**Backend**

Ανέλαβα να γράψω το backend της εφαρμογής το οποίο ήταν σχετικά απλό. Αποτελείται από το μοντέλο (κλάση) ενός «πιάτου» που προσφέρει το κατάστημα. Έχει πληροφορίες σχετικά με στοιχεία που προωθεί η εταιρία όπως το όνομα του πιάτου, την τιμή του πιάτου, την περιγραφή του πιάτου και εικόνα του πιάτου, αλλά και πληροφορίες metadata που χρησιμοποιεί η εφαρμογή για την πραγματοποίηση μιας παραγγελίας όπως την ποσότητα (Πόσα πιάτα θέλει ο πελάτης να πάρει). Ένα ερώτημα ήταν αν στην java μπορώ να περάσω by reference την λίστα για άμεση τροποποίηση. Η απάντηση που δόθηκε από μία αναζήτηση στο ιντερνετ είναι όχι και για αυτόν τον λόγο η λύση που σκέφτηκα ήταν να δημιουργήσω μία στατική λίστα καλαθιού μέσα στην δομή του «πιάτου»

Note: Τα σχόλια για την παραγγελία (π.χ. πόσο ψημένο θα είναι το κρέας, ή κάποια συγκεκριμένη παράκληση κατά την παράδοση) αποφασίστηκε να μην αποθηκεύονται σε δομή στο backend αλλά να στέλνονται κατευθείαν από το front end στο χαρτί παραγγελίας.

Στο backend της εφαρμογής συνδέεται επίσης μία βάση δεδομένων Sqlite μέσω της κλάσης DatabaseHandler που είναι πρακτικά μία wrapper κλάση για την διαχείριση της βάσης δεδομένων. Η βάση δεδομένων περιέχει τα πιάτα που προσφέρει το κατάστημα με όλα τα metadata τους και ενημερώνεται σε συχνή βάση από το κατάστημα (Θεωρητικά).

**Αρχική οθόνη**

Επίσης ανέλαβα να γράψω την αρχική οθόνη η οποία είναι και η πρώτη που εμφανίζεται στον χρήστη που ξεκινάει την εφαρμογή μας. Η οθόνη έχει ένα πλήρως custom design όπου τα χρώματα είναι επιλεγμένα σε σκούρο theme και τα γράμματα ανοιχτά. Η οθόνη περιέχει αρκετά γραφικά αντικείμενα, μερικά εκ των οποίων είναι custom που δημιουργήσαμε. Πιο συγκεκριμένα περιέχει:

* ListView Control
* Custom ListViewCell (ListViewItems)
* Button
* Labels
* Spinner
* Dialog
* Ένα custom κουμπί το οποίο αποτελείται από ένα HBox και σαν σώμα περιέχει ένα ImageView και ένα Label.
* ImageView (Logo)

Πέρα από τα γραφικά αντικείμενα η οθόνη αυτή περιέχει επίσης 4 animations.

* Ένα animation είναι στο logo όπου κάνει ένα rotate 180 μοιρών στον άξονα X κάθε φορά που ο χρήστης περνάει τον κέρσορα από πάνω του
* Ένα animation είναι στο brand name label το οποίο κάνει rotate αρνητικές 30 μοίρες στον άξονα Z κάθε φορά που ο χρήστης περνάει τον κέρσορα από πάνω του.
* Και τέλος τα τελευταία 2 animations βρίσκονται στα κουμπιά View Your Cart και Ολοκλήρωση παραγγελίας τα οποία κάνουν την ίδια λειτουργία. Όποτε ο χρήστης περνάει τον κέρσορα από πάνω τους αυτά μεγαλώνουν λίγο και αλλάζουν το χρώμα τους προς μια πιο σκούρα απόχρωση.

Για να εμβαθύνω λίγο στον τρόπο λειτουργίας της εφαρμογής θα φέρω ένα παράδειγμα χρήσης. Μπαίνω σαν χρήστης στην εφαρμογή για να παραγγείλω. Ανοίγω την εφαρμογή, και βλέπω την αρχική οθόνη. Κατευθείαν έχω φορτωμένο στην λίστα μου όλο το μενού του μαγαζιού στην λίστα, με φωτογραφίες, περιγραφές και τιμές. Μπορώ να επιλέξω κάποιο πιάτο πατώντας επάνω του. Μόλις πατήσω πάνω του κάτω χαμηλά ενεργοποιείται το μενού επιλογής ποσότητας το οποίο μπορώ να τροποποιήσω για να προσθέσω περισσότερα από ένα πιάτα της ίδιας κατηγορίας. Μπορώ να τροποποιήσω την επιλογή μου μέσω του διαθέσιμου Spinner. Αυτόματα ενημερώνεται για κάθε επιπλέον πιάτο που προσθέτω στην παραγγελία μου το τελικό ποσό για αυτό το πιάτο. Μόλις πατήσω το κουμπί «Add To Cart» το/τα πιάτο/α μου εισάγονται στο καλάθι μου και εμφανίζεται ένα μήνυμα επιτυχίας το οποίο περιέχει ενημερωτικά το καλάθι μου με τις τιμές, τις ποσότητες και την τελική τιμή της παραγγελίας μου. Αν θέλω να τροποποιήσω την παραγγελία μου μπορώ να πατήσω το κουμπί «View Your Cart» το οποίο μου εμφανίζει μία οθόνη με το καλάθι μου στο οποίο μπορώ να αφαιρέσω ή να τροποποιήσω την παραγγελία μου. Μόλις κατασταλάξω σε αυτά που θέλω, γυρνάω στην αρχική οθόνη και πατάω ολοκλήρωση παραγγελίας. Εμφανίζεται ένα νέο παράθυρο το οποίο περιέχει την τελική μου παραγγελία για επιβεβαίωση, τελική τιμή, ένα πλαίσιο κειμένου για τυχών σχόλια στην παραγγελία μου και τους τρόπους πληρωμής. Μπορώ να επιλέξω πληρωμή με μετρητά ή με χρεωστική/πιστωτική κάρτα. Στην περίπτωση μετρητών η παραγγελία ολοκληρώνεται άμεσα χωρίς περαιτέρω καθυστερήσεις. Στην περίπτωση κάρτας αλλάζει η οθόνη, σε μία οθόνη εισαγωγής των στοιχείων της κάρτας για ολοκλήρωση της παραγγελίας.

**Custom ListViewCells**

Τέλος ανέλαβα και έγραψα το ListViewCell το οποίο είναι πλήρως custom και έχουμε δει στο μάθημα (Τουλάχιστον στις 16 Απριλίου που γράφω αυτό το κείμενο). Υπήρξαν πάρα πολλά προβλήματα κατά την δημιουργία τους και μια μεγάλη προγραμματιστική παρέμβαση. Απαιτήθηκε πολύ αναζήτηση στο ιντερνετ για να μπορέσει να ξεπεραστεί το πρόβλημα με το πώς λειτουργεί το ListViewCell και πώς να δουλέψει ο controller. Να σημειωθεί πως δεν υπήρξε καθόλου copy-paste κώδικα από το ιντερνετ σε αυτή τη φάση της εφαρμογής επειδή πολύ απλά δεν λειτουργούσε τίποτα από αυτά που υπήρχαν. Μετά από πολύ trial and error κατάφερα να το κάνω να λειτουργήσει αλλά με το layout να δημιουργείται μέσω κώδικα. Δεν κατάφερα ποτέ να ξεπεράσω το πρόβλημα με τα fxml αρχεία όπου για κάποιο λόγο δεν δεχόταν τον controller. Ασχέτως με τα τεχνικά προβλήματα αυτής της φάσης στο τέλος δούλεψε το γραφικό στοιχείο μου και χρησιμοποιήθηκε στην αρχική οθόνη την οποία έφτιαξα και στην οθόνη εμφάνισης του καλαθιού του χρήστη.

Για να κατανοήσουμε τον τρόπο λειτουργίας είναι αρκετό να αναδειχτούν αυτές οι συγκεκριμένες γραμμές κώδικα. Αυτό που κάνουμε εδώ είναι να δημιουργήσουμε μία κλάση MainListViewItemCell η οποία είναι το κάθε ένα ξεχωριστό item της λίστας μας. Φτιάχνουμε μέσω κώδικα κάθε κομμάτι του item ξεχωριστά όταν καλούμε τον constructor. Αυτή η κλάση όπως φαίνεται κληρονομεί από την κλάση ListCell με τύπο δεδομένων Plate (Αυτό ονομάζεται Generic Class και λειτουργεί για οτιδήποτε τύπο και να δώσουμε πράγμα που το κάνει πολύ ισχυρό και ευέλικτο για κάθε πιθανή χρήση) .



Από την στιγμή που κληρονομούμε την κλάση ListCell μας δίνεται η δυνατότητα να κάνουμε Override την μέθοδο που περιέχει, και καλεί κάθε φορά που δημιουργείται το item μας, updateItem. Αυτή μας δίνει τις πληροφορίες για το τρέχων «πιάτο» από την λίστα και ένα Boolean που μας λέει αν υπάρχουν δεδομένα για να εμφανιστούν σε αυτό το κελί. Μπορεί το τρέχων κελί να χρησιμοποιείται σαν κενό (Αυτά τα λέει στο documentation της oracle). Αξίζει να σημειωθεί πως αυτή η μέθοδος καλείται αυτόματα από την δομή η οποία χειρίζεται τα items. Στην περίπτωση μας αυτή η δομή είναι η ListView. Επομένως με βάση όλα αυτά που είπαμε, αν έχουμε δεδομένα για να δείξουμε στο τρέχων κελί καλούμε την addContext μέθοδο για να τροποποιήσει τα ανάλογα γραφικά στοιχεία στο κελί μας.



Όμως το πραγματικό πρόβλημα δεν ήταν εδώ. Το δύσκολο είναι η επικοινωνία και δημιουργία των στοιχείων αυτών. Το setup της λίστας και των κελιών της φαίνεται στο παρακάτω screenshot:



Αυτό το κομμάτι κώδικα παρότι μικρό ήταν αρκετά περίπλοκο την ώρα της αναζήτησης. Καταρχάς στην δομή ListView φτιάχνουμε ένα CellFactory το οποίο πρακτικά είναι ο τρόπος με τον οποίο θα φτιάχνει τα κελιά για κάθε στοιχείο της λίστας. (Ή τουλάχιστον αυτό κατάλαβα εγώ). Η αμέσως επόμενη γραμμή βάζει στην δομή την λίστα με τα πιάτα που έχει το κατάστημα. Αυτές οι δύο γραμμές κώδικα σε συνδυασμό μαζί κάνουν το όλο σύστημα να δουλεύει, καθώς για κάθε αντιείμενο της λίστας καλεί το CellFactory το οποίο περιγράφει πως θα δημιουργηθεί το κελί, και μετά η ListView εσωτερικά καλεί την μέθοδο updateItem(Plate plate, Boolean empty) η οποία τροποποιεί τα γραφικά στοιχεία του κελιού για να γεμίσει με τις πληροφορίες του συγκεκριμένου «πιάτου»

**Πηγές αναζήτησης και βοήθειας στο ιντερνετ:**

JavaFX HBox Alignment: [https://stackoverflow.com/questions/29707882/javafx-hbox-alignment](https://stackoverflow.com/questions/29707882/javafx-hbox-alignment%20)

(Για το Backend): [https://stackoverflow.com/questions/40480/is-java-pass-by-reference-or-pass-by-value](https://stackoverflow.com/questions/40480/is-java-pass-by-reference-or-pass-by-value%20)

Customize list view (Populate Fields): [https://stackoverflow.com/questions/19588029/customize-listview-in-javafx-with-fxml](https://stackoverflow.com/questions/19588029/customize-listview-in-javafx-with-fxml%20)

Custom List View (Video): [https://www.youtube.com/watch?v=4rVr\_VT-4Z4](https://www.youtube.com/watch?v=4rVr_VT-4Z4%20)

Show image Programmatically: [https://stackoverflow.com/questions/22710053/how-can-i-show-an-image-using-the-imageview-component-in-javafx-and-fxml](https://stackoverflow.com/questions/22710053/how-can-i-show-an-image-using-the-imageview-component-in-javafx-and-fxml%20)

Dialog in javafx: [https://www.tutorialspoint.com/how-to-create-a-dialog-in-javafx](https://www.tutorialspoint.com/how-to-create-a-dialog-in-javafx%20)

Dialog in javafx: [https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Dialog.html](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Dialog.html%20)

Connect a listener to a spinner : [https://stackoverflow.com/questions/29926428/how-do-you-attach-a-listener-to-a-javafx-spinner](https://stackoverflow.com/questions/29926428/how-do-you-attach-a-listener-to-a-javafx-spinner%20)

rotate animations: [https://www.javatpoint.com/javafx-rotate-transition](https://www.javatpoint.com/javafx-rotate-transition%20)

Oracle Documentation over ListCell: [https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Cell.html#updateItem-T-boolean-](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Cell.html%23updateItem-T-boolean-)

Από εδώ τροποποιήθηκε το εξής μέρος κώδικα και χρησιμοποιήθηκε:

