ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS

ΠΜΣ «Πληροφοριακά Συστήματα και Υπηρεσίες»
Ειδίκευση «Προηγμένα Πληροφοριακά Συστήματα»

Μάθημα: **Η Γλώσσα Προγραμματισμού Java**

Υπεύθυνοι: Δρ. Ανδριάνα Πρέντζα
Δρ. Βασιλική Κούφη
Δρ. Ανδρέας Μενύχτας

Θέμα: Meal Lab APP σε Java (με χρήση Maven και JavaFX)

Φοιτητες: Γεράσιμος Πλατής – Βασίλης Χαχούλης
Α.Μ.: me25081 – me25095

Ημερομηνία: Ιανουάριος-Φεβρουάριος 2026

Αναφορά Εργασίας

---

1. Περιγραφή της χρήσης του REST API και της υλοποίησης των POJOs

Χρήση REST API (TheMealDB) με Retrofit

Η επικοινωνία με το εξωτερικό API TheMealDB υλοποιήθηκε με τη χρήση της βιβλιοθήκης Retrofit 2. Η επιλογή της Retrofit έγινε διότι απλοποιεί σημαντικά τη διαδικασία των HTTP κλήσεων, μετατρέποντας το API σε ένα Java Interface (MealApiInterface).

MealApiInterface.java

Οι βασικές λειτουργίες ορίστηκαν ως μέθοδοι στο Interface με χρήση annotations (π.χ. @GET):

- **searchMeals**: @GET("search.php") για αναζήτηση με όνομα.
- **FilterMealsByMain**: @GET("filter.php")
  - Για αναζήτηση με Υλικό (@Query("i") String ingrediant)
  - Για αναζήτηση με Περιοχή (@Query("a") String area)
  - Για αναζήτηση με Όνομα (@Query("s") String mealName)
- **GetReceipeById**: @GET("lookup.php") για εμφάνιση των λεπτομερείων της συνταγής με την χρήση getMealById(@Query("i") String mealId)
- **getRandomMeal**: @GET("random.php") για λήψη τυχαίας συνταγής.

MealService .java

Η κλάση MealService λειτουργεί ως ενδιάμεσος Wrapper, εκτελώντας τις κλήσεις (Call <MealResponse>) και επιστρέφοντας τα αποτελέσματα. Η Retrofit διαχειρίζεται αυτόματα τη σύνδεση και σε συνεργασία με τον Jackson - Connverter, μετατρέπει (deserialize) το response απευθείας σε αντικείμενα της Java (MealResponse,Recipe), χωρίς να χρειάζεται χειροκίνητο parsing.

Υλοποίηση POJOs (Plain Old Java Objects)

Για τη μοντελοποίηση των δεδομένων δημιουργήθηκαν οι εξής κλάσεις στο πακέτο model:

- **MealResponse**: Είναι μια κλάση που διευκολύνει την αποσειριοποίηση (deserialization) στοχεύοντας μέσα στο Array "meals".
- **Recipe**: Η κύρια κλάση που αναπαριστά μια συνταγή.
  Περιέχει πεδία :
  - idMeal (κωδικός),
  - strMeal (όνομα),
  - strCategory (κατηγορία),
  - strArea (κουζίνα),
  - strInstructions (οδηγίες) και
  - strMealThumb (URL εικόνας).

  Χρησιμοποιήθηκαν annotations @JsonAlias για να αντιστοιχιστούν τα πεδία του JSON (που έχουν προθέματα όπως "str") σε πιο καθαρά ονόματα μεταβλητών Java.

- **Ingredient:** Η κλάση αυτή λειτουργεί ως βοηθητικό μοντέλο για την αποθήκευση των υλικών μιας συνταγής και των αντίστοιχων δοσολογιών τους.

  Μία από τις βασικότερες προκλήσεις της υλοποίησης ήταν η ιδιόμορφη δομή του API TheMealDB, το οποίο επιστρέφει τα υλικά και τις ποσότητες ως ξεχωριστά πεδία (strIngredient1, strIngredient2, …, strIngredient20 και αντίστοιχα strMeasure1, strMeasure2, …).Τα πεδία αυτά επιστρέφονται στο Response της κλήσης setDynamicField με κενές τιμές ή με τιμές NULL σε τυχαία σείρα.

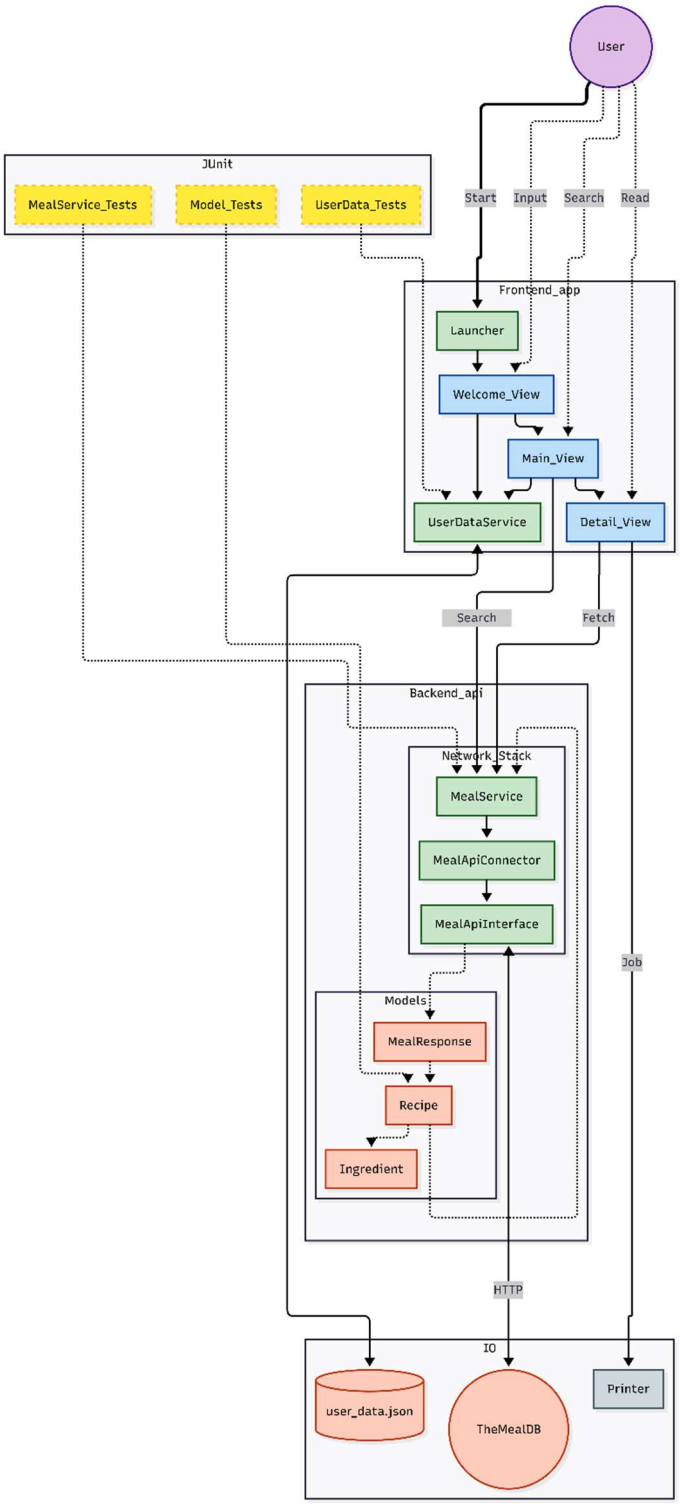
  Για τον λόγο αυτό, εφαρμόστηκε το annotation @JsonAnySetter της βιβλιοθήκης Jackson. Με την χρήση του, όλα τα δυναμικά πεδία του JSON συλλέγονται σε ένα Map<String, String>, χωρίς να απαιτείται αρχικοποιήση των πεδίων.

  Στη συνέχεια το περιεχόμενο του map επεξεργάζεται μόνο όταν ζητηθούν οι λεπτομέρειες της συνταγής, δημιουργώντας δυναμικά μια καθαρή και δομημένη λίστα List<Ingredient>. Η προσέγγιση αυτή μειώνει σημαντικά την πολυπλοκότητα της κλάσης Recipe και προσφέρει μεγαλύτερη ευελιξία σε ενδεχόμενες αλλαγές της δομής και της χρήση του API.

2. Αρχιτεκτονική & Λειτουργικότητα

Αρχιτεκτονική Εφαρμογής

Η εφαρμογή βασίστηκε στο πρότυπο MVC (Model-View-Controller) για τον διαχωρισμό της λογικής από τη διεπαφή χρήστη:



- **Model**: Οι κλάσεις Recipe, Ingredient, MealResponse και η υπηρεσία UserDataService που διαχειρίζεται τα τοπικά δεδομένα (Αγαπημένα, Ιστορικό).

- **View**: Τα αρχεία FXML (main_view.fxml, welcome_view.fxml, recipe_details.fxml) που ορίζουν τη δομή του User Interface και το αρχείο CSS (style.css) για τη μορφοποίηση.
- **Controller**: Οι κλάσεις MainController, WelcomeController και RecipeDetailsController που διαχειρίζονται τα events του χρήστη και ενημερώνουν το User Interface .

Ανάλυση Λειτουργιών

- Αναζήτηση: Υλοποιήθηκαν τεσσερίς τρόποι αναζήτησης (Όνομα, Υλικό, Κατηγορία, Περιοχή). Η επιλογή γίνεται μέσω ComboBox, το οποίο δυναμικά εμφανίζει/κρύβει τα αντίστοιχα πεδία εισαγωγής (TextField ή λίστα κατηγοριών) .
- Αγαπημένα & Ιστορικό: Χρησιμοποιήθηκε η κλάση UserDataService ως Singleton. Τα δεδομένα αποθηκεύονται τοπικά σε αρχείο JSON (user_data.json) ώστε να διατηρούνται μετά την επανεκκίνηση της εφαρμογής.

User Interface

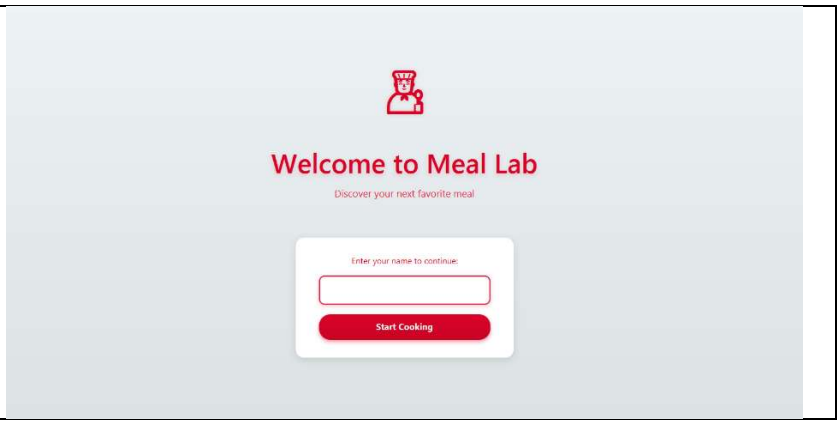
- Χρήση Accordion για την οργάνωση των ενοτήτων (Search, Favorites, History) ώστε να εξοικονομείται χώρος.
- Responsive σχεδιασμός με SplitPane για τον διαχωρισμό λίστας αποτελεσμάτων και λεπτομερειών.
- Εφαρμογή CSS για μορφοποιήση της εφαρμογής και χρήση Alert Dialogs για την ενημέρωση του χρήστη.
- Clean States: Όταν οι πίνακες είναι άδειοι, εμφανίζουμε κατάλληλα μηνύματα ("Placeholder text") ώστε ο χρήστης να μην βλέπει απλώς ένα κενό κουτί και να μπερδεύεται.
- Shortcuts: Προσθέσαμε συντομεύσεις πληκτρολογίου για ευκολία (ENTER για είσοδο, ESC για έξοδο), όπως έχουν οι σύγχρονες εφαρμογές

Επιπλέον Λειτουργικότητα

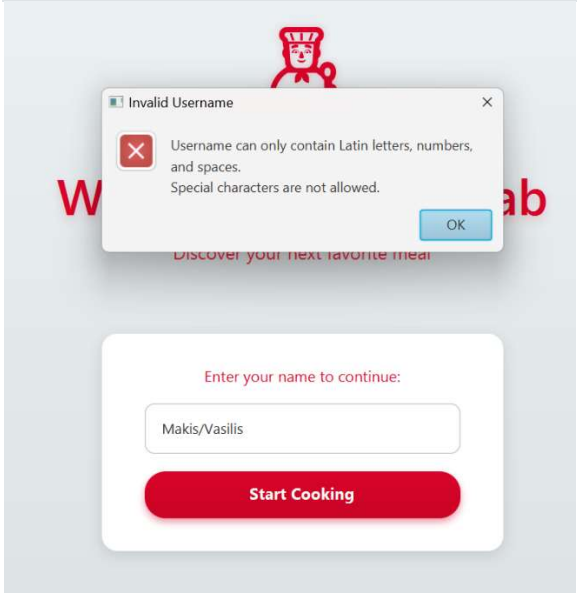
- Welcome Page μαζί με Login As a User: Η εφαρμογή ξεκινάει με μια εισαγωγική οθόνη (WelcomeController) που καλωσορίζει τον χρήστη πριν μεταβεί στην κύρια εφαρμογή.
- Διαχείριση Ιστορικού (Cooked History): Πέρα από την απλή προβολή, ο χρήστης έχει τη δυνατότητα να αναιρέσει μια ενέργεια ("Mark as Uncooked"), αφαιρώντας τη συνταγή από το ιστορικό μαγειρέματος.
- I'm Feeling Lucky: Λειτουργία που φέρνει μια εντελώς τυχαία συνταγή.
- Username Validation: Στο user login , ελέγχουμε αν το όνομα περιέχει χαρακτήρες όπως *, ?, /).

Επειδή το όνομα χρησιμοποιείται για τη δημιουργία αρχείου JSON (Favorites,Cooked), τέτοιοι χαρακτήρες θα προκαλούσαν σφάλμα κατά της αποθήκευση του στο σύστημα. Χρησιμοποιούμε Regular Expressions (Regex) για να επιτρέπουμε μόνο γράμματα και αριθμούς.

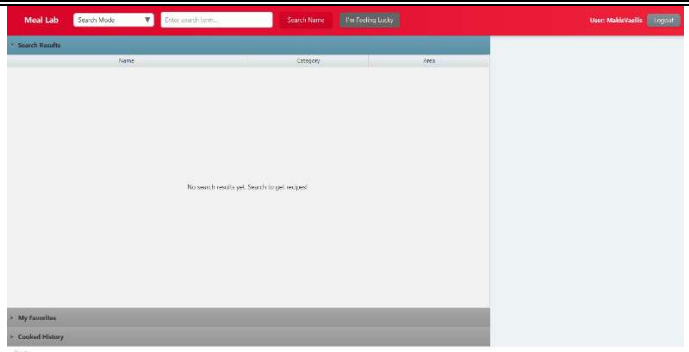
3. Εικόνες από την εφαρμογή

Αρχική οθόνη	

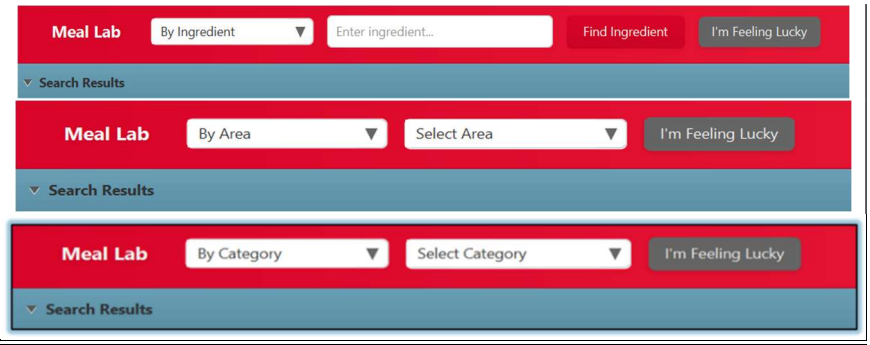
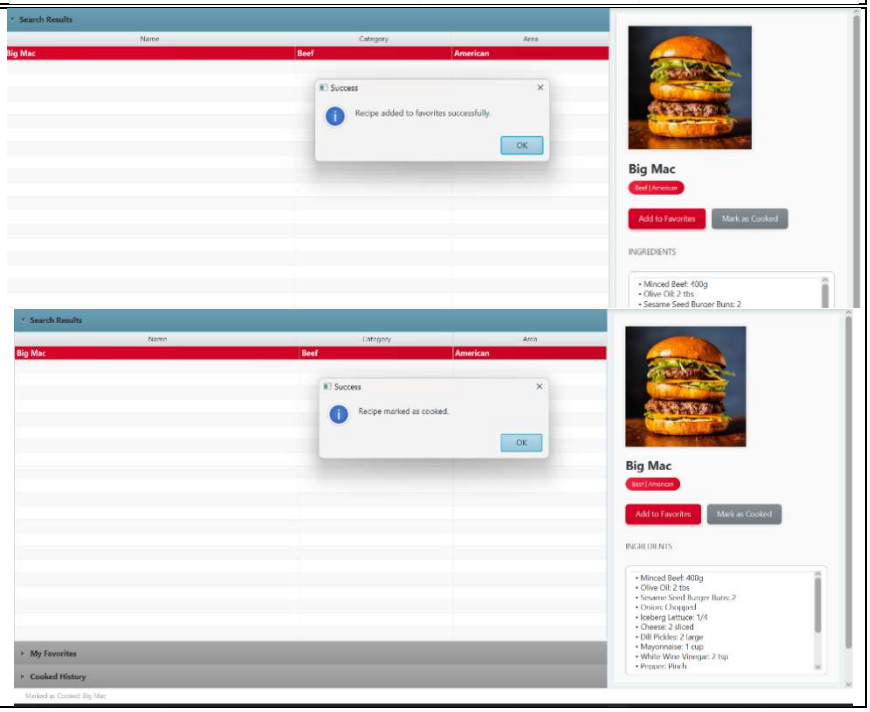
Login Warnings

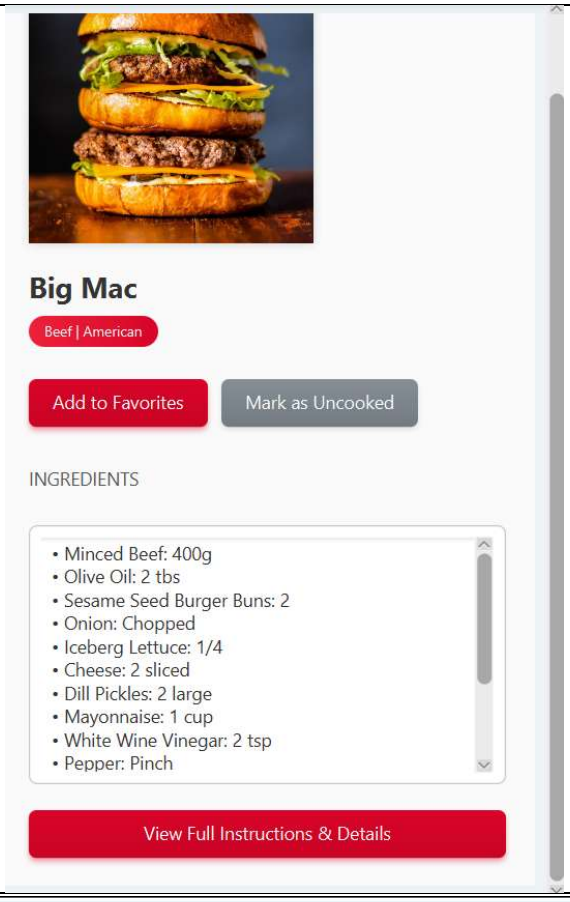
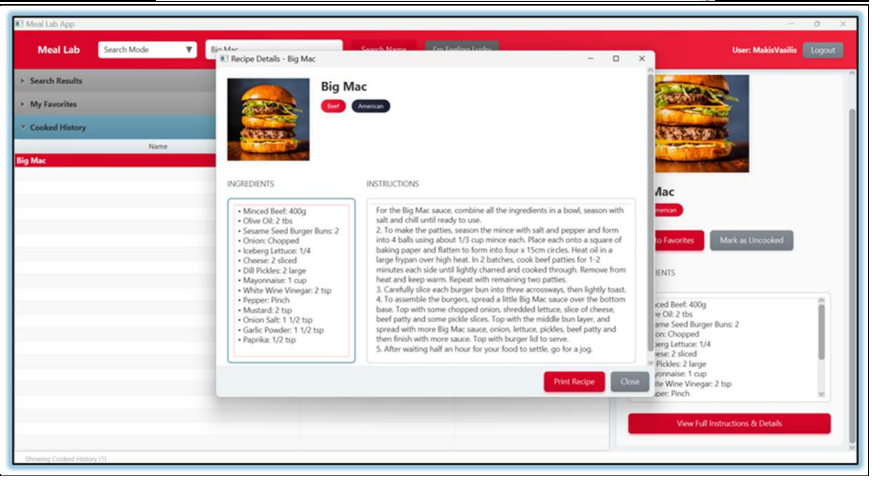


Main Screen



Μπάρες αναζήτησης και Filtering

	
Tab Favorites & Cooked	
Εισαγωγή Meal σε Favorites & Cooked	

Meal Side Panel	
Παράθυρο Συνταγής	

4. Ενδεικτικά αποτελέσματα των δοκιμών JUnit

Testing του API/Διενέργεια Smoke Testing

Πριν από την πλήρη ανάπτυξη της εφαρμογής, πραγματοποιήθηκε Smoke Testing μέσω της κλάσης Main για την επαλήθευση της σύνδεσης με το API. Αυτός ο αρχικός έλεγχος επέτρεψε την επιβεβαίωση των δικτυακών κλήσεων και του σωστού parsing των δεδομένων σε πραγματικό χρόνο. Αφού σταθεροποιήθηκε η επικοινωνία και υλοποιήθηκε η τελική βιβλιοθήκη δοκιμών (JUnit), ο κώδικας αυτός αφαιρέθηκε από την κλάση Main. Με αυτόν τον τρόπο διατηρήθηκε η καθαρότητα της αρχιτεκτονικής, μεταφέροντας όλη τη διαδικασία ελέγχου σε ένα απομονωμένο και αυτοματοποιημένο περιβάλλον testing.

Testing του API /J Units

1. IngredientTest.java

Διενεργήθηκε unit testing στην κλάση Ingredient για την επαλήθευση της ακεραιότητας των δεδομένων κατά την αρχικοποίηση και την τροποποίησή τους. Επιπλέον, ελέγχθηκε η μέθοδος toString() για να διασφαλιστεί η ορθή αναπαράσταση των δεδομένων στην επεξεργασία κειμένου.

2. MealResponseTest.java

Η συγκεκριμένη δοκιμή επικεντρώνεται στο Data Transfer Object (DTO) που διαχειρίζεται τις αποκρίσεις του API. Ελέγχθηκε η ικανότητα της κλάσης MealResponse να αποτυπώνει σωστά τις λίστες αντικειμένων Recipe, διασφαλίζοντας ότι το schema του παραμένει συνεκτικό κατά τη μεταφορά του.

3. RecipeTest.java

Αποτελεί τεστ για τη διαδικασία του Deserialization. Ελέγχει ότι ο ObjectMapper χειρίζεται σωστά τα null πεδία και τα boundary conditions των υλικών (1-20), αποτρέποντας runtime errors κατά το parsing.

4. MealApiConnectorTest.java

Πρόκειται για έλεγχο υποδομής που επιβεβαιώνει την ορθή παραμετροποίηση του Retrofit client. Επαληθεύει ότι το Singleton interface της επικοινωνίας με το API αρχικοποιείται σωστά και ότι τα dependecies για την εκτέλεση των δικτυακών κλήσεων είναι διαθέσιμα.

5. MealServiceTest.java

Πραγματοποιήθηκε έλεγχος του business logic. Η απομόνωση του Service από το πραγματικό δίκτυο επιτρέπει την προσομοίωση των edge cases όπως η αποτυχία σύνδεσης ή η λήψη κενών αποτελεσμάτων, διασφαλίζοντας τη σταθερότητα της εφαρμογής.

Έλεγχος Αποθήκευσης και Σταθερότητας Δεδομένων

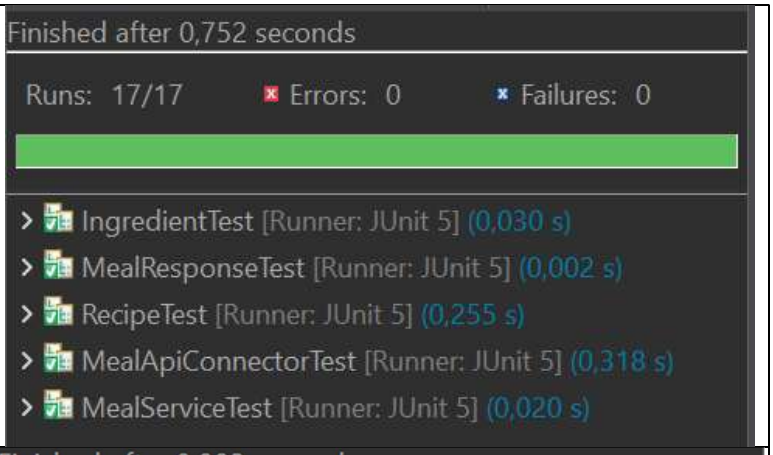
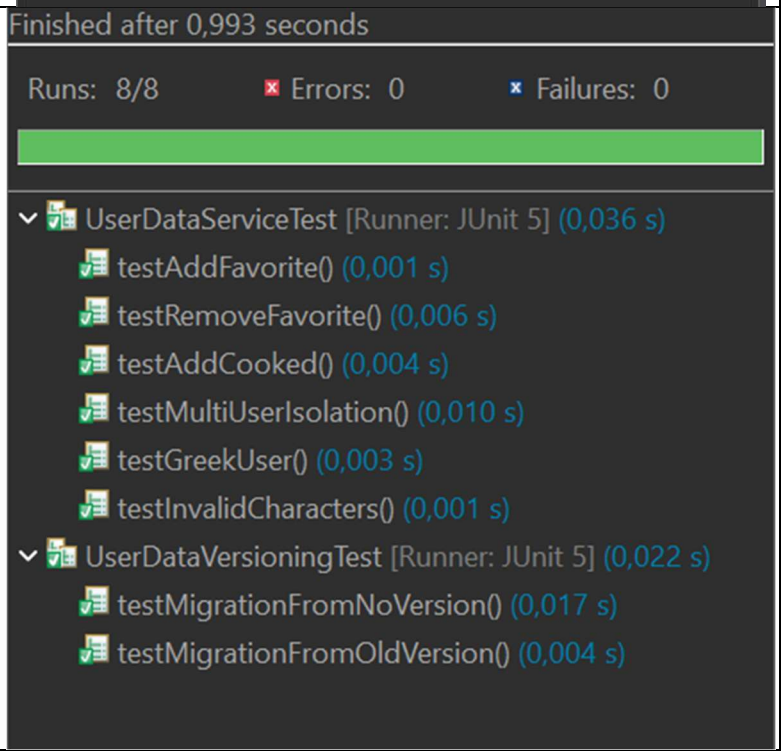
Στην ενότητα αυτή εξετάστηκε η αξιοπιστία του persistence layer και η διαχείριση των αρχείων JSON. Οι δοκιμές επιβεβαίωσαν τον πλήρη διαχωρισμό των δεδομένων ανά χρήστη και την ανθεκτικότητα του συστήματος στον καθαρισμό ονομάτων αρχείων (sanitization) από ειδικούς χαρακτήρες ή ελληνικά ονόματα, αποτρέποντας σφάλματα εγγραφής στο λειτουργικό σύστημα.

Επιπλέον, ενσωματώθηκε μηχανισμός Versioning για τη διασφάλιση της μελλοντικής συμβατότητας των δεδομένων. Μέσω των δοκιμών επαληθεύτηκε η ικανότητα της εφαρμογής να αναγνωρίζει παλαιότερες δομές αρχείων και να εκτελεί αυτόματα Data Migration στη νέα έκδοση. Η προσέγγιση αυτή εγγυάται την ακεραιότητα των αποθηκευμένων συνταγών και την ομαλή εξέλιξη της εφαρμογής σε μελλοντικές αναβαθμίσεις.

Debugging

Κατά τη φάση του debugging εντοπίστηκε ότι οι δοκιμές (JUnits) αλληλεπιδρούσαν με το κύριο αρχείο δεδομένων της εφαρμογής (user_data_*.json), προκαλώντας αλλοίωση στα πραγματικά δεδομένα του χρήστη. Για την απομόνωση των δοκιμών από το περιβάλλον παραγωγής, υλοποιήθηκε η μέθοδος setFileNameForTesting() στην κλάση UserDataService.

Με την προσέγγιση αυτή, τα JUnits ορίστηκαν να χρησιμοποιούν αποκλειστικά το προσωρινό αρχείο test_user_data.json. Παράλληλα, με τη χρήση των annotations @BeforeAll και @AfterAll, διασφαλίστηκε ότι το αρχείο δοκιμών δημιουργείται και διαγράφεται αυτόματα σε κάθε εκτέλεση. Η λύση αυτή εξασφάλισε την ακεραιότητα των δεδομένων του χρήστη και την καθαρότητα του περιβάλλοντος εκτέλεσης.

JUnits για το API	
JUnits για το UserService	

5. Συνεισφορά Μελών

Στο πλαίσιο της υλοποίησης της εφαρμογής, οι εργασίες μοιράστηκαν ανάμεσα στα μέλη της ομάδας με στόχο την αποτελεσματική κάλυψη όλων των τεχνικών απαιτήσεων της εργάσιας.

Ο Γεράσιμος Πλατής επικεντρώθηκε στην αρχιτεκτονική του backend και των δοκιμών testing. Συγκεκριμένα, ανέλαβε την πλήρη διαχείριση της επικοινωνίας με το API (Retrofit, @JsonAnySetter) και τη μοντελοποίηση των δεδομένων (POJO), ενώ υλοποίησε τους ελέγχους JUnit. Παράλληλα, είχε την αποκλειστική ευθύνη για τον συντονισμό και τη διαχείριση του κώδικα μέσω Git.

Ο Βασίλης Χαχούλης εστίασε στον σχεδιασμό της διεπαφής και τη διαχείριση τοπικών δεδομένων. Επιμελήθηκε το εικαστικό και λειτουργικό μέρος της εφαρμογής (wireframes, CSS, FXML), υλοποιώντας τη δυναμική σελίδα συνταγών, το σύστημα Login και τη λειτουργία εκτύπωσης. Τέλος, ανέπτυξε την υπηρεσία UserDataService για την αποθήκευση των δεδομένων της εφαρμογής.

6. Εργαλεία Ανάπτυξης

- **Java** (JDK 17+)
- **JavaFX**
- **Maven**
- **Git**
- **IDE**: VS Code & Eclipse
- **Postman**

7. Σύνδεσμος Αποθετηρίου (Git Repository)

https://github.com/makisplts1995/meallabapp