
 em.getTransaction().commit();  
   
  
 } catch (NoResultException e) {  
 // Σε περίπτωση που δεν βρεθεί κανένα γεύμα, δεν γίνεται τίποτα  
 } finally {  
 // Κλείσιμο entity manager και entity manager factory  
 em.close();  
 emf.close();  
 }  
 }  
}

**Γ)**

**POJO κλάσεις.**

Όπως αναφέρθηκε υπάρχουν [**2 POJO**](https://makeinjava.com/serialization-deserialization-object-pojo-java-example/)κλάσεις στην εφαρμογή η parent class **MealCategory**, όπου δημιουργεί τύπους γευμάτων (Beef, Vegan κτλ) και η **Meal** class που δημιουργεί γεύματα (Moussaka κτλ). Για πρακτικούς λόγος αφαιρούμε τα Queries που δημιουργήθηκαν αυτόματα (αν και κάποια από αυτά χρησιμοποιούνται) και παραθέτουμε μόνο αυτά που γράψαμε εμείς.

package mealsapp.MealClasses. MealCategory.java

/\*  
Η κλάση MealCategory αναπαριστά μια κατηγορία γευμάτων και περιλαμβάνει πεδία για το ID,   
την περιγραφή, την εικόνα και την περιγραφή της κατηγορίας. Στον κώδικα χρησιμοποιούνται  
annotations για την περιγραφή της κλάσης και των πεδίων της, καθώς και named queries για   
τον εντοπισμό δεδομένων στη βάση δεδομένων.  
\*/  
package mealsapp.MealClasses;  
  
import java.io.Serializable;  
import javax.persistence.Basic;  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.Id;  
import javax.persistence.NamedQueries;  
import javax.persistence.NamedQuery;  
import javax.persistence.Inheritance;  
import javax.persistence.InheritanceType;  
import com.google.gson.annotations.SerializedName;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
  
  
@Entity  
  
@Inheritance(strategy = InheritanceType.SINGLE\_TABLE)  
  
@NamedQueries({  
 @NamedQuery(name = "MealCategory.findAll", query = "SELECT m FROM MealCategory m"),  
   
 @NamedQuery(name = "MealCategory.findByIdcategory", query = "SELECT m FROM MealCategory m WHERE m.idcategory = :idcategory"),  
   
 @NamedQuery(name = "MealCategory.findByStrcategory", query = "SELECT m FROM MealCategory m WHERE m.strcategory = :strcategory"),  
   
 @NamedQuery(name = "MealCategory.findByStrcategorythumb", query = "SELECT m FROM MealCategory m WHERE m.strcategorythumb = :strcategorythumb"),  
   
 @NamedQuery(name = "MealCategory.findByStrcategorydescription", query = "SELECT m FROM MealCategory m WHERE m.strcategorydescription = :strcategorydescription"),  
  
 @NamedQuery(name = "MealCategory.findBycategoryAndDtype", query = "SELECT m FROM MealCategory m WHERE m.strcategory = :strcategory AND TYPE(m) = MealCategory")  
  
})  
public class MealCategory implements Serializable {  
  
 private static final long serialVersionUID = 1L;  
  
 @Basic(optional = false)  
 @GeneratedValue(strategy = GenerationType.SEQUENCE)  
 @Id  
 @Column(name = "IDCATEGORY")  
 @SerializedName(value = "idCategory", alternate = "idcategory")  
 private String idcategory;  
  
 @Column(name = "STRCATEGORY")  
 @SerializedName(value = "strCategory", alternate = "strcategory")  
 private String strcategory;  
  
 @Column(name = "STRCATEGORYTHUMB")  
 @SerializedName(value = "strCategoryThumb", alternate = "strcategorythumb")  
 private String strcategorythumb;  
  
 @Column(name = "STRCATEGORYDESCRIPTION", length = 6000)  
 @SerializedName(value = "strCategoryDescription", alternate = "strcategorydescription")  
 private String strcategorydescription;  
  
 public MealCategory() {  
 }  
   
 public MealCategory(String idcategory, String strcategory, String strcategorythumb, String strcategorydescription) {  
 }  
  
 public MealCategory(String idcategory) {  
 this.idcategory = idcategory;  
 }  
  
 public String getIdcategory() {  
 return idcategory;  
 }  
  
 public void setIdcategory(String idcategory) {  
 this.idcategory = idcategory;  
 }  
  
 public String getStrcategory() {  
 return strcategory;  
 }  
  
 public void setStrcategory(String strcategory) {  
 this.strcategory = strcategory;  
 }  
  
 public String getStrcategorythumb() {  
 return strcategorythumb;  
 }  
  
 public void setStrcategorythumb(String strcategorythumb) {  
 this.strcategorythumb = strcategorythumb;  
 }  
  
 public String getStrcategorydescription() {  
 return strcategorydescription;  
 }  
  
 public void setStrcategorydescription(String strcategorydescription) {  
 this.strcategorydescription = strcategorydescription;  
 }  
  
 @Override  
 public int hashCode() {  
 int hash = 0;  
 hash += (idcategory != null ? idcategory.hashCode() : 0);  
 return hash;  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 // TODO: Warning - this method won't work in the case the id fields are not set  
 if (!(object instanceof MealCategory)) {  
 return false;  
 }  
 MealCategory other = (MealCategory) object;  
 if ((this.idcategory == null && other.idcategory != null) || (this.idcategory != null && !this.idcategory.equals(other.idcategory))) {  
 return false;  
 }  
 return true;  
 }  
  
 @Override  
 public String toString() {  
 return "mealsapp.MealCategory[ idcategory=" + idcategory + " ]";  
 }  
  
}

package mealsapp.MealClasses. Meal.java

package mealsapp.MealClasses;  
  
import java.io.Serializable;  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.NamedQueries;  
import javax.persistence.NamedQuery;  
import com.google.gson.annotations.SerializedName;  
  
  
@Entity  
  
@NamedQueries({  
 @NamedQuery(name = "Meal.findByTimesofvisit", query = "SELECT m FROM Meal m WHERE m.timesofvisit = :timesofvisit"),  
 @NamedQuery(name = "Meal.sortByTimesofVisit", query = "SELECT m FROM Meal m ORDER BY m.timesofvisit DESC, m.strmeal"),  
 @NamedQuery(name = "Meal.getTimesOfVisit", query = "SELECT m.timesofvisit FROM Meal m WHERE m.strmeal = :strmeal"),  
 @NamedQuery(name = "Meal.getIdmeal", query = "SELECT m.idmeal FROM Meal m WHERE m.strmeal = :strmeal"),  
 @NamedQuery(name = "Meal.getLikeStrMeal", query = "SELECT m FROM Meal m WHERE m.strmeal LIKE :strmeal ORDER BY m.strmeal ASC"),  
 @NamedQuery(name = "Meal.findByAbsoluteStrmeal", query = "SELECT m FROM Meal m WHERE ABS(m.strmeal) = :strmeal"),  
 @NamedQuery(name = "Meal.findBygetIdcategory", query = "SELECT m.idcategory FROM Meal m WHERE m.idcategory = :idcategory")  
   
   
  
})  
public class Meal extends MealCategory implements Serializable {  
  
 private static final long serialVersionUID = 1L;  
  
 @Column(name = "IDMEAL")  
 @SerializedName(value = "idMeal", alternate = "idmeal")  
 private String idmeal;  
  
 @Column(name = "STRMEAL")  
 @SerializedName(value = "strMeal", alternate = "strmeal")  
 private String strmeal;  
  
 @Column(name = "STRAREA")  
 @SerializedName(value = "strArea", alternate = "strarea")  
 private String strarea;  
  
 @Column(name = "STRINSTRUCTIONS", length = 6000)  
 @SerializedName(value = "strInstructions", alternate = "strinstructions")  
 private String strinstructions;  
  
 @Column(name = "STRMEALTHUMB")  
 @SerializedName(value = "strMealThumb", alternate = "strmealthumb")  
 private String strmealthumb;  
  
 @Column(name = "STRYOUTUBE")  
 @SerializedName(value = "strYoutube", alternate = "stryoutube")  
 private String stryoutube;  
  
 @Column(name = "STRINGREDIENT1")  
 @SerializedName(value = "strIngredient1", alternate = "stringredient1")  
 private String stringredient1;  
  
 @Column(name = "STRINGREDIENT2")  
 @SerializedName(value = "strIngredient2", alternate = "stringredient2")  
 private String stringredient2;  
  
 @Column(name = "STRINGREDIENT3")  
 @SerializedName(value = "strIngredient3", alternate = "stringredient3")  
 private String stringredient3;  
  
 @Column(name = "STRINGREDIENT4")  
 @SerializedName(value = "strIngredient4", alternate = "stringredient4")  
 private String stringredient4;  
  
 @Column(name = "STRINGREDIENT5")  
 @SerializedName(value = "strIngredient5", alternate = "stringredient5")  
 private String stringredient5;  
  
 @Column(name = "STRINGREDIENT6")  
 @SerializedName(value = "strIngredient6", alternate = "stringredient6")  
 private String stringredient6;  
  
 @Column(name = "STRINGREDIENT7")  
 @SerializedName(value = "strIngredient7", alternate = "stringredient7")  
 private String stringredient7;  
  
 @Column(name = "STRINGREDIENT8")  
 @SerializedName(value = "strIngredient8", alternate = "stringredient8")  
 private String stringredient8;  
  
 @Column(name = "STRINGREDIENT9")  
 @SerializedName(value = "strIngredient9", alternate = "stringredient9")  
 private String stringredient9;  
  
 @Column(name = "STRINGREDIENT10")  
 @SerializedName(value = "strIngredient10", alternate = "stringredient10")  
 private String stringredient10;  
  
 @Column(name = "STRINGREDIENT11")  
 @SerializedName(value = "strIngredient11", alternate = "stringredient11")  
 private String stringredient11;  
  
 @Column(name = "STRINGREDIENT12")  
 @SerializedName(value = "strIngredient12", alternate = "stringredient12")  
 private String stringredient12;  
  
 @Column(name = "STRINGREDIENT13")  
 @SerializedName(value = "strIngredient13", alternate = "stringredient13")  
 private String stringredient13;  
  
 @Column(name = "STRINGREDIENT14")  
 @SerializedName(value = "strIngredient14", alternate = "stringredient14")  
 private String stringredient14;  
  
 @Column(name = "STRINGREDIENT15")  
 @SerializedName(value = "strIngredient15", alternate = "stringredient15")  
 private String stringredient15;  
  
 @Column(name = "STRINGREDIENT16")  
 @SerializedName(value = "strIngredient16", alternate = "stringredient16")  
 private String stringredient16;  
  
 @Column(name = "STRINGREDIENT17")  
 @SerializedName(value = "strIngredient17", alternate = "stringredient17")  
 private String stringredient17;  
  
 @Column(name = "STRINGREDIENT18")  
 @SerializedName(value = "strIngredient18", alternate = "stringredient18")  
 private String stringredient18;  
  
 @Column(name = "STRINGREDIENT19")  
 @SerializedName(value = "strIngredient19", alternate = "stringredient19")  
 private String stringredient19;  
  
 @Column(name = "STRINGREDIENT20")  
 @SerializedName(value = "strIngredient20", alternate = "stringredient20")  
 private String stringredient20;  
  
 @Column(name = "STRMEASURE1")  
 @SerializedName(value = "strMeasure1", alternate = "strmeasure1")  
 private String strmeasure1;  
 @Column(name = "STRMEASURE2")  
 @SerializedName(value = "strMeasure2", alternate = "strmeasure2")  
 private String strmeasure2;  
 @Column(name = "STRMEASURE3")  
 @SerializedName(value = "strMeasure3", alternate = "strmeasure3")  
 private String strmeasure3;  
 @Column(name = "STRMEASURE4")  
 @SerializedName(value = "strMeasure4", alternate = "strmeasure4")  
 private String strmeasure4;  
 @Column(name = "STRMEASURE5")  
 @SerializedName(value = "strMeasure5", alternate = "strmeasure5")  
 private String strmeasure5;  
 @Column(name = "STRMEASURE6")  
 @SerializedName(value = "strMeasure6", alternate = "strmeasure6")  
 private String strmeasure6;  
 @Column(name = "STRMEASURE7")  
 @SerializedName(value = "strMeasure7", alternate = "strmeasure7")  
 private String strmeasure7;  
 @Column(name = "STRMEASURE8")  
 @SerializedName(value = "strMeasure8", alternate = "strmeasure8")  
 private String strmeasure8;  
 @Column(name = "STRMEASURE9")  
 @SerializedName(value = "strMeasure9", alternate = "strmeasure9")  
 private String strmeasure9;  
 @Column(name = "STRMEASURE10")  
 @SerializedName(value = "strMeasure10", alternate = "strmeasure10")  
 private String strmeasure10;  
 @Column(name = "STRMEASURE11")  
 @SerializedName(value = "strMeasure11", alternate = "strmeasure11")  
 private String strmeasure11;  
 @Column(name = "STRMEASURE12")  
 @SerializedName(value = "strMeasure12", alternate = "strmeasure12")  
 private String strmeasure12;  
 @Column(name = "STRMEASURE13")  
 @SerializedName(value = "strMeasure13", alternate = "strmeasure13")  
 private String strmeasure13;  
 @Column(name = "STRMEASURE14")  
 @SerializedName(value = "strMeasure14", alternate = "strmeasure14")  
 private String strmeasure14;  
 @Column(name = "STRMEASURE15")  
 @SerializedName(value = "strMeasure15", alternate = "strmeasure15")  
 private String strmeasure15;  
 @Column(name = "STRMEASURE16")  
 @SerializedName(value = "strMeasure16", alternate = "strmeasure16")  
 private String strmeasure16;  
 @Column(name = "STRMEASURE17")  
 @SerializedName(value = "strMeasure17", alternate = "strmeasure17")  
 private String strmeasure17;  
 @Column(name = "STRMEASURE18")  
 @SerializedName(value = "strMeasure18", alternate = "strmeasure18")  
 private String strmeasure18;  
 @Column(name = "STRMEASURE19")  
 @SerializedName(value = "strMeasure19", alternate = "strmeasure19")  
 private String strmeasure19;  
 @Column(name = "STRMEASURE20")  
 @SerializedName(value = "strMeasure20", alternate = "strmeasure20")  
 private String strmeasure20;  
  
 @Column(name = "TIMESOFVISIT")  
   
 private Integer timesofvisit;  
  
 public Meal() {  
 this.timesofvisit = 0;  
 }  
  
 public Meal(String strmeal) {  
 this.strmeal = strmeal;  
 this.timesofvisit = 0;  
 }  
  
 public String getIdmeal() {  
 return idmeal;  
 }  
  
 public void setIdmeal(String idmeal) {  
 this.idmeal = idmeal;  
 }  
  
 public String getStrmeal() {  
 return strmeal;  
 }  
  
 public void setStrmeal(String strmeal) {  
 this.strmeal = strmeal;  
 }  
  
 public String getStrarea() {  
 return strarea;  
 }  
  
 public void setStrarea(String strarea) {  
 this.strarea = strarea;  
 }  
  
 public String getStrinstructions() {  
 return strinstructions;  
 }  
  
 public void setStrinstructions(String strinstructions) {  
 this.strinstructions = strinstructions;  
 }  
  
 public String getStrmealthumb() {  
 return strmealthumb;  
 }  
  
 public void setStrmealthumb(String strmealthumb) {  
 this.strmealthumb = strmealthumb;  
 }  
  
 public String getStryoutube() {  
 return stryoutube;  
 }  
  
 public void setStryoutube(String stryoutube) {  
 this.stryoutube = stryoutube;  
 }  
  
 public String getStringredient1() {  
 return stringredient1;  
 }  
  
 public void setStringredient1(String stringredient1) {  
 this.stringredient1 = stringredient1;  
 }  
  
 public String getStringredient2() {  
 return stringredient2;  
 }  
  
 public void setStringredient2(String stringredient2) {  
 this.stringredient2 = stringredient2;  
 }  
  
 public String getStringredient3() {  
 return stringredient3;  
 }  
  
 public void setStringredient3(String stringredient3) {  
 this.stringredient3 = stringredient3;  
 }  
  
 public String getStringredient4() {  
 return stringredient4;  
 }  
  
 public void setStringredient4(String stringredient4) {  
 this.stringredient4 = stringredient4;  
 }  
  
 public String getStringredient5() {  
 return stringredient5;  
 }  
  
 public void setStringredient5(String stringredient5) {  
 this.stringredient5 = stringredient5;  
 }  
  
 public String getStringredient6() {  
 return stringredient6;  
 }  
  
 public void setStringredient6(String stringredient6) {  
 this.stringredient6 = stringredient6;  
 }  
  
 public String getStringredient7() {  
 return stringredient7;  
 }  
  
 public void setStringredient7(String stringredient7) {  
 this.stringredient7 = stringredient7;  
 }  
  
 public String getStringredient8() {  
 return stringredient8;  
 }  
  
 public void setStringredient8(String stringredient8) {  
 this.stringredient8 = stringredient8;  
 }  
  
 public String getStringredient9() {  
 return stringredient9;  
 }  
  
 public void setStringredient9(String stringredient9) {  
 this.stringredient9 = stringredient9;  
 }  
  
 public String getStringredient10() {  
 return stringredient10;  
 }  
  
 public void setStringredient10(String stringredient10) {  
 this.stringredient10 = stringredient10;  
 }  
  
 public String getStringredient11() {  
 return stringredient11;  
 }  
  
 public void setStringredient11(String stringredient11) {  
 this.stringredient11 = stringredient11;  
 }  
  
 public String getStringredient12() {  
 return stringredient12;  
 }  
  
 public void setStringredient12(String stringredient12) {  
 this.stringredient12 = stringredient12;  
 }  
  
 public String getStringredient13() {  
 return stringredient13;  
 }  
  
 public void setStringredient13(String stringredient13) {  
 this.stringredient13 = stringredient13;  
 }  
  
 public String getStringredient14() {  
 return stringredient14;  
 }  
  
 public void setStringredient14(String stringredient14) {  
 this.stringredient14 = stringredient14;  
 }  
  
 public String getStringredient15() {  
 return stringredient15;  
 }  
  
 public void setStringredient15(String stringredient15) {  
 this.stringredient15 = stringredient15;  
 }  
  
 public String getStringredient16() {  
 return stringredient16;  
 }  
  
 public void setStringredient16(String stringredient16) {  
 this.stringredient16 = stringredient16;  
 }  
  
 public String getStringredient17() {  
 return stringredient17;  
 }  
  
 public void setStringredient17(String stringredient17) {  
 this.stringredient17 = stringredient17;  
 }  
  
 public String getStringredient18() {  
 return stringredient18;  
 }  
  
 public void setStringredient18(String stringredient18) {  
 this.stringredient18 = stringredient18;  
 }  
  
 public String getStringredient19() {  
 return stringredient19;  
 }  
  
 public void setStringredient19(String stringredient19) {  
 this.stringredient19 = stringredient19;  
 }  
  
 public String getStringredient20() {  
 return stringredient20;  
 }  
  
 public void setStringredient20(String stringredient20) {  
 this.stringredient20 = stringredient20;  
 }  
  
 public String getStrmeasure1() {  
 return strmeasure1;  
 }  
  
 public void setStrmeasure1(String strmeasure1) {  
 this.strmeasure1 = strmeasure1;  
 }  
  
 public String getStrmeasure2() {  
 return strmeasure2;  
 }  
  
 public void setStrmeasure2(String strmeasure2) {  
 this.strmeasure2 = strmeasure2;  
 }  
  
 public String getStrmeasure3() {  
 return strmeasure3;  
 }  
  
 public void setStrmeasure3(String strmeasure3) {  
 this.strmeasure3 = strmeasure3;  
 }  
  
 public String getStrmeasure4() {  
 return strmeasure4;  
 }  
  
 public void setStrmeasure4(String strmeasure4) {  
 this.strmeasure4 = strmeasure4;  
 }  
  
 public String getStrmeasure5() {  
 return strmeasure5;  
 }  
  
 public void setStrmeasure5(String strmeasure5) {  
 this.strmeasure5 = strmeasure5;  
 }  
  
 public String getStrmeasure6() {  
 return strmeasure6;  
 }  
  
 public void setStrmeasure6(String strmeasure6) {  
 this.strmeasure6 = strmeasure6;  
 }  
  
 public String getStrmeasure7() {  
 return strmeasure7;  
 }  
  
 public void setStrmeasure7(String strmeasure7) {  
 this.strmeasure7 = strmeasure7;  
 }  
  
 public String getStrmeasure8() {  
 return strmeasure8;  
 }  
  
 public void setStrmeasure8(String strmeasure8) {  
 this.strmeasure8 = strmeasure8;  
 }  
  
 public String getStrmeasure9() {  
 return strmeasure9;  
 }  
  
 public void setStrmeasure9(String strmeasure9) {  
 this.strmeasure9 = strmeasure9;  
 }  
  
 public String getStrmeasure10() {  
 return strmeasure10;  
 }  
  
 public void setStrmeasure10(String strmeasure10) {  
 this.strmeasure10 = strmeasure10;  
 }  
  
 public String getStrmeasure11() {  
 return strmeasure11;  
 }  
  
 public void setStrmeasure11(String strmeasure11) {  
 this.strmeasure11 = strmeasure11;  
 }  
  
 public String getStrmeasure12() {  
 return strmeasure12;  
 }  
  
 public void setStrmeasure12(String strmeasure12) {  
 this.strmeasure12 = strmeasure12;  
 }  
  
 public String getStrmeasure13() {  
 return strmeasure13;  
 }  
  
 public void setStrmeasure13(String strmeasure13) {  
 this.strmeasure13 = strmeasure13;  
 }  
  
 public String getStrmeasure14() {  
 return strmeasure14;  
 }  
  
 public void setStrmeasure14(String strmeasure14) {  
 this.strmeasure14 = strmeasure14;  
 }  
  
 public String getStrmeasure15() {  
 return strmeasure15;  
 }  
  
 public void setStrmeasure15(String strmeasure15) {  
 this.strmeasure15 = strmeasure15;  
 }  
  
 public String getStrmeasure16() {  
 return strmeasure16;  
 }  
  
 public void setStrmeasure16(String strmeasure16) {  
 this.strmeasure16 = strmeasure16;  
 }  
  
 public String getStrmeasure17() {  
 return strmeasure17;  
 }  
  
 public void setStrmeasure17(String strmeasure17) {  
 this.strmeasure17 = strmeasure17;  
 }  
  
 public String getStrmeasure18() {  
 return strmeasure18;  
 }  
  
 public void setStrmeasure18(String strmeasure18) {  
 this.strmeasure18 = strmeasure18;  
 }  
  
 public String getStrmeasure19() {  
 return strmeasure19;  
 }  
  
 public void setStrmeasure19(String strmeasure19) {  
 this.strmeasure19 = strmeasure19;  
 }  
  
 public String getStrmeasure20() {  
 return strmeasure20;  
 }  
  
 public void setStrmeasure20(String strmeasure20) {  
 this.strmeasure20 = strmeasure20;  
 }  
  
 public Integer getTimesofvisit() {  
 return timesofvisit;  
 }  
  
 public void setTimesofvisit(Integer timesofvisit) {  
 this.timesofvisit = timesofvisit;  
 }  
  
 @Override  
 public int hashCode() {  
 int hash = 0;  
 hash += (strmeal != null ? strmeal.hashCode() : 0);  
 return hash;  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 // TODO: Warning - this method won't work in the case the id fields are not set  
 if (!(object instanceof Meal)) {  
 return false;  
 }  
 Meal other = (Meal) object;  
 if ((this.strmeal == null && other.strmeal != null) || (this.strmeal != null && !this.strmeal.equals(other.strmeal))) {  
 return false;  
 }  
 return true;  
 }  
  
 @Override  
 public String toString() {  
 return "mealsapp.Meal[ strmeal=" + strmeal + " ]";  
 }  
  
}

## Ερώτημα Β – Δημιουργία GUI Εφαρμογής

Ξεκινήσαμε με ένα απλοϊκό mockup ώστε να μπορούμε να οραματιστούμε τις δυνατότητες και λειτουργίες της εφαρμογής, η πλατφόρμα Figma βοήθησε πολύ παραγωγικά σε αυτό το story. Για λόγους πληρότητας παραθέτουμε στα παραδοτέα αρχεία ένα Word με την εξέλιξη του GUI.

Κώδικας διεπαφής περιλαμβάνεται μόνο ο κώδικα που αναπτύχθηκε από την ομάδα.

package mealsapp.GUI. MainForm.java

package mealsapp.GUI;  
  
import com.google.gson.JsonObject;  
import com.itextpdf.text.DocumentException;  
import com.itextpdf.text.pdf.BaseFont;  
import java.awt.Color;  
import java.awt.Cursor;  
import java.awt.Desktop;  
import java.awt.Image;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.awt.event.MouseListener;  
import java.io.FileNotFoundException;  
import java.io.IOException;  
import java.net.MalformedURLException;  
import java.net.URI;  
import java.net.URISyntaxException;  
import java.net.URL;  
import java.util.ArrayList;  
import java.util.HashMap;  
import javax.swing.\*;  
import java.util.List;  
import java.util.Map;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.swing.table.DefaultTableModel;  
import mealsapp.API.JsonParsers;  
import mealsapp.MealClasses.Meal;  
import static mealsapp.Charts.ColumnChart.ColumnChart;  
import static mealsapp.Charts.PieChart.PieChart;  
import mealsapp.DB.mealsQueryOperations;  
import static mealsapp.GUI.commandList.meal;  
import mealsapp.MealClasses.MealCategory;  
  
public class MainForm extends javax.swing.JFrame {  
  
 */\*\*  
 \* Creates new form MainForm  
 \*/* Color defaultColor, clickedColor;  
  
 MealCategory chossenMealCategory = null;  
 List<MealCategory> mealCategories = null;  
 List<Meal> meals = null;  
 ArrayList<String> areas = null;  
 ArrayList<String[]> ingredients = null;  
 boolean editMode = false;  
  
 public MainForm() {  
 initComponents();  
  
 clickedColor = new Color(61, 61, 61);  
 defaultColor = new Color(158, 134, 49);  
  
 //Τοποθετηση Εικόνας στην αρχική   
 ImageIcon imgIcon = new ImageIcon("resources/Mousaka.png");  
 imgIcon = new ImageIcon(imgIcon.getImage().getScaledInstance(jLabel18.getWidth(), jLabel18.getHeight(), Image.SCALE\_SMOOTH));  
 jLabel18.setIcon(imgIcon);  
  
 //Τοποθέτηση της εφαρμογής στο κέντρο της οθόνης κατα το άνοιγμα  
 setLocationRelativeTo(null);  
 }  
  
 */\*\*  
 \* 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.  
 \*/* private void jLabel1MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jLabel1MousePressed  
  
 FocusPanel.setSelectedIndex(1);  
 menuSearchPnl.setBackground(clickedColor);  
 }//GEN-LAST:event\_jLabel1MousePressed  
  
 private void jLabel1MouseReleased(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jLabel1MouseReleased  
  
 menuSearchPnl.setBackground(defaultColor);  
 }//GEN-LAST:event\_jLabel1MouseReleased  
  
 private void statsBtnMousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_statsBtnMousePressed  
  
 FocusPanel.setSelectedIndex(3);  
 menuStatsPnl.setBackground(clickedColor);  
 statsComboMenu.setSelectedIndex(0);  
 }//GEN-LAST:event\_statsBtnMousePressed  
  
 private void extBtnMousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_extBtnMousePressed  
  
 menuExitPnl.setBackground(clickedColor);  
 System.exit(0);  
 }//GEN-LAST:event\_extBtnMousePressed  
  
 private void sbcComboBoxActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_sbcComboBoxActionPerformed  
  
 String s = (String) sbcComboBox.getSelectedItem();  
 switch (s) {  
 case "Search By Name" -> {  
 GeneralSearchPnl.setSelectedIndex(0);  
 sbcComboBox.setSelectedIndex(1);  
 }  
 case "Search By Category" -> {  
 GeneralSearchPnl.setSelectedIndex(1);  
 sbcComboBox.setSelectedIndex(0);  
 //Inserts Accending Meal Categories to JTable and display them  
  
 mealCategories = JsonParsers.getAllMealCategories();  
 DefaultListModel<String> model = commandList.getCategoriesModel(mealCategories);  
 jList3.setModel(model);  
  
 if ((model.isEmpty())) {  
 MainForm.errorFound("Failed to grab Meals Categories, please check your Internet connection!", "Error");  
 }  
 }  
 case "Search By Area" -> {  
 GeneralSearchPnl.setSelectedIndex(2);  
 sbcComboBox.setSelectedIndex(2);  
 areas = JsonParsers.getAllArea();  
 DefaultListModel<String> model = commandList.getAreaModel(areas);  
 jList5.setModel(model);  
 if ((model.isEmpty())) {  
 MainForm.errorFound("Failed to grab Areas, please check your Internet connection!", "Error");  
 }  
  
 }  
  
 case "Search By Ingredient" -> {  
 GeneralSearchPnl.setSelectedIndex(3);  
 sbcComboBox.setSelectedIndex(3);  
 ingredients = JsonParsers.getAllIngredient();  
 DefaultListModel<String> model = commandList.getIngredientsModel(ingredients);  
 jList2.setModel(model);  
 if ((model.isEmpty())) {  
 MainForm.errorFound("Failed to grab Ingredients, please check your Internet connection!", "Error");  
 }  
  
 }  
 }  
  
 }//GEN-LAST:event\_sbcComboBoxActionPerformed  
  
 private void statsComboMenuActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_statsComboMenuActionPerformed  
  
 String s = (String) statsComboMenu.getSelectedItem();  
 switch (s) {  
 case "3D Columns" -> {  
  
 jTabbedPane2.setSelectedIndex(0);  
  
 TwoDStatsPnl.setSize(840, 608);  
  
 TwoDStatsPnl.removeAll();  
 TwoDStatsPnl.add(ColumnChart());  
 TwoDStatsPnl.setVisible(true);  
  
 }  
 case "3D Pie" -> {  
 jTabbedPane2.setSelectedIndex(1);  
 jTabbedPane2.setSelectedIndex(0);  
  
 TwoDStatsPnl.setSize(800, 600);  
  
 TwoDStatsPnl.removeAll();  
 TwoDStatsPnl.add(PieChart());  
  
 TwoDStatsPnl.setVisible(true);  
 }  
 case "Table" -> {  
  
 jTabbedPane2.setSelectedIndex(2);  
 // Set the table model to the ingredientsTable  
 StatsJTable.setModel(commandList.getTableModel());  
 //Makes table not editable  
 StatsJTable.setDefaultEditor(Object.class, null);  
 }  
  
 }  
 }//GEN-LAST:event\_statsComboMenuActionPerformed  
  
 private void sbnSearchBtnMousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_sbnSearchBtnMousePressed  
 try {  
  
 commandList.searchCommand(sbnTxt.getText());  
 } catch (MalformedURLException ex) {  
 Logger.getLogger(MainForm.class.getName()).log(Level.SEVERE, null, ex);  
 }  
  
  
 }//GEN-LAST:event\_sbnSearchBtnMousePressed  
  
 private void updateBtnMousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_updateBtnMousePressed  
  
 if (ingredientsTable.getDefaultEditor(Object.class) == null) {  
 ingredientsTable.setDefaultEditor(Object.class, new DefaultCellEditor(new JTextField()));  
 }  
  
 // Enable editing of the other components  
 sbnAreaTxt.setEditable(true);  
 sbnCategoryTxt.setEditable(true);  
 sbnMealNameTxt.setEditable(true);  
 sbnInstructions.setEditable(true);  
 ingredientsTable.removeAll();  
 editMode = true;  
 }//GEN-LAST:event\_updateBtnMousePressed  
  
 private void deleteBtnMousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_deleteBtnMousePressed  
  
 boolean validChoise = commandList.userDeletesMeal(sbnMealNameTxt.getText());  
 if (validChoise) {  
  
 //Clears GUI  
 MainForm.sbnAreaTxt.setText("");  
 sbnCategoryTxt.setText("");  
 sbnMealNameTxt.setText("");  
 sbnAreaTxt.setText("");  
 sbnInstructions.setText("");  
  
 //Clears JTable Data  
 int numRows = ingredientsTable.getRowCount();  
 int numCols = ingredientsTable.getColumnCount();  
  
 for (int i = 0; i < numRows; i++) {  
 for (int j = 0; j < numCols; j++) {  
 ingredientsTable.setValueAt(null, i, j);  
 }  
 }  
  
 sbnImageLabel.setIcon(null);  
 sbnImageLabel.setText("");  
 UrlLblSbn.setCursor((Cursor.getDefaultCursor()));  
// UrlLblSbn.removeMouseListener(UrlLblSbn.getMouseListeners()[0]);  
 UrlLblSbn.setText("");  
  
 //Clears search results Lists  
 jList1.setModel(new DefaultListModel<>());  
  
 }  
  
 }//GEN-LAST:event\_deleteBtnMousePressed  
  
 private void statsBtnMouseReleased(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_statsBtnMouseReleased  
  
 menuStatsPnl.setBackground(defaultColor);  
  
 }//GEN-LAST:event\_statsBtnMouseReleased  
  
 private void sbnSaveBtnMousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_sbnSaveBtnMousePressed  
  
 sbnAreaTxt.setEditable(false);  
 sbnCategoryTxt.setEditable(false);  
 sbnMealNameTxt.setEditable(false);  
 sbnInstructions.setEditable(false);  
  
 //setEditable(false) for table  
 ingredientsTable.setDefaultEditor(Object.class, null);  
  
 Map<String, Object> dataFromUser = new HashMap<>();  
  
 dataFromUser.put("sbnCategoryTxt", sbnCategoryTxt);  
 dataFromUser.put("sbnTitleTxt", sbnMealNameTxt);  
 dataFromUser.put("sbnInstructions", sbnInstructions);  
 dataFromUser.put("ingredientsTable", ingredientsTable);  
 dataFromUser.put("sbnAreaTxt", sbnAreaTxt);  
 dataFromUser.put("sbnImageLabel", sbnImageLabel.getText());  
 editMode = false;  
  
 commandList.saveMeal(sbnMealNameTxt.getText(), dataFromUser);  
 }//GEN-LAST:event\_sbnSaveBtnMousePressed  
  
 private void statsExpBtnMousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_statsExpBtnMousePressed  
  
 int userView = statsComboMenu.getSelectedIndex();  
 try {  
 commandList.printPDF(userView);  
 } catch (FileNotFoundException ex) {  
 Logger.getLogger(MainForm.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (DocumentException ex) {  
 Logger.getLogger(MainForm.class.getName()).log(Level.SEVERE, null, ex);  
 } catch (IOException ex) {  
 Logger.getLogger(MainForm.class.getName()).log(Level.SEVERE, null, ex);  
 }  
 }//GEN-LAST:event\_statsExpBtnMousePressed  
  
 private void JListMealMousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_JListMealMousePressed  
  
 try {  
 //String UserSelectedMeal = JListMeal.getSelectedValue();  
  
 if (JListMeal.getSelectedValue() != null) {  
  
 FocusPanel.setSelectedIndex(1);  
  
 //searchByMealID(String userMeal, String idString)  
 for (int i = 0; i < meals.size(); i++) {  
 if (JListMeal.getSelectedValue().equals(meals.get(i).getStrmeal())) {  
 commandList.searchByMealID(meals.get(i).getStrmeal(), meals.get(i).getIdmeal());  
 }  
 }  
 }  
 } catch (MalformedURLException ex) {  
 //Logger.getLogger(MainForm.class.getName()).log(Level.SEVERE, null, ex);  
 }  
  
 }//GEN-LAST:event\_JListMealMousePressed  
  
 private void jList3MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jList3MousePressed  
  
 try {  
  
 for (int i = 0; i < mealCategories.size(); i++) {  
 if (jList3.getSelectedValue().equals(mealCategories.get(i).getStrcategory())) {  
  
 chossenMealCategory = mealCategories.get(i);  
 //put description etc  
 categoryInstructionsPnl.setText(chossenMealCategory.getStrcategorydescription());  
  
 categoryInstructionsPnl.setCaretPosition(0);  
 jList3.setSelectionBackground(defaultColor);  
 //set image  
 try {  
 if (chossenMealCategory.getStrcategorythumb().contains("www")) {  
 ImageIcon image = new ImageIcon(new URL(chossenMealCategory.getStrcategorythumb()));  
 Image scaledImage = image.getImage().getScaledInstance(sbnImageLabel.getWidth(), sbnImageLabel.getHeight(), Image.SCALE\_SMOOTH);  
 ImageIcon scaledIcon = new ImageIcon(scaledImage);  
 jLabelMealCategoryImage.setIcon(scaledIcon);  
 jLabelMealCategoryImage.setText("");  
  
 }  
  
 } catch (Exception e) {  
 jLabelMealCategoryImage.setIcon(null);  
 jLabelMealCategoryImage.setText("Image Not Available");  
 }  
 meals = JsonParsers.getAllMealByCategory(chossenMealCategory.getStrcategory());  
 JListMeal.setModel(commandList.updateModelMealList(meals));  
  
 }  
 }  
 } catch (Exception e) {  
 }  
  
  
 }//GEN-LAST:event\_jList3MousePressed  
  
 private void jLabel11MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jLabel11MousePressed  
  
 FocusPanel.setSelectedIndex(2);  
 }//GEN-LAST:event\_jLabel11MousePressed  
  
 private void RandomMealBtnMousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_RandomMealBtnMousePressed  
  
 try {  
 commandList.randomMeal();  
  
 } catch (MalformedURLException ex) {  
 Logger.getLogger(MainForm.class  
 .getName()).log(Level.SEVERE, null, ex);  
 }  
  
  
 }//GEN-LAST:event\_RandomMealBtnMousePressed  
  
 private void SearchByNameBtnMousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_SearchByNameBtnMousePressed  
  
 String searchValue = searchByNameTxt.getText();  
  
 if (ApiDataRadio.isSelected()) {  
  
 JsonObject getAllMealsJsonString = JsonParsers.getAllMealsJsonString(searchValue);  
  
 meals = JsonParsers.getAllMealBySearch(getAllMealsJsonString);  
  
 jList1.setModel(commandList.updateModelMealList(meals));  
  
 } else if (DBdataRadio.isSelected()) {  
 meals = mealsQueryOperations.getLikeMealsFromDB(searchValue);  
  
 jList1.setModel(commandList.updateModelMealList(meals));  
 }  
  
  
 }//GEN-LAST:event\_SearchByNameBtnMousePressed  
  
 private void jList6MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jList6MousePressed  
  
 try {  
 //String UserSelectedMeal = JListMeal.getSelectedValue();  
  
 if (jList5.getSelectedValue() != null) {  
 for (int i = 0; i < meals.size(); i++) {  
  
 if (jList6.getSelectedValue().equals(meals.get(i).getStrmeal())) {  
 commandList.searchByMealID(meals.get(i).getStrmeal(), meals.get(i).getIdmeal());  
 FocusPanel.setSelectedIndex(1);  
 }  
 }  
 }  
 } catch (MalformedURLException ex) {  
 //Logger.getLogger(MainForm.class.getName()).log(Level.SEVERE, null, ex);  
 }  
 }//GEN-LAST:event\_jList6MousePressed  
  
 private void jList5MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jList5MousePressed  
  
 try {  
  
 for (int i = 0; i < areas.size(); i++) {  
 if (jList5.getSelectedValue().equals(areas.get(i))) {  
  
 jList5.setSelectionBackground(defaultColor);  
  
 meals = JsonParsers.getAllMealByArea(jList5.getSelectedValue());  
  
 jList6.setModel(commandList.updateModelMealList(meals));  
  
 /\*for (int i = 0; i < mealCategories.size(); i++) {  
 if (jList3.getSelectedValue().equals(mealCategories.get(i).getStrcategory())) {  
   
 chossenMealCategory = mealCategories.get(i);  
 //put description etc  
 categoryInstructionsPnl.setText(chossenMealCategory.getStrcategorydescription());  
   
 categoryInstructionsPnl.setCaretPosition(0);  
 jList3.setSelectionBackground(defaultColor);  
 //set image  
 try {  
 if (chossenMealCategory.getStrcategorythumb().contains("www")) {  
 ImageIcon image = new ImageIcon(new URL(chossenMealCategory.getStrcategorythumb()));  
 Image scaledImage = image.getImage().getScaledInstance(sbnImageLabel.getWidth(), sbnImageLabel.getHeight(), Image.SCALE\_SMOOTH);  
 ImageIcon scaledIcon = new ImageIcon(scaledImage);  
 jLabelMealCategoryImage.setIcon(scaledIcon);  
 jLabelMealCategoryImage.setText("");  
   
 }  
   
 } catch (Exception e) {  
 jLabelMealCategoryImage.setIcon(null);  
 jLabelMealCategoryImage.setText("Image Not Available");  
 }  
 meals = JsonParsers.getAllMealByCategory(chossenMealCategory.getStrcategory());  
 JListMeal.setModel(commandList.updateModelMealList(meals));\*/  
 }  
 }  
 } catch (Exception e) {  
 }  
  
  
 }//GEN-LAST:event\_jList5MousePressed  
  
 private void jList2MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jList2MousePressed  
 try {  
  
 for (int i = 0; i < ingredients.size(); i++) {  
 if (jList2.getSelectedValue().equals(ingredients.get(i)[0])) {  
  
 // chossenMealCategory = mealCategories.get(i);  
 //put description etc  
 jTextPane1.setText(ingredients.get(i)[1]);  
  
 categoryInstructionsPnl.setCaretPosition(0);  
 jList2.setSelectionBackground(defaultColor);  
 //set image  
 try {  
 ImageIcon image = new ImageIcon(new URL("https://www.themealdb.com/images/ingredients/" + ingredients.get(i)[0] + ".png"));  
 Image scaledImage = image.getImage().getScaledInstance(jLabelMealIngreImage.getWidth(), jLabelMealIngreImage.getHeight(), Image.SCALE\_SMOOTH);  
 ImageIcon scaledIcon = new ImageIcon(scaledImage);  
 jLabelMealIngreImage.setIcon(scaledIcon);  
 jLabelMealIngreImage.setText("");  
  
 } catch (Exception e) {  
 jLabelMealIngreImage.setIcon(null);  
 jLabelMealIngreImage.setText("Image Not Available");  
 }  
 meals = JsonParsers.getAllMealByIngridient(ingredients.get(i)[0]);  
 jList4.setModel(commandList.updateModelMealList(meals));  
  
 }  
 }  
 } catch (Exception e) {  
 }  
  
 }//GEN-LAST:event\_jList2MousePressed  
  
 private void jList4MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jList4MousePressed  
  
 try {  
 //String UserSelectedMeal = JListMeal.getSelectedValue();  
  
 if (jList2.getSelectedValue() != null) {  
 for (int i = 0; i < meals.size(); i++) {  
  
 if (jList4.getSelectedValue().equals(meals.get(i).getStrmeal())) {  
 commandList.searchByMealID(meals.get(i).getStrmeal(), meals.get(i).getIdmeal());  
 FocusPanel.setSelectedIndex(1);  
 }  
 }  
 }  
 } catch (MalformedURLException ex) {  
 //Logger.getLogger(MainForm.class.getName()).log(Level.SEVERE, null, ex);  
 }  
 }//GEN-LAST:event\_jList4MousePressed  
  
 private void jList1MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jList1MousePressed  
 if (ApiDataRadio.isSelected()) {  
 for (int i = 0; i < meals.size(); i++) {  
 if (meals.get(i).getStrmeal().equals(jList1.getSelectedValue())) {  
  
 try {  
  
 commandList.searchedMealAndImportToGui(meals.get(i));  
 FocusPanel.setSelectedIndex(1);  
  
 } catch (MalformedURLException ex) {  
 Logger.getLogger(MainForm.class  
 .getName()).log(Level.SEVERE, null, ex);  
 }  
 }  
  
 }  
 } else if (DBdataRadio.isSelected()) {  
  
 for (int i = 0; i < meals.size(); i++) {  
 if (meals.get(i).getStrmeal().equals(jList1.getSelectedValue())) {  
  
 try {  
  
 //commandList.searchedMealAndImportToGuiOnlyDB(meals.get(i));  
 MainForm.displayMeal(meals.get(i));  
 mealsQueryOperations.addVisit(meals.get(i).getStrmeal());  
 FocusPanel.setSelectedIndex(1);  
 jList1.setModel(new DefaultListModel<>());  
  
 } catch (MalformedURLException ex) {  
 Logger.getLogger(MainForm.class  
 .getName()).log(Level.SEVERE, null, ex);  
  
 }  
 }  
  
 }  
 }  
  
 }//GEN-LAST:event\_jList1MousePressed  
  
 private void ingredientsTableMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_ingredientsTableMouseClicked  
 if (editMode == false) {  
 FocusPanel.setSelectedIndex(2);  
 String ingredient = ingredientsTable.getValueAt(ingredientsTable.getSelectedRow(), 0).toString();  
 GeneralSearchPnl.setSelectedIndex(3);  
 sbcComboBox.setSelectedIndex(3);  
 ingredients = JsonParsers.getAllIngredient();  
 DefaultListModel<String> model = commandList.getIngredientsModel(ingredients);  
 jList2.setModel(model);  
 if ((model.isEmpty())) {  
 MainForm.errorFound("Failed to grab Ingredients, please check your Internet connection!", "Error");  
 } else {  
 for (int i = 0; i < ingredients.size(); i++) {  
  
 if (ingredients.get(i)[0].equals(ingredient)) {  
 jList2.setSelectedIndex(i);  
 categoryInstructionsPnl.setCaretPosition(0);  
 jList2.setSelectionBackground(defaultColor);  
  
 jList2.ensureIndexIsVisible(jList2.getSelectedIndex());  
  
 //set image  
 try {  
 ImageIcon image = new ImageIcon(new URL("https://www.themealdb.com/images/ingredients/" + ingredients.get(i)[0] + ".png"));  
 Image scaledImage = image.getImage().getScaledInstance(jLabelMealIngreImage.getWidth(), jLabelMealIngreImage.getHeight(), Image.SCALE\_SMOOTH);  
 ImageIcon scaledIcon = new ImageIcon(scaledImage);  
 jLabelMealIngreImage.setIcon(scaledIcon);  
 jLabelMealIngreImage.setText("");  
  
 } catch (Exception e) {  
 jLabelMealIngreImage.setIcon(null);  
 jLabelMealIngreImage.setText("Image Not Available");  
 }  
 meals = JsonParsers.getAllMealByIngridient(ingredients.get(i)[0]);  
 jList4.setModel(commandList.updateModelMealList(meals));  
  
 }  
  
 }  
  
 }  
  
 }  
 }//GEN-LAST:event\_ingredientsTableMouseClicked  
  
 */\*\*  
 \** ***@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   
 \*/  
 //</editor-fold>  
 //</editor-fold>  
 /\* Create and display the form \*//\*  
 java.awt.EventQueue.invokeLater(new Runnable() {  
 public void run() {  
 new MainForm().setVisible(true);  
 }  
 });  
 }\*/  
 // Variables declaration - do not modify//GEN-BEGIN:variables  
 private javax.swing.JRadioButton ApiDataRadio;  
 private javax.swing.JPanel ByAreaPnl;  
 private javax.swing.JPanel ByCategoryPnl;  
 private javax.swing.JPanel ByIngredientPnl;  
 private javax.swing.JPanel ByNamePnl;  
 private javax.swing.JPanel CategoryDisplay;  
 private javax.swing.JRadioButton DBdataRadio;  
 private javax.swing.JTabbedPane FocusPanel;  
 private javax.swing.JTabbedPane GeneralSearchPnl;  
 private javax.swing.JPanel IntroPnl;  
 private javax.swing.JList<String> JListMeal;  
 private javax.swing.JPanel LeftPanel;  
 private javax.swing.JPanel MainPanel;  
 private javax.swing.JPanel MealDisplay;  
 private javax.swing.JPanel MealPanel;  
 private javax.swing.JPanel PieStatsPnl;  
 private javax.swing.JButton RandomMealBtn;  
 private javax.swing.JButton SearchByNameBtn;  
 private javax.swing.JPanel StatsDisplay;  
 private javax.swing.JTable StatsJTable;  
 private javax.swing.JPanel TableStatsPnl;  
 private javax.swing.JPanel TopPanel;  
 private javax.swing.JPanel TwoDStatsPnl;  
 private static javax.swing.JLabel UrlLblSbn;  
 private javax.swing.ButtonGroup buttonGroup1;  
 private javax.swing.JTextPane categoryInstructionsPnl;  
 private javax.swing.JLabel dataFromLbl;  
 private javax.swing.JButton deleteBtn;  
 private javax.swing.JLabel extBtn;  
 private static javax.swing.JTable ingredientsTable;  
 private javax.swing.JLabel jLabel1;  
 private javax.swing.JLabel jLabel10;  
 private javax.swing.JLabel jLabel11;  
 private javax.swing.JLabel jLabel12;  
 private javax.swing.JLabel jLabel13;  
 private javax.swing.JLabel jLabel14;  
 private javax.swing.JLabel jLabel15;  
 private javax.swing.JLabel jLabel16;  
 private javax.swing.JLabel jLabel17;  
 private javax.swing.JLabel jLabel18;  
 private javax.swing.JLabel jLabel19;  
 private javax.swing.JLabel jLabel2;  
 private javax.swing.JLabel jLabel20;  
 private javax.swing.JLabel jLabel3;  
 private javax.swing.JLabel jLabel4;  
 private javax.swing.JLabel jLabel5;  
 private javax.swing.JLabel jLabel6;  
 private javax.swing.JLabel jLabel7;  
 private javax.swing.JLabel jLabel8;  
 private javax.swing.JLabel jLabel9;  
 private javax.swing.JLabel jLabelMealCategoryImage;  
 private javax.swing.JLabel jLabelMealIngreImage;  
 private javax.swing.JList<String> jList1;  
 private javax.swing.JList<String> jList2;  
 private javax.swing.JList<String> jList3;  
 private javax.swing.JList<String> jList4;  
 private javax.swing.JList<String> jList5;  
 private javax.swing.JList<String> jList6;  
 private javax.swing.JPanel jPanel1;  
 private javax.swing.JPanel jPanel11;  
 private javax.swing.JPanel jPanel12;  
 private javax.swing.JPanel jPanel13;  
 private javax.swing.JPanel jPanel14;  
 private javax.swing.JPanel jPanel15;  
 private javax.swing.JPanel jPanel16;  
 private javax.swing.JPanel jPanel17;  
 private javax.swing.JPanel jPanel18;  
 private javax.swing.JPanel jPanel19;  
 private javax.swing.JPanel jPanel20;  
 private javax.swing.JPanel jPanel21;  
 private javax.swing.JPanel jPanel22;  
 private javax.swing.JPanel jPanel23;  
 private javax.swing.JPanel jPanel24;  
 private javax.swing.JPanel jPanel3;  
 private javax.swing.JPanel jPanel4;  
 private javax.swing.JPanel jPanel5;  
 private javax.swing.JScrollPane jScrollPane1;  
 private javax.swing.JScrollPane jScrollPane10;  
 private javax.swing.JScrollPane jScrollPane11;  
 private javax.swing.JScrollPane jScrollPane12;  
 private javax.swing.JScrollPane jScrollPane2;  
 private javax.swing.JScrollPane jScrollPane3;  
 private javax.swing.JScrollPane jScrollPane4;  
 private javax.swing.JScrollPane jScrollPane5;  
 private javax.swing.JScrollPane jScrollPane6;  
 private javax.swing.JScrollPane jScrollPane7;  
 private javax.swing.JScrollPane jScrollPane8;  
 private javax.swing.JScrollPane jScrollPane9;  
 private javax.swing.JTabbedPane jTabbedPane2;  
 private javax.swing.JTextPane jTextPane1;  
 private javax.swing.JPanel menuExitPnl;  
 private javax.swing.JPanel menuSearchPnl;  
 private javax.swing.JPanel menuStatsPnl;  
 public javax.swing.JComboBox<String> sbcComboBox;  
 private static javax.swing.JTextField sbnAreaTxt;  
 private static javax.swing.JTextField sbnCategoryTxt;  
 private static javax.swing.JLabel sbnImageLabel;  
 private static javax.swing.JTextArea sbnInstructions;  
 private static javax.swing.JTextField sbnMealNameTxt;  
 private javax.swing.JButton sbnSaveBtn;  
 private javax.swing.JButton sbnSearchBtn;  
 private static javax.swing.JTextField sbnTxt;  
 private javax.swing.JTextField searchByNameTxt;  
 private javax.swing.JLabel statsBtn;  
 private javax.swing.JComboBox<String> statsComboMenu;  
 private javax.swing.JButton statsExpBtn;  
 private javax.swing.JButton updateBtn;  
 // End of variables declaration//GEN-END:variables  
  
 //Μέθοδος για τη λήψη των γευμάτων με βάση την κατηγορία τους  
 private void fetchMealByCategory(String category) {  
 // Δημιουργία ενός αντικειμένου τύπου commandList για την αποστολή του αιτήματος στο API  
 commandList mealList = new commandList();  
  
 // Λήψη των γευμάτων για τη συγκεκριμένη κατηγορία  
 List<String> meals = mealList.fetchMealsByCategory(category);  
  
 // Συνένωση των ονομάτων των γευμάτων σε ένα μόνο κείμενο με διαχωριστικά ανά γραμμή  
 StringBuilder mealText = new StringBuilder();  
 for (String meal : meals) {  
 mealText.append(meal).append("\n");  
 }  
  
 }  
  
 //Μέθοδος για δημιουργία ερωτημάτων στο χρήστη για την προέλευση των δεδομένων του γεύματος  
 public static int wantDataQuestion(String mealString) {  
 // Δημιουργία των επιλογών για τον χρήστη  
 Object[] options = {"DB", "API", "Cancel"};  
  
 // Εμφάνιση παραθύρου επιλογών με την ερώτηση για την προέλευση των δεδομένων  
 int choise = JOptionPane.showOptionDialog(null,  
 mealString + " found in DB, would you like to show data from DB or API?",  
 "Warning!",  
 JOptionPane.DEFAULT\_OPTION,  
 JOptionPane.WARNING\_MESSAGE,  
 null,  
 options,  
 options[2]);  
 // Επιστροφή της επιλογής του χρήστη  
 return choise;  
 }  
  
 //Μέθοδος για τη δημιουργία ερωτημάτων στο χρήστη για τη διαγραφή δεδομένων του γεύματος  
 public static int wantDeleteQuestion(String mealString) {  
 // Δημιουργία των επιλογών για τον χρήστη  
 Object[] options = {"Delete", "Keep Statistics", "Cancel"};  
  
 // Εμφάνιση παραθύρου επιλογών με την ερώτηση για τη διαγραφή δεδομένων  
 int choise = JOptionPane.showOptionDialog(null,  
 mealString + " data will be lost, choose what data you want to delete!",  
 "Warning!",  
 JOptionPane.DEFAULT\_OPTION,  
 JOptionPane.WARNING\_MESSAGE,  
 null,  
 options,  
 options[2]);  
  
 // Επιστροφή της επιλογής του χρήστη  
 return choise;  
 }  
  
 //Μέθοδος για τη δημιουργία ερωτημάτων στο χρήστη με δύο επιλογές (Ναι/Όχι)  
 public static int wantQuestionYesNo(String message) {  
 // Δημιουργία των επιλογών για τον χρήστη  
 Object[] options = {"Yes", "No"};  
  
 // Εμφάνιση παραθύρου επιλογών με την ερώτηση για τη διαγραφή δεδομένων  
 int choise = JOptionPane.showOptionDialog(null,  
 message,  
 "Warning!",  
 JOptionPane.DEFAULT\_OPTION,  
 JOptionPane.WARNING\_MESSAGE,  
 null,  
 options,  
 options[0]);  
  
 // Επιστροφή της επιλογής του χρήστη  
 return choise;  
 }  
  
 //Μέθοδος για τη δημιουργία παραθύρου ενημέρωσης με ένα κουμπί (ΟΚ)  
 public static int infomessage(String message) {  
 // Δημιουργία των επιλογών για τον χρήστη  
 Object[] options = {"OK"};  
  
 // Εμφάνιση παραθύρου ενημέρωσης με το μήνυμα που δόθηκε σαν παράμετρο  
 int choise = JOptionPane.showOptionDialog(null,  
 message,  
 "Information!",  
 JOptionPane.DEFAULT\_OPTION,  
 JOptionPane.INFORMATION\_MESSAGE,  
 null,  
 options,  
 options[0]);  
  
 // Επιστροφή της επιλογής του χρήστη  
 return choise;  
 }  
  
 //Μέθοδος για τη δημιουργία παραθύρου ενημέρωσης λάθους  
 public static int errorFound(String message, String message2) {  
 // Εμφάνιση παραθύρου με το μήνυμα λάθους που δόθηκε σαν παράμετρο  
 JOptionPane.showMessageDialog(null,  
 message,  
 message2,  
 JOptionPane.OK\_OPTION);  
  
 return 0;  
 }  
  
 //Μέθοδος για τη δημιουργία ερωτημάτων στο χρήστη με δύο επιλογές (Ναι/Όχι)  
 public static int wantDataQuestionYesNo(String message) {  
  
 // Δημιουργία των επιλογών για τον χρήστη  
 Object[] options = {"Yes", "No"};  
  
 // Εμφάνιση παραθύρου ερώτησης με το μήνυμα που δόθηκε σαν παράμετρο και τις επιλογές που δημιουργήθηκαν παραπάνω  
 int choise = JOptionPane.showOptionDialog(null,  
 message,  
 "Warning!",  
 JOptionPane.DEFAULT\_OPTION,  
 JOptionPane.WARNING\_MESSAGE,  
 null,  
 options,  
 options[0]);  
  
 // Επιστροφή της επιλογής του χρήστη  
 return choise;  
 }  
  
 //Μέθοδος για την εμφάνιση μιας συνταγής στο παράθυρο της εφαρμογής  
 public static void displayMeal(Meal meal) throws MalformedURLException {  
 // Εμφάνιση του ονόματος της συνταγής  
 sbnMealNameTxt.setText(meal.getStrmeal());  
  
 // Εμφάνιση της κατηγορίας της συνταγής  
 sbnCategoryTxt.setText(meal.getStrcategory());  
  
 // Εμφάνιση της περιοχής προέλευσης της συνταγής  
 sbnAreaTxt.setText(meal.getStrarea());  
  
 // Εμφάνιση των οδηγιών για την παρασκευή της συνταγής  
 sbnInstructions.setText(meal.getStrinstructions());  
  
 // Θέση του κέρσορα στην αρχή των οδηγιών  
 sbnInstructions.setCaretPosition(0);  
  
 // Λήψη των υλικών της συνταγής και εμφάνισή τους σε πίνακα  
 Object[][] table = HelpCommands.getTable(meal);  
 String[] columnNames = {"Ingredient", "Measure"};  
  
 DefaultTableModel model = new DefaultTableModel(table, columnNames);  
  
 ingredientsTable.setModel(new javax.swing.table.DefaultTableModel(table, columnNames));  
  
 ingredientsTable  
 .setDefaultEditor(Object.class,  
 null);  
 // Εμφάνιση της εικόνας της συνταγής, αν υπάρχει διαθέσιμη εικόνα στο URL  
 try {  
 if (meal.getStrmealthumb().contains("www")) {  
 ImageIcon image = new ImageIcon(new URL(meal.getStrmealthumb()));  
 Image scaledImage = image.getImage().getScaledInstance(sbnImageLabel.getWidth(), sbnImageLabel.getHeight(), Image.SCALE\_SMOOTH);  
 ImageIcon scaledIcon = new ImageIcon(scaledImage);  
 sbnImageLabel.setIcon(scaledIcon);  
 sbnImageLabel.setText(meal.getStrmealthumb());  
  
 }  
  
 } catch (Exception e) {  
 sbnImageLabel.setIcon(null);  
 sbnImageLabel.setText("Image Not Available");  
 }  
  
 // Εμφάνιση συνδέσμου για το βίντεο με οδηγίες, αν υπάρχει διαθέσιμος σύνδεσμος  
 try {  
 if (meal.getStryoutube().matches("^https?://.\*$")) {  
 UrlLblSbn.setText("<html><a href=\"" + meal.getStryoutube() + "\"><font color=white>Click here to watch instructions video.</a></html>");  
 UrlLblSbn.setForeground(new Color(0xffffff));  
 UrlLblSbn.setCursor(new Cursor(Cursor.HAND\_CURSOR));  
 MouseListener[] mouseListeners = UrlLblSbn.getMouseListeners();  
 if (mouseListeners.length > 0) {  
 UrlLblSbn.removeMouseListener(mouseListeners[0]);  
 }  
 UrlLblSbn.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mousePressed(MouseEvent e) {  
 if (Desktop.isDesktopSupported()) {  
 Desktop desktop = Desktop.getDesktop();  
 try {  
 desktop.browse(new URI(meal.getStryoutube()));  
 } catch (IOException | URISyntaxException ex) {  
 ex.printStackTrace();  
 }  
 }  
 }  
 });  
 } else {  
 MouseListener[] mouseListeners = UrlLblSbn.getMouseListeners();  
 if (mouseListeners.length > 0) {  
 UrlLblSbn.removeMouseListener(mouseListeners[0]);  
 }  
 UrlLblSbn.setCursor(Cursor.getDefaultCursor());  
 UrlLblSbn.setText("No video link available");  
 }  
 } catch (Exception e) {  
 MouseListener[] mouseListeners = UrlLblSbn.getMouseListeners();  
 if (mouseListeners.length > 0) {  
 UrlLblSbn.removeMouseListener(mouseListeners[0]);  
 }  
 UrlLblSbn.setCursor(Cursor.getDefaultCursor());  
 UrlLblSbn.setText("No video link available");  
 }  
  
 }  
  
}

Συμπληρωματικές classes για το GUI:

package mealsapp.GUI. commandList.java

package mealsapp.GUI;  
  
import com.google.gson.JsonObject;  
import com.itextpdf.text.DocumentException;  
import java.io.FileNotFoundException;  
import java.io.IOException;  
import java.net.MalformedURLException;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Map;  
import javax.swing.DefaultListModel;  
import javax.swing.table.DefaultTableModel;  
import mealsapp.API.JsonParsers;  
import mealsapp.DB.mealsCategoryQueryOperations;  
import mealsapp.DB.mealsQueryOperations;  
import mealsapp.MealClasses.Meal;  
import mealsapp.MealClasses.MealCategory;  
import mealsapp.Charts.PrintList;  
  
  
public class commandList {  
 // Αντικείμενο τύπου Meal για χρήση στις μεθόδους της κλάσης  
 public static Meal meal = new Meal();  
   
 // Μέθοδος που επιστρέφει μια λίστα με τα ονόματα των συνταγών που ανήκουν σε μια κατηγορία  
 List<String> fetchMealsByCategory(String category) {  
 throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody  
 }  
   
 // Μέθοδος που δημιουργεί ένα αρχείο PDF για εκτύπωση των δεδομένων στον πίνακα  
 public static void printPDF(int userView) throws FileNotFoundException, DocumentException, IOException {  
 //Δημιουργία νέου αντικειμένου PrintList  
 PrintList view = new PrintList();  
 // Κλήση της μεθόδου exportToPDF για δημιουργία του αρχείου PDF  
 view.exportToPDF(userView);  
 }  
  
 // Μέθοδος που κάνει αναζήτηση γεύματος στη βάση δεδομένων ή στο API.  
 public static void searchCommand(String userMeal) throws MalformedURLException {  
  
 //Παίρνουμε την τιμή από το κείμενο αναζήτησης  
 JsonObject mealJsonObject = JsonParsers.getMealJsonString(userMeal);  
 //αν η τιμή δεν είναι null  
 if (mealJsonObject != null) {  
 //αναζήτηση γεύματος στο https://www.themealdb.com/  
 String mealName = mealJsonObject.get("strMeal").getAsString();  
 //Παίρνουμε τα JSON και ελέγχουμε αν τα δεδομένα του JSON δεν είναι null  
 if (mealsQueryOperations.findMealByStrMeal(mealName) == true) {  
 //Ελέγχουμε αν ο χρήστης έχει αποθηκεύσει / επεξεργαστεί το γεύμα  
 if (mealsQueryOperations.findIfMealIsSavedByUser(mealName) == true) {  
 //Ρωτάμε τον χρήστη αν θέλει τα δεδομένα από τη βάση δεδομένων ή το API.  
 int userChoice = MainForm.wantDataQuestion(mealName);  
 //Παίρνουμε τα δεδομένα από τη βάση δεδομένων και προσθέτουμε +1  
 if (userChoice == 0) {  
  
 meal = mealsQueryOperations.getMealByStrMeal(mealName);  
  
 MainForm.displayMeal(meal);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
 //Εμφανίζουμε τα δεδομένα από το API και προσθέτουμε +1  
 } else if (userChoice == 1) {  
  
 meal = JsonParsers.createMealFromJson(mealJsonObject);  
  
 MainForm.displayMeal(meal);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
   
 } else {  
  
 }  
 //Αν το γεύμα δεν είναι αποθηκευμένο, παίρνουμε τα δεδομένα από το API και προσθέτουμε +1  
 } else {  
 //Δημιουργούμε ένα αντικείμενο Meal βάσει του JSON  
 Meal helpMeal = JsonParsers.logMealObjByName(mealJsonObject);  
 //Προσθέτουμε το γεύμα στη βάση δεδομένων  
 mealsQueryOperations.insertNewMeal(helpMeal);  
 //Παίρνουμε τα δεδομένα από το JSON και προσθέτουμε +1  
 meal = JsonParsers.createMealFromJson(mealJsonObject);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
 //Εμφανίζουμε τα δεδομένα του γεύματος  
 MainForm.displayMeal(meal);  
  
 }  
   
 } else {  
 //Δημιουργούμε ένα αντικείμενο Meal βάσει του JSON  
 Meal helpMeal = JsonParsers.logMealObjByName(mealJsonObject);  
 //Προσθέτουμε το γεύμα στη βάση δεδομένων  
 mealsQueryOperations.insertNewMeal(helpMeal);  
 //Παίρνουμε τα δεδομένα από το JSON και προσθέτουμε +1  
 meal = JsonParsers.createMealFromJson(mealJsonObject);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
 //Εμφανίζουμε τα δεδομένα του γεύματος  
 MainForm.displayMeal(meal);  
 }  
 //Αν το γεύμα έχει αποθηκευτεί στη βάση δεδομένων, παίρνουμε τα δεδομένα από τη βάση και προσθέτουμε +1  
 } else if (mealsQueryOperations.findMealByStrMeal(userMeal) == true) {  
   
 meal = mealsQueryOperations.getMealByStrMeal(userMeal);  
  
 MainForm.displayMeal(meal);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
  
   
 //Αν το γεύμα δεν υπάρχει στη βάση δεδομένων ή στο https://www.themealdb.com/, εμφανίζουμε μήνυμα σφάλματος  
 } else {  
 MainForm.errorFound(userMeal + " not found in themealdb.com or local db!", "Error Message");  
 }  
  
 }  
   
 // Μέθοδος που κάνει αναζήτηση ένα γεύμα βάσει του αναγνωριστικού του id  
 public static void searchByMealID(String userMeal, String idString) throws MalformedURLException {  
  
   
 //Παίρνουμε το γεύμα βάσει του ID  
 Meal meal = JsonParsers.createMealObjByID(idString);  
  
 try {  
  
 String mealName = meal.getStrmeal();  
 //Ελέγχουμε αν ο χρήστης έχει αποθηκεύσει / επεξεργαστεί το γεύμα  
 if (mealsQueryOperations.findMealByStrMeal(mealName) == true) {  
 //Ρωτάμε τον χρήστη αν θέλει τα δεδομένα από τη βάση δεδομένων ή το API.  
 if (mealsQueryOperations.findIfMealIsSavedByUser(mealName) == true) {  
  
 int userChoice = MainForm.wantDataQuestion(mealName);  
  
 if (userChoice == 0) {  
 //Εμφανίζουμε τα δεδομένα από τη βάση δεδομένων και προσθέτουμε +1  
 meal = mealsQueryOperations.getMealByStrMeal(mealName);  
   
 MainForm.displayMeal(meal);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
  
 } else if (userChoice == 1) {  
 //Εμφανίζουμε τα δεδομένα από το API και προσθέτουμε +1  
 MainForm.displayMeal(meal);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
   
 } else {  
  
 }  
  
 } else {  
  
 //Προσθέτουμε το γεύμα στη βάση δεδομένων και εμφανίζουμε τα δεδομένα του γεύματος, προσθέτοντας +1  
 mealsQueryOperations.insertNewMeal(meal);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
 MainForm.displayMeal(meal);  
  
 }  
 //Αν το γεύμα δεν είναι διαθέσιμο στο API, ελέγχουμε αν υπάρχει στη βάση δεδομένων  
 } else {  
 //Προσθέτουμε το γεύμα στη βάση δεδομένων και εμφανίζουμε τα δεδομένα του γεύματος, προσθέτοντας +1  
 Meal helpMeal = new Meal(meal.getStrmeal());  
 mealsQueryOperations.insertNewMeal(helpMeal);  
   
 mealsQueryOperations.addVisit(meal.getStrmeal());  
 MainForm.displayMeal(meal);  
 }  
  
   
 } catch (Exception e) {  
 MainForm.errorFound("Check your connection!", "Error");  
 }  
  
 }  
   
 /\*   
 Μέθοδος που παίρνει ένα αντικείμενο γεύματος (τύπου Meal) και ελέγχει αν αυτό το γεύμα υπάρχει ήδη στη βάση δεδομένων.   
 Αν υπάρχει, ελέγχει επίσης αν ο χρήστης έχει αποθηκεύσει ή επεξεργαστεί το γεύμα και τον ρωτάει αν θέλει τα δεδομένα   
 του γεύματος από τη βάση δεδομένων ή από την API. Αν το γεύμα δεν υπάρχει στη βάση δεδομένων, τότε προσθέτει το γεύμα   
 στη βάση δεδομένων και το εμφανίζει.  
 \*/  
 public static void searchedMealAndImportToGui(Meal meal) throws MalformedURLException {  
  
 // Λαμβάνουμε το γεύμα βάσει του ονόματος που προέρχεται από την αναζήτηση στο API.  
 try {  
  
 String mealName = meal.getStrmeal();  
 // Ελέγχουμε αν ο χρήστης έχει αποθηκεύσει / επεξεργαστεί το γεύμα  
 if (mealsQueryOperations.findMealByStrMeal(mealName) == true) {  
   
 // Ρωτάμε τον χρήστη αν θέλει τα δεδομένα από τη βάση δεδομένων ή το API.  
 if (mealsQueryOperations.findIfMealIsSavedByUser(mealName) == true) {  
  
 int userChoice = MainForm.wantDataQuestion(mealName);  
  
 if (userChoice == 0) {  
 // Εμφανίζουμε τα δεδομένα από τη βάση δεδομένων και προσθέτουμε +1.  
 meal = mealsQueryOperations.getMealByStrMeal(mealName);  
 // Εμφανίζουμε τα δεδομένα από το API και προσθέτουμε +1.  
 MainForm.displayMeal(meal);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
  
 } else if (userChoice == 1) {  
   
 // Εμφανίζουμε τα δεδομένα από το API και προσθέτουμε +1.  
 MainForm.displayMeal(meal);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
   
 } else {  
  
 }  
  
 } else {  
  
 // Εμφανίζουμε τα δεδομένα από το API και προσθέτουμε +1.  
 mealsQueryOperations.insertNewMeal(meal);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
 MainForm.displayMeal(meal);  
  
 }  
   
 } else {  
   
 // Εμφανίζουμε τα δεδομένα από το API και προσθέτουμε +1.  
 Meal helpMeal = new Meal(meal.getStrmeal());  
 mealsQueryOperations.insertNewMeal(helpMeal);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
 MainForm.displayMeal(meal);  
 }  
  
 // Εμφανίζουμε μήνυμα αποτυχίας.  
 } catch (Exception e) {  
 MainForm.errorFound("Check your connection!", "Error");  
 }  
  
 }  
   
 /\*  
 Μέθοδος που αναζητά ένα γεύμα στην τοπική βάση δεδομένων. Αν ο χρήστης εισαγάγει ένα όνομα γεύματος, ο κώδικας ελέγχει  
 αν υπάρχει στη βάση δεδομένων και αν είναι αποθηκευμένο από τον χρήστη. Αν ναι, το γεύμα εμφανίζεται στο πρόγραμμα.   
 Αν όχι, εμφανίζεται ένα μήνυμα λάθους.  
 \*/  
 public static void searchCommandDB(String userMeal) throws MalformedURLException {  
  
 // Έλεγχος για κενό string  
 if (!"".equals(userMeal)) {  
 // Έλεγχος αν υπάρχει το γεύμα στη βάση δεδομένων  
 if (mealsQueryOperations.findMealByStrMeal(userMeal) == true) {  
 // Έλεγχος αν το γεύμα έχει αποθηκευτεί από τον χρήστη  
 if (mealsQueryOperations.findIfMealIsSavedByUser(userMeal) == true) {  
  
 // Λήψη του γεύματος και εμφάνιση στην κύρια φόρμα  
 meal = mealsQueryOperations.getMealByStrMeal(userMeal);  
 MainForm.displayMeal(meal);  
   
 // Αύξηση του μετρητή επισκέψεων του γεύματος  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
   
  
   
 } else {  
 // Εμφάνιση σφάλματος αν το γεύμα δεν έχει αποθηκευτεί από τον χρήστη  
 MainForm.errorFound(userMeal + " not found!", "Error Message");  
 }  
 } else {  
 // Εμφάνιση σφάλματος αν το γεύμα δεν υπάρχει στη βάση δεδομένων  
 MainForm.errorFound(userMeal + " not found!", "Error Message");  
 }  
  
 } else {  
 // Εμφάνιση σφάλματος αν το πεδίο αναζήτησης είναι κενό  
 MainForm.errorFound("Search field should contain characters when searching in DB!", "Error Message");  
 }  
 }  
  
 // Μέθοδος που κάνει εύρεση ενός τυχαίου γεύματος  
 public static void randomMeal() throws MalformedURLException {  
  
 // Λήψη τυχαίου γεύματος από το API  
 JsonObject mealJsonObject = JsonParsers.randomMeal();  
   
 // Έλεγχος για τυχόν σφάλματα στη λήψη του γεύματος  
 if (mealJsonObject != null) {  
 // Ανάκτηση του ονόματος του γεύματος από το JSON αντικείμενο  
 String mealName = mealJsonObject.get("strMeal").getAsString();  
   
 // Έλεγχος αν το γεύμα υπάρχει ήδη στη βάση δεδομένων  
 if (mealsQueryOperations.findMealByStrMeal(mealName) == true) {  
  
 // Έλεγχος αν το γεύμα έχει ήδη αποθηκευτεί από τον χρήστη  
 if (mealsQueryOperations.findIfMealIsSavedByUser(mealName) == true) {  
 // Ερώτηση για την εμφάνιση των δεδομένων του γεύματος  
 int userChoice = MainForm.wantDataQuestion(mealName);  
  
 if (userChoice == 0) {  
 // Λήψη των δεδομένων του γεύματος από τη βάση δεδομένων και εμφάνιση στην κύρια φόρμα  
 meal = mealsQueryOperations.getMealByStrMeal(mealName);  
 MainForm.displayMeal(meal);  
 // Αύξηση του μετρητή επισκέψεων του γεύματος  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
  
 } else if (userChoice == 1) {  
 // Λήψη των δεδομένων του γεύματος από το JSON αντικείμενο και εμφάνιση στην κύρια φόρμα  
 meal = JsonParsers.createMealFromJson(mealJsonObject);  
  
 MainForm.displayMeal(meal);  
 // Αύξηση του μετρητή επισκέψεων του γεύματος  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
   
 } else {  
  
 }  
  
 } else {  
 // Εισαγωγή του νέου γεύματος στη βάση δεδομένων και εμφάνιση των δεδομένων του στην κύρια φόρμα  
 Meal helpMeal = JsonParsers.logMealObjByName(mealJsonObject);  
 mealsQueryOperations.insertNewMeal(helpMeal);  
 meal = JsonParsers.createMealFromJson(mealJsonObject);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
 MainForm.displayMeal(meal);  
  
 }  
  
 } else {  
 // Εισαγωγή του νέου γεύματος στη βάση δεδομένων και εμφάνιση των δεδομένων του στην κύρια φόρμα  
 Meal helpMeal = JsonParsers.logMealObjByName(mealJsonObject);  
 mealsQueryOperations.insertNewMeal(helpMeal);  
 meal = JsonParsers.createMealFromJson(mealJsonObject);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
 MainForm.displayMeal(meal);  
 }  
  
  
 } else {  
 // Εμφάνιση σφάλματος αν υπάρξει πρόβλημα με τη σύνδεση στο διαδίκτυο  
 MainForm.errorFound("Something went wrong check your Internet Connection!", "Error");  
 }  
  
 }  
  
 // Μέθοδος που κάνει αποθήκευση ενός γεύματος  
 public static void saveMeal(String sbnTitleTxt, Map<String, Object> dataFromUser) {  
 // Έλεγχος για το αν υπάρχει όνομα γεύματος  
 if (!sbnTitleTxt.equals("")) {  
 // Αφαίρεση κενών από την αρχή του ονόματος για σωστή αποθήκευση  
 sbnTitleTxt = sbnTitleTxt.replaceAll("^\\s+", "");  
 // Εμφάνιση ερωτήματος για αποθήκευση του γεύματος  
 int userChoice = MainForm.wantQuestionYesNo("Do you want to save " + sbnTitleTxt + "?");  
  
 if (userChoice == 0) {  
  
 // Έλεγχος για το αν το όνομα δεν είναι κενό  
 if (sbnTitleTxt.isBlank() == false) {  
 // Λήψη δεδομένων για το γεύμα από τον χρήστη και αποθήκευση στο αντικείμενο Meal  
 HelpCommands.grabMealDataFromUser(meal, sbnTitleTxt, dataFromUser);  
  
 // Έλεγχος για το αν το γεύμα υπάρχει ήδη στη βάση δεδομένων  
 if (mealsQueryOperations.findMealByStrMeal(meal.getStrmeal()) == true) {  
 // Αποθήκευση του γεύματος στη βάση δεδομένων  
 mealsQueryOperations.saveMeal(meal);  
   
  
 } else {   
 // Εισαγωγή του νέου γεύματος στη βάση δεδομένων και αύξηση του μετρητή επισκέψεων  
 mealsQueryOperations.insertNewMeal(meal);  
 mealsQueryOperations.addVisit(meal.getStrmeal());  
   
   
 }  
  
 MainForm.infomessage(sbnTitleTxt + " meal saved!");  
 }  
  
 }  
 } else {  
 // Εμφάνιση σφάλματος αν δεν υπάρχει όνομα γεύματος  
 MainForm.errorFound("Meal name should contain characters!", "Error");  
 }  
 }  
   
 // Μέθοδος που διαγράφει ενα γεύμα απο την βάση δεδομένων του χρήστη  
 public static boolean userDeletesMeal(String sbnTitleTxt) {  
 // Έλεγχος για το αν το γεύμα υπάρχει στη βάση δεδομένων του χρήστη  
 if (mealsQueryOperations.findMealByStrMeal(sbnTitleTxt) == true) {  
 // Εμφάνιση επιβεβαίωσης για διαγραφή του γεύματος  
 int userChoice = MainForm.wantDeleteQuestion(sbnTitleTxt);  
 if (userChoice == 1) {  
  
 // Διαγραφή των δεδομένων του γεύματος από τη βάση δεδομένων  
 meal = mealsQueryOperations.getMealByStrMeal(sbnTitleTxt);  
 HelpCommands.deleteMealData(meal);  
 mealsQueryOperations.saveMeal(meal);  
   
 // Εμφάνιση μηνύματος για επιτυχή διαγραφή του γεύματος  
 MainForm.infomessage(sbnTitleTxt + " meal deleted!");  
  
 meal = new Meal();  
 return true;  
  
 }  
 // Ακύρωση διαγραφής των δεδομένων του γεύματος  
 else if (userChoice == 0) {  
   
 mealsQueryOperations.deleteSingleMeal(sbnTitleTxt);  
 meal = mealsQueryOperations.getMealByStrMeal(sbnTitleTxt);  
   
 // Εμφάνιση μηνύματος για επιτυχή διαγραφή των δεδομένων του γεύματος  
 MainForm.infomessage(sbnTitleTxt + " meal data deleted!");  
  
 meal = new Meal();  
 return true;  
   
 }  
  
 } else {  
 // Εμφάνιση σφάλματος αν το γεύμα δε βρέθηκε στη βάση δεδομένων  
 MainForm.errorFound(sbnTitleTxt + " meal not found in local db!", "Error");  
 return false;  
  
 }  
 return false;  
  
 }  
   
 // Μέθοδος για λήψη των κατηγοριών των γευμάτων από το API και εισαγωγή τους στη βάση δεδομένων  
 public static DefaultListModel<String> getCategoriesModel(List<MealCategory> mealCategories) {  
   
 // Λήψη των κατηγοριών των γευμάτων από το API και εισαγωγή τους στη βάση δεδομένων  
 mealsCategoryQueryOperations.importMealCategories(mealCategories);  
 DefaultListModel<String> model = new DefaultListModel<>();  
 for (int i = 0; i < mealCategories.size(); i++) {  
   
 // Προσθήκη των ονομάτων των κατηγοριών στη λίστα  
 model.addElement(mealCategories.get(i).getStrcategory());  
 }  
 return model;  
 }  
   
 // Μέθοδος για λήψη των περιοχών από τη βάση δεδομένων και εισαγωγή τους στη λίστα  
 public static DefaultListModel<String> getAreaModel(ArrayList<String> getAllArea) {  
   
 DefaultListModel<String> model = new DefaultListModel<>();  
 for (int i = 0; i < getAllArea.size(); i++) {  
 // Προσθήκη των ονομάτων των περιοχών στη λίστα  
 model.addElement(getAllArea.get(i));  
 }  
 return model;  
 }  
   
 // Μέθοδος για λήψη των συστατικών από τη βάση δεδομένων και εισαγωγή τους στη λίστα  
 public static DefaultListModel<String> getIngredientsModel(ArrayList<String[]> getAllIngredients) {  
   
 DefaultListModel<String> model = new DefaultListModel<>();  
 for (int i = 0; i < getAllIngredients.size(); i++) {  
 // Προσθήκη των ονομάτων των συστατικών στη λίστα  
 model.addElement(getAllIngredients.get(i)[0]);  
 }  
 return model;  
 }  
   
 // Μέθοδος για δημιουργία του πίνακα με τις πληροφορίες των γευμάτων από τη βάση δεδομένων  
 public static DefaultTableModel getTableModel() {  
  
 List<Meal> meals = mealsQueryOperations.getbyDecOrder();  
 String[][] table = new String[meals.size()][2];  
  
 for (int i = 0; i < meals.size(); i++) {  
 // Προσθήκη του ονόματος του γεύματος και του αριθμού των φορών που έχει επισκεφτεί το γεύμα στον πίνακα  
 table[i][0] = meals.get(i).getStrmeal();  
 table[i][1] = meals.get(i).getTimesofvisit().toString();  
  
 }  
  
 String[] columnNames = {"Meals", "Times Of Visit"};  
 // Δημιουργία μοντέλου πίνακα με τα δεδομένα και τα ονόματα των στηλών  
 DefaultTableModel model = new DefaultTableModel(table, columnNames);  
  
 return model;  
 }  
   
 // Μέθοδος για ανανέωση της λίστας με τα γεύματα  
 public static DefaultListModel<String> updateModelMealList(List<Meal> meals) {  
  
 DefaultListModel<String> model = new DefaultListModel<>();  
 for (int i = 0; i < meals.size(); i++) {  
 // Προσθήκη των ονομάτων των γευμάτων  
 model.addElement(meals.get(i).getStrmeal());  
  
 }  
 return model;  
 }  
}

package mealsapp.GUI. HelpCommands.java

package mealsapp.GUI;  
  
import java.util.Map;  
import javax.swing.JTable;  
import javax.swing.JTextArea;  
import javax.swing.JTextField;  
import mealsapp.MealClasses.Meal;  
  
  
  
public class HelpCommands {  
   
 //Μέθοδος η οποία δέχεται ένα αντικείμενο Meal και επιστρέφει ένα δισδιάστατο πίνακα (Object[][]) με τα συστατικά και τις δοσολογίες του γεύματος.  
 public static Object[][] getTable(Meal meal) {  
   
 Object[][] data = {  
 {meal.getStringredient1(), meal.getStrmeasure1()},  
 {meal.getStringredient2(), meal.getStrmeasure2()},  
 {meal.getStringredient3(), meal.getStrmeasure3()},  
 {meal.getStringredient4(), meal.getStrmeasure4()},  
 {meal.getStringredient5(), meal.getStrmeasure5()},  
 {meal.getStringredient6(), meal.getStrmeasure6()},  
 {meal.getStringredient7(), meal.getStrmeasure7()},  
 {meal.getStringredient8(), meal.getStrmeasure8()},  
 {meal.getStringredient9(), meal.getStrmeasure9()},  
 {meal.getStringredient10(), meal.getStrmeasure10()},  
 {meal.getStringredient11(), meal.getStrmeasure11()},  
 {meal.getStringredient12(), meal.getStrmeasure12()},  
 {meal.getStringredient13(), meal.getStrmeasure13()},  
 {meal.getStringredient14(), meal.getStrmeasure14()},  
 {meal.getStringredient15(), meal.getStrmeasure15()},  
 {meal.getStringredient16(), meal.getStrmeasure16()},  
 {meal.getStringredient17(), meal.getStrmeasure17()},  
 {meal.getStringredient18(), meal.getStrmeasure18()},  
 {meal.getStringredient19(), meal.getStrmeasure19()},  
 {meal.getStringredient20(), meal.getStrmeasure20()}  
 };  
 return data;  
 }  
   
 //Μέθοδος η οποία λαμβάνει στοιχεία ενός γεύματος από τον χρήστη μέσω μιας διεπαφής χρήστη και τα αποθηκεύει στο αντικείμενο Meal.  
 public static Meal grabMealDataFromUser(Meal meal, String sbnTitleTxt, Map<String, Object> dataFromUser) {  
 // Ορίζει τον τίτλο του γεύματος ως το κείμενο που εισήχθη από τον χρήστη  
 meal.setStrmeal(sbnTitleTxt);  
 // Λαμβάνει τις τιμές των πεδίων Category,Instructions,Ingredients,Area και Image   
 JTextField sbnCategoryTxt = (JTextField) dataFromUser.get("sbnCategoryTxt");  
 JTextArea sbnInstructions = (JTextArea) dataFromUser.get("sbnInstructions");  
 JTable ingredientsTable = (JTable) dataFromUser.get("ingredientsTable");  
 JTextField sbnAreaTxt = (JTextField) dataFromUser.get("sbnAreaTxt");  
 String sbnImageLabel = (String) dataFromUser.get("sbnImageLabel");  
  
 meal.setStrcategory(sbnCategoryTxt.getText());  
 meal.setStrarea(sbnAreaTxt.getText());  
 meal.setStrinstructions(sbnInstructions.getText());  
 meal.setStrmealthumb(sbnImageLabel);  
  
 meal.setStringredient1((String) ingredientsTable.getValueAt(0, 0));  
 meal.setStringredient2((String) ingredientsTable.getValueAt(1, 0));  
 meal.setStringredient3((String) ingredientsTable.getValueAt(2, 0));  
 meal.setStringredient4((String) ingredientsTable.getValueAt(3, 0));  
 meal.setStringredient5((String) ingredientsTable.getValueAt(4, 0));  
 meal.setStringredient6((String) ingredientsTable.getValueAt(5, 0));  
 meal.setStringredient7((String) ingredientsTable.getValueAt(6, 0));  
 meal.setStringredient8((String) ingredientsTable.getValueAt(7, 0));  
 meal.setStringredient9((String) ingredientsTable.getValueAt(8, 0));  
 meal.setStringredient10((String) ingredientsTable.getValueAt(9, 0));  
 meal.setStringredient11((String) ingredientsTable.getValueAt(10, 0));  
 meal.setStringredient12((String) ingredientsTable.getValueAt(11, 0));  
 meal.setStringredient13((String) ingredientsTable.getValueAt(12, 0));  
 meal.setStringredient14((String) ingredientsTable.getValueAt(13, 0));  
 meal.setStringredient15((String) ingredientsTable.getValueAt(14, 0));  
 meal.setStringredient16((String) ingredientsTable.getValueAt(15, 0));  
 meal.setStringredient17((String) ingredientsTable.getValueAt(16, 0));  
 meal.setStringredient18((String) ingredientsTable.getValueAt(17, 0));  
 meal.setStringredient19((String) ingredientsTable.getValueAt(18, 0));  
 meal.setStringredient20((String) ingredientsTable.getValueAt(19, 0));  
  
 meal.setStrmeasure1((String) ingredientsTable.getValueAt(0, 1));  
 meal.setStrmeasure2((String) ingredientsTable.getValueAt(1, 1));  
 meal.setStrmeasure3((String) ingredientsTable.getValueAt(2, 1));  
 meal.setStrmeasure4((String) ingredientsTable.getValueAt(3, 1));  
 meal.setStrmeasure5((String) ingredientsTable.getValueAt(4, 1));  
 meal.setStrmeasure6((String) ingredientsTable.getValueAt(5, 1));  
 meal.setStrmeasure7((String) ingredientsTable.getValueAt(6, 1));  
 meal.setStrmeasure8((String) ingredientsTable.getValueAt(7, 1));  
 meal.setStrmeasure9((String) ingredientsTable.getValueAt(8, 1));  
 meal.setStrmeasure10((String) ingredientsTable.getValueAt(9, 1));  
 meal.setStrmeasure11((String) ingredientsTable.getValueAt(10, 1));  
 meal.setStrmeasure12((String) ingredientsTable.getValueAt(11, 1));  
 meal.setStrmeasure13((String) ingredientsTable.getValueAt(12, 1));  
 meal.setStrmeasure14((String) ingredientsTable.getValueAt(13, 1));  
 meal.setStrmeasure15((String) ingredientsTable.getValueAt(14, 1));  
 meal.setStrmeasure16((String) ingredientsTable.getValueAt(15, 1));  
 meal.setStrmeasure17((String) ingredientsTable.getValueAt(16, 1));  
 meal.setStrmeasure18((String) ingredientsTable.getValueAt(17, 1));  
 meal.setStrmeasure19((String) ingredientsTable.getValueAt(18, 1));  
 meal.setStrmeasure20((String) ingredientsTable.getValueAt(19, 1));  
  
 ingredientsTable.getValueAt(0, 0);  
  
 return meal;  
  
 }  
   
 //Μέθοδος που διαγράφει τα δεδομένα μιας συνταγής από το αντικείμενο Meal  
 public static Meal deleteMealData(Meal meal) {  
 // Αρχικοποίηση των πεδίων της συνταγής με κενές τιμές  
 meal.setStrcategory("");  
 meal.setIdmeal("");  
 meal.setStrarea("");  
 meal.setStrinstructions("");  
 meal.setStrmealthumb("");  
 meal.setStryoutube("");  
 meal.setStringredient1("");  
 meal.setStringredient2("");  
 meal.setStringredient3("");  
 meal.setStringredient4("");  
 meal.setStringredient5("");  
 meal.setStringredient6("");  
 meal.setStringredient7("");  
 meal.setStringredient8("");  
 meal.setStringredient9("");  
 meal.setStringredient10("");  
 meal.setStringredient11("");  
 meal.setStringredient12("");  
 meal.setStringredient13("");  
 meal.setStringredient14("");  
 meal.setStringredient15("");  
 meal.setStringredient16("");  
 meal.setStringredient17("");  
 meal.setStringredient18("");  
 meal.setStringredient19("");  
 meal.setStringredient20("");  
 meal.setStrmeasure1("");  
 meal.setStrmeasure2("");  
 meal.setStrmeasure3("");  
 meal.setStrmeasure4("");  
 meal.setStrmeasure5("");  
 meal.setStrmeasure6("");  
 meal.setStrmeasure7("");  
 meal.setStrmeasure8("");  
 meal.setStrmeasure9("");  
 meal.setStrmeasure10("");  
 meal.setStrmeasure11("");  
 meal.setStrmeasure12("");  
 meal.setStrmeasure13("");  
 meal.setStrmeasure14("");  
 meal.setStrmeasure15("");  
 meal.setStrmeasure16("");  
 meal.setStrmeasure17("");  
 meal.setStrmeasure18("");  
 meal.setStrmeasure19("");  
 meal.setStrmeasure20("");  
   
 // Επιστροφή του αντικειμένου Meal με τα δεδομένα της συνταγής διαγραμμένα  
 return meal;  
 }  
  
}

## Ερώτημα Γ – Παρουσίαση στατιστικών στοιχείων γευμάτων

package mealsapp.Charts. PrintList.java

package mealsapp.Charts;  
  
import java.awt.Desktop;  
import mealsapp.MealClasses.Meal;  
import java.io.File;  
import mealsapp.DB.mealsQueryOperations;  
import java.util.List;  
import com.itextpdf.text.PageSize;  
import com.itextpdf.text.Paragraph;  
import com.itextpdf.text.Element;  
import com.itextpdf.text.pdf.PdfPCell;  
import com.itextpdf.text.pdf.PdfPTable;  
import javax.swing.JFileChooser;  
import javax.swing.JOptionPane;  
import javax.swing.filechooser.FileNameExtensionFilter;  
import mealsapp.GUI.MainForm;  
import java.awt.Graphics2D;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import com.itextpdf.text.Document;  
import com.itextpdf.text.DocumentException;  
import com.itextpdf.text.pdf.BaseFont;  
import com.itextpdf.text.pdf.PdfContentByte;  
import com.itextpdf.text.pdf.PdfTemplate;  
import com.itextpdf.text.pdf.PdfWriter;  
import java.awt.geom.Rectangle2D;  
import java.io.IOException;  
  
public class PrintList {  
  
 // Μέθοδος για εξαγωγή σε PDF ανάλογα με την επιλογή του χρήστη (3Δ Στήλες, Pie Chart ή Λίστα)  
 public void exportToPDF(int userView) throws FileNotFoundException, DocumentException, IOException {  
  
 JFileChooser fileChooser = new JFileChooser();  
 fileChooser.setDialogTitle("Save PDF File");  
 fileChooser.setFileSelectionMode(JFileChooser.FILES\_ONLY);  
 fileChooser.setFileFilter(new FileNameExtensionFilter("PDF Files", "pdf"));  
  
 int userSelection = fileChooser.showSaveDialog(null);  
 if (userSelection == JFileChooser.APPROVE\_OPTION) {  
  
 File pdfFile = fileChooser.getSelectedFile();  
 // Προσθήκη επέκτασης .pdf στο όνομα αρχείου, αν δεν υπάρχει ήδη  
 if (!pdfFile.getName().toLowerCase().endsWith(".pdf")) {  
 pdfFile = new File(pdfFile.getAbsolutePath() + ".pdf");  
 }  
 if (pdfFile.exists()) {  
 // Ερώτηση αν θέλετε να αντικαταστήσετε το αρχείο PDF σε περίπτωση που υπάρχει ήδη.  
 int replaceOption = MainForm.wantQuestionYesNo("The file already exists. Do you want to replace it?");  
 System.out.println(replaceOption);  
 if (replaceOption == 1) {  
 } else {  
 if (userView == 0) {  
 generateColomunPDF(pdfFile);  
 } else if (userView == 1) {  
 generatePiePDF(pdfFile);  
  
 } else if (userView == 2) {  
 pdfColumnContent(pdfFile);  
 }  
 }  
 } else {  
 if (userView == 0) {  
 generateColomunPDF(pdfFile);  
 } else if (userView == 1) {  
 generatePiePDF(pdfFile);  
 } else if (userView == 2) {  
 pdfColumnContent(pdfFile);  
  
 }  
  
 }  
 }  
 }  
  
 //Mέθοδος που χρησιμοποιείται για να δημιουργήσει έναν πίνακα με τα δεδομένα των γευμάτων και να τα εξάγει σε μορφή PDF.  
 public static void pdfColumnContent(File pdfFile) {  
  
 try {  
 // Λίστα με όλα τα γεύματα που έχουν καταγραφεί στη βάση δεδομένων, ταξινομημένα κατά φθίνουσα σειρά  
 List<Meal> meals = mealsQueryOperations.getbyDecOrder();// Λίστα με όλα τα γεύματα που έχουν καταγραφεί στη βάση δεδομένων, ταξινομημένα κατά φθίνουσα σειρά  
  
 // Δημιουργία ενός νέου κενού αρχείου PDF  
 Document document = new Document(PageSize.A4);// Δημιουργία ενός νέου κενού αρχείου PDF  
 // Αρχικοποίηση PdfWriter για να γραφεί το αρχείο pdf στον σκληρό δίσκο  
 PdfWriter.getInstance(document, new FileOutputStream(pdfFile)); // Αρχικοποίηση PdfWriter για να γραφεί το αρχείο pdf στον σκληρό δίσκο  
  
 // Άνοιγμα του κενού αρχείου PDF  
 document.open();  
 // Δημιουργία πίνακα με δύο στήλες  
 PdfPTable table = new PdfPTable(2);  
 // Ορισμός του ποσοστού του πίνακα που πρέπει να καλύπτει το PDF  
 table.setWidthPercentage(100);  
 // Δημιουργία του κελιού με την επικεφαλίδα της πρώτης στήλης  
 PdfPCell cell1 = new PdfPCell(new Paragraph("Meal"));  
 // Δημιουργία του κελιού με την επικεφαλίδα της δεύτερης στήλης  
 PdfPCell cell2 = new PdfPCell(new Paragraph("Times of visit"));  
 // Ορισμός της οριζόντιας στοίχισης του κελιού της πρώτης στήλης  
 cell1.setHorizontalAlignment(Element.ALIGN\_LEFT);  
 // Ορισμός της οριζόντιας στοίχισης του κελιού της δεύτερης στήλης  
 cell2.setHorizontalAlignment(Element.ALIGN\_LEFT);  
 // Προσθήκη του κελιού της πρώτης στήλης στον πίνακα  
 table.addCell(cell1);  
 // Προσθήκη του κελιού της δεύτερης στήλης στον πίνακα  
 table.addCell(cell2);  
  
 // Επανάληψη για όλα τα γεύματα που βρίσκονται στη λίστα  
 for (int i = 0; i < meals.size(); i++) {  
 // Δημιουργία κελιών για τα δεδομένα του πίνακα  
 PdfPCell dataCell1 = new PdfPCell(new Paragraph(meals.get(i).getStrmeal()));  
 PdfPCell dataCell2 = new PdfPCell(new Paragraph(meals.get(i).getTimesofvisit().toString()));  
  
 dataCell1.setHorizontalAlignment(Element.ALIGN\_LEFT);  
 dataCell2.setHorizontalAlignment(Element.ALIGN\_LEFT);  
  
 dataCell1.setPaddingBottom(10);  
 dataCell2.setPaddingBottom(10);  
  
 table.addCell(dataCell1);  
 table.addCell(dataCell2);  
 }  
  
 // Φόρτωση της γραμματοσειράς Harlow Solid Italic  
 var fontPath = "resources/Harlow\_Solid\_Italic.ttf";  
 var bf = BaseFont.createFont(fontPath, BaseFont.IDENTITY\_H, BaseFont.EMBEDDED);  
 // Δημιουργία ενός αντικειμένου font με γραμματοσειρά Harlow Solid Italic  
 var font = new com.itextpdf.text.Font(bf, 22);  
 // Δημιουργία επικεφαλίδας με γραμματοσειρά Harlow Solid Italic  
 var title = new com.itextpdf.text.Paragraph("Meals App Table Report", font);  
 title.setAlignment(com.itextpdf.text.Element.ALIGN\_CENTER);//Στοίχιση στο κέντρο  
 document.add(title);  
 document.add(new Paragraph(" "));  
 document.add(table);  
 document.close();  
 // Εμφάνιση μηνύματος επιτυχημένης αποθήκευσης  
 JOptionPane.showMessageDialog(null, "Saved successfully", "Message", JOptionPane.INFORMATION\_MESSAGE);  
 // Άνοιγμα του αρχείου PDF για προεπισκόπηση μετά την επιτυχημένη αποθήκευση  
 Desktop.getDesktop().open(pdfFile);  
 } catch (Exception e) {  
  
 }  
  
 }  
  
 //Μέθοδος για τη δημιουργία ενός PDF αρχείου με ένα διάγραμμα στήλης.  
 public static void generateColomunPDF(File pdfFile) throws FileNotFoundException, DocumentException, IOException {  
 try {  
 // Δημιουργία κενού αρχείου PDF με ρυθμίσεις σελίδας Α4 landscape  
 Document document = new Document(PageSize.A4.rotate());  
 // Προετοιμασία του αντικειμένου που θα εγγράψει το αρχείο PDF στο δίσκο  
 PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream(pdfFile));  
  
 document.open();  
  
 // Προετοιμασία ενός αντικειμένου για να σχεδιαστεί το γράφημα σε αυτό  
 PdfContentByte contentByte = writer.getDirectContent();  
 PdfTemplate template = contentByte.createTemplate(500, 400);  
 // Προετοιμασία ενός αντικειμένου γραφικών για να σχεδιαστεί το γράφημα πάνω σε αυτό  
 @SuppressWarnings("deprecation")  
 Graphics2D graphics2d = template.createGraphicsShapes(500, 400);  
  
 Rectangle2D rectangle2d = new Rectangle2D.Double(0, 0, 500, 400);  
 // Σχεδίαση του γραφήματος με τη βοήθεια της κλάσης ColumnChart  
 ColumnChart.generateBarChart().draw(graphics2d, rectangle2d);  
  
 graphics2d.dispose();  
 // Προσθήκη του γραφήματος στο PDF στις δοθήσες συντεταγμένες   
 contentByte.addTemplate(template, 120, 100);  
  
 document.close();  
 writer.close();  
 // Άνοιγμα του αρχείου PDF για προεπισκόπηση μετά την επιτυχημένη αποθήκευση  
 Desktop.getDesktop().open(pdfFile);  
  
 } catch (Exception e) {  
 }  
  
 }  
  
 //Μέθοδος για τη δημιουργία ενός PDF αρχείου με ένα γράφημα τύπου πίτας  
 public static void generatePiePDF(File pdfFile) throws FileNotFoundException, DocumentException {  
  
 try {  
 // Δημιουργία και ανοίγματος του κεντρικού κειμένου του PDF αρχείου  
 Document document = new Document(PageSize.A4.rotate());  
 PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream(pdfFile));  
 document.open();  
 // Δημιουργία επιφάνειας σχεδίασης και προσαρμογή των διαστάσεών της  
 PdfContentByte contentByte = writer.getDirectContent();  
 PdfTemplate template = contentByte.createTemplate(500, 400);  
 @SuppressWarnings("deprecation")  
 Graphics2D graphics2d = template.createGraphicsShapes(500, 400);  
 // Δημιουργία του Pie Chart  
 Rectangle2D rectangle2d = new Rectangle2D.Double(0, 0, 500, 400);  
  
 PieChart.generateImagePieChart().draw(graphics2d, rectangle2d);  
  
 graphics2d.dispose();  
 // Προσθήκη του σχεδιασμένου Pie Chart στο PDF  
 contentByte.addTemplate(template, 160, 100);  
   
 document.close();  
 writer.close();  
 //Ανοιγμα του αρχείου στον υπολογιστή του χρήστη  
 Desktop.getDesktop().open(pdfFile);  
 } catch (Exception e) {  
 }  
 }  
}

package mealsapp.Charts. ColumnChart.java

/\*  
Αυτό το πακέτο αποτελεί μέρος της εφαρμογής MealsApp και αφορά τη δημιουργία  
ενός γραφήματος τύπου 3Δ στήλης χρησιμοποιώντας τη βιβλιοθήκη JFreeChart.  
\*/  
package mealsapp.Charts;  
  
import java.util.List;  
import javax.swing.JFrame;  
import mealsapp.DB.mealsQueryOperations;  
import mealsapp.MealClasses.Meal;  
import org.jfree.chart.ChartPanel;  
import org.jfree.chart.ChartFactory;  
import org.jfree.chart.JFreeChart;  
import javax.swing.JPanel;  
import org.jfree.data.category.DefaultCategoryDataset;  
  
  
public class ColumnChart extends JFrame {  
/\*  
 \* Η μέθοδος αυτή δημιουργεί και επιστρέφει ένα dataset το οποίο περιέχει τις πληροφορίες  
 \* για τα πιο συχνά επισκεπτόμενα γεύματα και πόσες φορές τα επισκέφθηκε ο χρήστης.  
 \*/  
 public static DefaultCategoryDataset columnDataset() {  
 List<Meal> orderedMeal = mealsQueryOperations.getbyDecOrder();  
 final DefaultCategoryDataset dataset = new DefaultCategoryDataset();  
 for (int i = 0; i < orderedMeal.size() && i <= 14; i++) {  
 dataset.addValue(orderedMeal.get(i).getTimesofvisit(), "", orderedMeal.get(i).getStrmeal());  
 }  
 return dataset;  
 }  
/\*  
 \* Η μέθοδος αυτή δημιουργεί και επιστρέφει ένα JPanel το οποίο περιέχει το γράφημα στήλης  
 \* που παράχθηκε από το dataset της columnDataset().  
 \*/  
 public static JPanel ColumnChart() {  
  
 JPanel chartPanel = new ChartPanel(generateBarChart());  
 chartPanel.setSize(814, 594);  
  
 return chartPanel;  
 }  
  
   
 /\*  
 \* Η μέθοδος αυτή δημιουργεί και επιστρέφει ένα αντικείμενο JFreeChart που αναπαριστά  
 \* το γράφημα στήλης με βάση το dataset της columnDataset().  
 \*/  
 public static JFreeChart generateBarChart() {  
   
 JFreeChart chart = ChartFactory.createBarChart3D(  
 "Most Visited Meals", // τίτλος του γραφήματος  
 "Meals", // ετικέτα τουν άξονα των κατηγοριών  
 "Times of Visit", // ετικέτα τουν άξονα των τιμών  
 columnDataset(), // τα δεδομένα  
 org.jfree.chart.plot.PlotOrientation.HORIZONTAL, // προσανατολισμός  
 false, // εμφάνιση λεζάντας  
 false,//εμφάνιση tooltips  
 false);//εμφάνιση urls  
 org.jfree.chart.axis.ValueAxis rangeAxis = chart.getCategoryPlot().getRangeAxis();// παίρνουμε τον άξονα τιμών του γραφήματος  
 rangeAxis.setAutoTickUnitSelection(false);//και απενεργοποιούμε την αυτόματη επιλογή μονάδων του άξονα τιμών για εμφάνιση ακέραιας μονάδας  
 return chart;  
 }  
   
   
   
   
}

package mealsapp.Charts. PieChart.java

/\*  
Αυτό το πακέτο αποτελεί μέρος της εφαρμογής MealsApp και   
δημιουργεί ένα διάγραμμα πίτας των πιο συχνά επισκεπτόμενων γευμάτων.  
\*/  
package mealsapp.Charts;  
  
  
import java.text.NumberFormat;  
import java.util.List;  
import mealsapp.DB.mealsQueryOperations;  
import mealsapp.MealClasses.Meal;  
import org.jfree.chart.ChartFactory;  
import org.jfree.chart.ChartPanel;  
import org.jfree.chart.JFreeChart;  
import org.jfree.data.general.DefaultPieDataset;  
import org.jfree.ui.ApplicationFrame;  
import javax.swing.JPanel;  
import org.jfree.chart.labels.PieSectionLabelGenerator;  
import org.jfree.chart.labels.StandardPieSectionLabelGenerator;  
import org.jfree.chart.plot.PiePlot;  
  
*/\*\*  
 \* Η κλάση PieChart δημιουργεί ένα διάγραμμα πίτας των πιο συχνά επισκεπτόμενων γευμάτων.  
 \*/*public class PieChart extends ApplicationFrame {  
 */\*\*  
 \* Η μέθοδος pieDataset() δημιουργεί τα δεδομένα του διαγράμματος πίτας.  
 \* Ανακτά τα 15 πιο επισκεπτόμενα γεύματα και τα προσθέτει στο dataset.  
 \*/* public static DefaultPieDataset pieDataset() {  
 List<Meal> orderedMeal = mealsQueryOperations.getbyDecOrder();  
 final DefaultPieDataset dataset = new DefaultPieDataset();  
 for (int i = 0; i < orderedMeal.size() && i <= 14; i++) {  
 dataset.setValue(orderedMeal.get(i).getStrmeal(), orderedMeal.get(i).getTimesofvisit());  
 }  
 return dataset;  
 }  
 */\*\*  
 \* Η μέθοδος PieChart() δημιουργεί ένα πάνελ που περιλαμβάνει το διάγραμμα πίτας και  
 \*επιστρέφει το πάνελ στο κυρίως πρόγραμμα για να το προβάλει.  
 \*/* public static JPanel PieChart() {  
 // Δημιουργεί το διάγραμμα πίτας .  
 JFreeChart chart = generateImagePieChart();  
 // Δημιουργεί ένα πάνελ που περιλαμβάνει το διάγραμμα πίτας.  
 JPanel chartPanel = new ChartPanel(chart);  
 chartPanel.setSize(814, 594);  
  
 // Ορίζουμε ένα formatter για τη μορφοποίηση των αριθμών του γραφήματος.  
 NumberFormat formatter = NumberFormat.getIntegerInstance();  
 // Δημιουργούμε έναν generator που θα φτιάχνει τις ετικέτες των τομέων της πίτας, βάσει των δεδομένων που δίνουμε.  
 PieSectionLabelGenerator labelGenerator = new StandardPieSectionLabelGenerator("{0} : {1}", formatter, formatter);  
 // Ορίζουμε τον generator ως generator ετικετών στο γράφημα.  
 ((PiePlot) chart.getPlot()).setLabelGenerator(labelGenerator);  
  
 return chartPanel;  
  
 }  
  
 public PieChart(String title) {  
 super(title);  
 }  
 // Κατασκευάζουμε το γράφημα της πίτας και επιστρέφουμε το αντικείμενο τύπου JFreeChart  
 public static JFreeChart generateImagePieChart() {  
   
 JFreeChart chart = ChartFactory.createPieChart3D(  
 "Most Visited Meals", // τίτλος του γραφήματος  
 pieDataset(), // δεδομένα  
 false, //εμφάνιση λεζάντας  
 true,//εμφάνιση tooltips  
 true);//εμφάνιση urls  
  
 return chart;  
 }  
}

## Ερώτημα Δ – Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής

|  |  |
| --- | --- |
| **Απαίτηση** | **Εικόνα** |
| R1 | ***Αρχική οθόνη της εφαρμογής Meals App.***  Ο χρήστης εχει 4 επιλογές.  Πατώντας στο Search ο χρήστης μεταφέρεται στην οθόνη αναζήτησης ενός γεύματος.  Πατώντας στο Statistics ο χρήστης μεταφέρεται στην οθόνη προβολής στατιστικών.  Πατώντας στο Advanced Search ο χρήστης μεταφέρεται στην οθόνη όπου μπορεί να δει τα γεύματα ανά κατηγορία, ανά χώρα, ανά συστατικό καθώς επίσης να κάνει και εξειδικευμένη αναζήτηση βάση ονόματος για την εμφάνιση σε λίστα γεύματα από την ΑΡΙ ή την ΒΔ  Τέλος πατώντας το Exit κλείνει η εφαρμογή και η ΒΔ. |

|  |  |
| --- | --- |
| R2 | **Οθόνη εμφάνισης γεύματος.**  Εδώ ο χρήστης έχει τις εξής επιλογές:  Γράφοντας στο searchbox το όνομα ενός γεύματος που θέλει να αναζητήσει και πατώντας το κουμπί <<Ι Am Feeling Lucky>> το σύστημα θα επιστρέψει την πρώτη επιλογή που ταιριάζει στο όνομα που έβαλε ο χρήστης. Σε περίπτωση που δεν υπάρχει ούτε στην ΒΔ ούτε στην ΑΡΙ τότε επιστρέφει popup μήνυμα σφάλματος.  Επιλέγοντας το κουμπί <<Random Meal>> το πρόγραμμα επιστρέφει στο χρήστη ένα τυχαίο γεύμα.  Επιλέγοντας το κουμπί <<Edit>> δίνεται στο χρήστη η δυνατότητα να επεξεργαστεί όλα τα στοιχεία και στη συνέχεια να τα αποθηκεύσει στη ΒΔ  Επιλέγοντας το κουμπί <<Save>> ο χρήστης αποθηκεύει στη ΒΔ το γεύμα ή τις τροποποιήσεις που έχει κάνει πάνω στο γεύμα.  Επιλέγοντας το κουμπί <<Delete>> ο χρήστης εμφανίζεται ένα popup παράθυρο όπου μπορεί να επιλέξει εάν επιθυμεί να διαγράφει το γεύμα από την βάση οριστικά ή εάν επιθυμεί να διατηρήσει τα στατιστικά στοιχεία του γεύματος.  Κάνοντας διπλό κλικ ο χρήστης πάνω σε κάποιο συστατικό ενός γεύματος, εμφανίζεται μια οθόνη η οποία περιέχει σε μορφή λίστας όλα τα διαθέσιμα γεύματα από την ΑΡΙ τα οποία έχουν μέσα τους το συστατικό που επέλεξε ο χρήστης. |

|  |  |
| --- | --- |
| R3 | **Οθόνη εμφάνισης γευμάτων ανά κατηγορία.**  Ο χρήστης επιλέγοντας στο αριστερό μέρος της οθόνης μια κατηγορία εμφανίζονται στο δεξί μέρος όλα τα γεύματα που ανήκουν σε αυτή. Στη συνέχεια ο χρήστης πατώντας πάνω σε ένα γεύμα ανοίγει η οθόνη αναζήτησης γεύματος και εμφανίζονται όλα τα δεδομένα του. Σε περίπτωση που το γεύμα είναι αποθηκευμένο και στη ΒΔ τότε εμφανίζεται μήνυμα στο χρήστη να επιλέξει από που θέλει να δει τα δεδομένα.    **Οθόνη εμφάνισης γευμάτων ανά περιοχή.**  Σε αυτή την οθόνη δίνεται στο χρήστη η δυνατότητα να επιλέξει στο αριστερό μέρος μια περιοχή και να εμφανιστούν στα δεξιά τα γεύματα. Στη συνέχεια ο χρήστης πατώντας πάνω σε ένα γεύμα ανοίγει η οθόνη αναζήτησης γεύματος και εμφανίζονται όλα τα δεδομένα του. Σε περίπτωση που το γεύμα είναι αποθηκευμένο και στη ΒΔ τότε εμφανίζεται μήνυμα στο χρήστη να επιλέξει από που θέλει να δει τα δεδομένα.    **Οθόνη εμφάνισης λίστας γευμάτων ανά συστατικό.**  Στην συγκεκριμένη οθόνη ο χρήστης μπορεί να επιλέξει ένα συστατικό και στα δεξιά θα εμφανιστούν τα γεύματα που το περιέχουν. Στη συνέχεια ο χρήστης πατώντας πάνω σε ένα γεύμα ανοίγει η οθόνη αναζήτησης γεύματος και εμφανίζονται όλα τα δεδομένα του. Σε περίπτωση που το γεύμα είναι αποθηκευμένο και στη ΒΔ τότε εμφανίζεται μήνυμα στο χρήστη να επιλέξει από που θέλει να δει τα δεδομένα.    **Οθόνη εξειδικευμένης αναζήτησης γευμάτων.**  Στην συγκεκριμένη οθόνη ο χρήστης μπορεί να πατήσει το κουμπί της αναζήτησης χωρίς να έχει βάλει κάποιο όνομα και το πρόγραμμα θα εμφανίσει μια λίστα με γεύματα από την ΑΡΙ. Εάν επιλέξει την Database και πατήσει αναζήτηση θα επιστρέψει όλα τα γεύματα που υπάρχουν στην ΒΔ. Σε περίπτωση που επιθυμεί κάποιο συγκεκριμένο γεύμα μπορεί να γράψει π.χ. beef και θα εμφανίσει όλα τα γεύματα που στο όνομά τους περιέχουν την λέξη beef. Στη συνέχεια ο χρήστης πατώντας πάνω σε ένα γεύμα ανοίγει η οθόνη αναζήτησης γεύματος και εμφανίζονται όλα τα δεδομένα του. Σε περίπτωση που γίνεται αναζήτηση στην ΑΡΙ και το γεύμα είναι αποθηκευμένο και στη ΒΔ τότε εμφανίζεται μήνυμα στο χρήστη να επιλέξει από που θέλει να δει τα δεδομένα. Ενώ στην περίπτωση που έχει επιλεγεί η Database τότε επιστρέφει τα δεδομένα από αυτή. |
| R4 | **Οθόνη προβολής στατιστικών δεδομένων (3D Columns) για τα γεύματα που έχουν προβληθεί.**  Ο χρήστης επιλέγοντας το combobox μπορεί να επιλέξει ανάμεσα σε 3 τρόπους προβολής. Πρώτος τρόπος είναι η απεικόνιση σε 3D Columns όπως φαίνεται και στην παραπάνω εικόνα. Στο άνω δεξιά μέρος της οθόνης, ο χρήστης πιέζοντας το κουμπί <<SaveAs PDF>> μπορεί να αποθηκεύσει τα στατιστικά με την επιλεγμένη εμφάνιση. Η αποθήκευση γίνεται σε θέση που υποδεικνύει ο χρήστης στο popup panel που εμφανίζεται , με την ολοκλήρωση της αποθήκευσης το αρχείο ανοίγει αυτόματα για προεπισκόπηση.    ***Οθόνη προβολής στατιστικών δεδομένων (3D Pie) για τα γεύματα που έχουν προβληθεί.***  Επιλέγοντας ο χρήστης στο combobox τον δεύτερο τρόπο(3D Pie) εμφανίζεται η παραπάνω εικόνα. Στο άνω δεξιά μέρος της οθόνης, ο χρήστης μπορεί και εδώ πιέζοντας το κουμπί <<SaveAs PDF>> να αποθηκεύσει τα στατιστικά με την επιλεγμένη εμφάνιση. Η αποθήκευση γίνεται σε θέση που υποδεικνύει ο χρήστης στο popup panel που εμφανίζεται και με την ολοκλήρωση της αποθήκευσης το αρχείο ανοίγει αυτόματα για προεπισκόπηση.    ***Οθόνη προβολής στατιστικών δεδομένων (Table) για τα γεύματα που έχουν προβληθεί.***  Επιλέγοντας ο χρήστης στο combobox τον τρίτο τρόπο(Table) εμφανίζεται η παραπάνω εικόνα. Στο άνω δεξιά μέρος της οθόνης, ο χρήστης μπορεί και εδώ πιέζοντας το κουμπί <<SaveAs PDF>> να αποθηκεύσει τα στατιστικά με την επιλεγμένη εμφάνιση. Η αποθήκευση γίνεται σε θέση που υποδεικνύει ο χρήστης στο popup panel που εμφανίζεται και με την ολοκλήρωση της αποθήκευσης το αρχείο ανοίγει αυτόματα για προεπισκόπηση. |

# ΚΡΙΤΙΚΟΣ ΑΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΕΡΓΟΥ

Βασικά προβλήματα που κληθήκαμε να αντιμετωπίσουμε ήταν η κατανόηση των db που χρησιμοποιούν derby, παρακολουθήσαμε δεκάδες video του ΕΑΠ αλλά στις πιο πολλές περιπτώσεις η φαινομενική λύση δεν ήταν λειτουργική σε εμάς.

Οι db συνήθως δεν εμφανίζονταν στην καρτέλα “Services” ή δεν συνδεόνταν.  
Ένα άλλο πρόβλημα ήταν η επιλογή των σωστών libraries διότι ενώ υπήρχαν προτάσεις από το ΕΑΠ οι καινούργιες εκδόσεις του Netbeans πολλές φορές δεν υποστηρίζονται.

Γενικά το Neabeans δεν παρέχει τις ίδιες ευκολίες που έχει το Intellij αλλά η βιβλιογραφία του ΕΑΠ είναι πιο πολύ προσανατολισμένη προς το Neabeans, θα πρέπει να δοθούν νέες κατευθύνσεις ώστε στις εργασίες να χρησιμοποιούνται σύγχρονα εργαλεία και με αυτό τον τρόπο να ανεβαίνει το επίπεδο και ο παραγωγικός ανταγωνισμός μεταξύ φοιτητών.

Ένα άλλο πρόβλημα ήταν οι δεξιότητές μας στην Java, αυτό είχε αποτέλεσμα πολλές φορές να έχουμε αδυναμία στην επίλυση ενός προβλήματος ή να αναπτύσσεται περίπλοκος κώδικας για ένα απλοϊκό ζήτημα.

Όλα τα παραπάνω είχαν σαν αποτέλεσμα να έχουμε μεγάλες αποκλίσεις από το γενικό μας χρονοδιάγραμμα και να αφιερώνουμε μέχρι και δεκάδες ώρες μέσα στην ημέρα για να φτάσουμε στην υλοποίηση.  
Αν θέλαμε να κάνουμε μόνο τις απαιτήσεις που περιγράφει η εκφώνηση θα είχαμε τελειώσει πιο σύντομα αλλά θέλαμε να κάνουμε μία καλή και αξιόλογη εργασία κατανοώντας σταδιακά τα εργαλεία που είχαμε διαθέσιμα.

Το εργαλείο Trello μας βοήθησε να οραματιστούμε και να οργανώσουμε την αρχή της εργασίας αλλά στην πορεία και εξαιτίας των αυξημένων απαιτήσεων και στόχων δεν κατέστη δυνατό στο σύντομο διάστημα της εκπόνησης να το παρακολουθήσουμε.`

Οι κίνδυνοι που κληθήκαμε να αντιμετωπίσουμε ήταν τα λογικά λάθη, δηλαδή ενώ είχαμε ένα κώδικα που δεν «κράσαρε» δίνοντας την εντύπωση ότι όλα ήταν σωστά, είχε ως αποτέλεσμα να γυρνάμε πίσω στον σχεδιασμό για να αντιμετωπίσουμε τα ζητήματα που προκύπταν από προηγούμενα stories .

Με μεγάλη έκπληξη διακρίναμε την μεγάλη κοινότητα προγραμματιστών - και όχι μόνο – που είναι πάντα πρόθυμοι να σε κατευθύνουν με τις σωστές οδηγίες.

Στο επόμενο έργο θα αλλάζαμε το IDE που χρησιμοποιήσαμε και άλλες τεχνολογίες για το GUI όπως Java FX μέσω του scene builder.

Τέλος, πέρα από τις δυσκολίες που αντιμετωπίσαμε μπορούμε να διακρίνουμε μία σημαντική εξέλιξη των δεξιοτήτων μας στον προγραμματισμό και εκτιμήσαμε ιδιαίτερα την συνεργασία μεταξύ μας. Όλοι μας ανεξαρτήτως εμπειρίας συμμετείχαν ενεργά στην εργασία, ο καθένας με τον δικό του τρόπο, που τελικά είναι και το πιο σημαντικό σε μία ομαδική εργασία.

# VIDEO

<https://www.youtube.com/watch?v=zteb_6k1nrA>

# ΑΝΑΦΟΡΕΣ

* Java API for JSON processing (JSON-P) tutorial [ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://docs.oracle.com/javaee/7/tutorial/jsonp.htm>

[Πρόσβαση Φεβρουάριο 2023]

* Derby DB documentation[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://db.apache.org/derby/manuals/index.html#docs_10.14>

[Πρόσβαση Φεβρουάριο 2023]

* Derby DB tutorial[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.tutorialspoint.com/apache_derby/apache_derby_quick_guide.htm>

[Πρόσβαση Φεβρουάριο 2023]

* Deitel, P. & Deitel, H., 2015. *Java Προγραμματισμός.* Δέκατη έκδοση, 3η Ελληνική Επανέκδοση επιμ. Αθήνα: Εκδόσεις Γκιούρδας.
* ΕΑΠ, 2023. Πρόσθετο υλικό Ενότητας ΠΛΗ24 και Γραπτές εργασίες προηγούμενων ετών. [Ηλεκτρονικό]

Διαθέσιμο στο: https://study.eap.gr

[Πρόσβαση Ιανουάριος 2023-Φεβρουάριος 2023-Μάρτιος 2023].

* Δρόσος, Ν. & Θάνος, Ά., 2012. Εφαρμογές ΒΔ με χρήση του JPA: Java Persistence API. Πάτρα: ΕΑΠ.
* Θάνος, Ά., 2012. Γραφικά Περιβάλλοντα με Swing. Πάτρα, ΕΑΠ.
* Σφέτσος, Π., 2015. Συνοπτική Περιγραφή της μεθοδολογίας Scrum. ΕΔΥ επιμ. Πάτρα: ΕΑΠ.
* Φιτσιλής, Π., 2007. Ανάπτυξη συστήματος λογισμικού με UML και Java. ΕΔΥ επιμ. Πάτρα: ΕΑΠ.
* Χατζηγεωργίου, Α., 2011. Εισαγωγή στις ευέλικτες μεθοδολογίες ανάπτυξης λογισμικού,(Agile Software Development), SCRUM. Πάτρα, Πανεπιστήμιο Πατρών.
* ﻿Java Programming Tutorials από τον Derek Banas: [ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=TBWX97e1E9g&list=PLE7E8B7F4856C9B19>

[Πρόσβαση Φεβρουάριο 2023]

* ﻿Java Swing: Η επίσημη τεκμηρίωση της Java Swing [ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://docs.oracle.com/en/java/javase/16/docs/api/java.desktop/javax/swing/package-summary.html>

[Πρόσβαση Φεβρουάριο 2023]

* ﻿Tech Primers: [ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/channel/UCs6nmQViDpUw0nuIx9c_WvA>

[Πρόσβαση Φεβρουάριο 2023]

* ﻿Durga Software Solutions: [ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/user/durgasoftware>

[Πρόσβαση Φεβρουάριο 2023]

* Scrolls JTable to the selected row. - Java Swing[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <http://www.java2s.com/example/java/swing/scrolls-jtable-to-the-selected-row.html>

[Πρόσβαση Φεβρουάριο 2023]

* Multi Dimensional ArrayList in Java[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.baeldung.com/java-multi-dimensional-arraylist>

[Πρόσβαση Φεβρουάριο 2023]

* NetBeans/JavaDB(Derby Setup for Home) [ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://home.csulb.edu/~mopkins/cecs323/netbeans.shtml>

[Πρόσβαση Φεβρουάριο 2023]

* Planning Poker[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.mountaingoatsoftware.com/agile/planning-poker>

[Πρόσβαση Φεβρουάριο 2023]

* Java Tutorial[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.tutorialspoint.com/index.htm>

[Πρόσβαση Φεβρουάριο 2023]

* 2012 - Εφαρμογές ΒΔ με χρήση τουJPA: Java Persistence API[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=MpxkaMgiLAk>

[Πρόσβαση Φεβρουάριο 2023]

* 2021 - Java Persistence API (JPA) Example[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=5jR0Ikpyqrg>

[Πρόσβαση Φεβρουάριο 2023]

* Σύνδεση Java IntelliJ με DerbyDB[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=od3-Y0X0WcM&t=393s>

[Πρόσβαση Φεβρουάριο 2023]

* Using Apache Derby (Java DB) from JDBC[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=sOv0DIHzNUU>

[Πρόσβαση Φεβρουάριο 2023]

* Creating a Database with JDBC using Apache Derby in IntelliJ IDEA[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=r9RZJChQyGo>

[Πρόσβαση Φεβρουάριο 2023]

* 2012-ΤΔ09-Σχεδίαση GUI Παραδείγματα σχεδίασης GUI[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=pizTFoBW_E8>

[Πρόσβαση Φεβρουάριο 2023]

* 2013 - ΤΔ12 GUI) Live Demo[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=lIZi68oChNg>

[Πρόσβαση Φεβρουάριο 2023]

* 2013 - ΠΑΡΟΥΣΙΑΣΗ ΝΕΤΒΕANS BINDINGS[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=HjSuhzP-e_c>

[Πρόσβαση Φεβρουάριο 2023]

* Making a Simple GUI App with NetBeans[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=JZTIIchUa6E>

[Πρόσβαση Φεβρουάριο 2023]

* Java GUI in Netbeans[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=GZ9MT2myBf8>

[Πρόσβαση Φεβρουάριο 2023]

* Creating First Java Swing GUI Application with NetBeans IDE[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=YSpqHOwYrk4>

[Πρόσβαση Φεβρουάριο 2023]

* How to Build a Swing GUI with IntelliJ IDEA[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=-SmNpKskfJc>

[Πρόσβαση Φεβρουάριο 2023]

* 2020-ΤΔ17-Introduction\_to\_TRELLO[ΗΛΕΚΤΡΟΝΙΚΑ]

Διαθέσιμο στο: <https://www.youtube.com/watch?v=oy_b2F1jrY8>

[Πρόσβαση Φεβρουάριο 2023]