Τμήμα Μηχανικών Πληροφορικής Τ.Ε

ΝΟΜΙΚΟΣ ΔΗΜΗΤΡΙΟΣ Α.Μ 2007134

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Με θέμα:

Ταυτόχρονη Ανάπτυξη Πολυεπίπεδης Εφαρμογής για Πολλαπλές φορητές συσκευές

Επιβλέπων Καθηγητής: Νίκος Πανάγος

Περιεχόμενα

1 ABSTRACT

Σε αυτή την εργασία θα δούμε την υλοποίηση μιας πολυεπίπεδης εφαρμογής για πολλαπλές φορητές πλατφόρμες.

Στο πρώτο κεφάλαιο θα περιγράψουμε το πρόβλημα το οποίο χρήζει επίλυση και το πώς θα προχωρήσουμε σε αυτή. Επίσης αναφέρονται οι λόγοι οι οποίοι οδήγησαν στην επιλογή των συγκεκριμένων τεχνολογιών. Τέλος θα αναλύσουμε τις ανάγκες τις οποίες θα πρέπει να καλύπτει η εφαρμογή η οποία θα δημιουργηθεί.

Στο δεύτερο κεφάλαιο θα γίνει περιγραφή των τεχνολογιών οι οποίες θα χρησιμοποιηθούν καθώς και οι λόγοι προτίμησης κάποιων συγκεκριμένων. Στην συνέχεια παρατίθενται πληροφορίες για το πως πρέπει να είναι το σύνηθες γραφικό περιβάλλον κάθε πλατφόρμας το οποίο θα ακολουθήσουμε πιστά και εμείς. Επίσης θα γίνει ανάλυση της Αρχιτεκτονικής της Εφαρμογής μας.

Στο τρίτο κεφάλαιο θα δούμε το πως έγινε η κατανομή του χρόνου που αφιερώθηκε για την συνολική ανάπτυξη του όλου project, τον τρόπο τον οποίο έγινε το τεστ της εφαρμογής στις πλατφόρμες για τις οποίες αναπτύχθηκε και τέλος θα αναφέρουμε διάφορες βελτιστοποιήσεις οι οποίες θα μπορούσαν να κάνουν την εφαρμογή καλύτερη και θα υλοποιηθούν στο μέλλον.

2 Πρόλογος

Στις μέρες μας οι φορητές συσκευές οι οποίες κατέχουν οι άνθρωποι πολλαπλασιάζονται με τεράστιους ρυθμούς. Όλοι οι άνθρωποι θέλουν εύκολη πρόσβαση στην πληροφορία η οποία πλέον συνήθως ανανεώνεται με ρυθμούς γρηγορότερους από αυτούς που μπορούμε να παρακολουθήσουμε. Οι φορητές συσκευές μας δίνουν πρόσβαση στην πληροφορία οπουδήποτε και αν είμαστε, πράγμα το οποίο τις καθιστά απαραίτητες ειδικά αν οι αποφάσεις μας κρίνονται από αυτές.

Αυτό έχει ως αποτέλεσμα να υπάρχει έντονη ζήτηση για λογισμικό το οποίο θα φέρει εύκολα και γρήγορα την πληροφορία στον πελάτη. Η δημιουργία των λογισμικών αυτών όμως είναι μια διαδικασία η οποία έχει σπάσει σε πολλά μονοπάτια λόγω των διαφορετικότητας του λειτουργικού των κινητών.

Καθώς αρκετές από τις συσκευές αυτές είναι διαφορετικές ως προς τον τρόπο που είναι φτιαγμένο το λειτουργικό τους σύστημα, αυτό έχει ως απαίτηση μια εφαρμογή ώστε να είναι διαθέσιμη σε μεγάλο κοινό, να δημιουργείται από την αρχή ξεχωριστά από την αρχή για την κάθε πλατφόρμα.

Στόχος της εργασίας είναι να δημιουργήσουμε μια εφαρμογή για φορητές συσκευές Android & iOS ταυτόχρονα η οποία θα μας δίνει την δυνατότητα για λήψη και απεικόνιση στατιστικών οικονομικών πληροφοριών και στοιχείων σύμφωνα με τις κινήσεις που συμβαίνουν μέσα από το ERP μιας επιχείρησης.

3 Περιγραφή Προβλήματος

Η εταιρεία XXXXXX έχει αναπτύξει ένα εμπορικό ERP σε Delphi. Το ERP αυτό προσφέρει την δυνατότητα διαχείρισης πελατών και αποθήκης καθώς και τον έλεγχο πωλήσεων ενός καταστήματος. Επίσης υπάρχει η δυνατότητα σύνδεσης του ERP σε online Βάση Δεδομένων έτσι ώστε όλοι οι χρήστες μιας ομάδας να ενημερώνουν την ίδια. Η εταιρεία αυτή επίσης δημιουργεί e-shops τα οποία μπορούν οι πελάτες να τα διαχειριστούν έμμεσα από το ERP αυτό έχοντας την δυνατότητα να επηρεάσουν την διαθεσιμότητα των προϊόντων. Τέλος δίνει στους πελάτες την δυνατότητα να βλέπουν οικονομικά στατιστικά για την πορεία της επιχείρησής τους σύμφωνα με τις κινήσεις οι οποίες καταγράφονται μέσα σε αυτό.

Με τον καιρό δημιουργήθηκε η ανάγκη να υπάρξει ένα μέσο παρακολούθησης αυτών των στατιστικών με κύριο προσόν την φορητότητα και να μπορούν να ενημερώνονται ανά πάσα στιγμή. Οι χρήστες που επιλέγουν η βάση δεδομένων τους να βρίσκεται σε Cloud είναι δυνατόν να έχουν ένα τέτοιο μέσο. Σύμφωνα με τον τρόπο που εξελίσσεται η τεχνολογία στις μέρες μας οι κινητές συσκευές θα ήταν αυτό που έρχεται κατευθείαν στο μυαλό μας ως κατάλληλο έδαφος για την ανάπτυξη μιας εφαρμογής τέτοιου τύπου.

Δημιουργώντας μια εφαρμογή για κινητές συσκευές όμως πρέπει να έχουμε στο νου μας το εύρος των λειτουργικών συστημάτων που υπάρχουν στην αγορά και να προσπαθήσουμε να καλύψουμε το μεγαλύτερο μέρος της αγοράς με όσο λιγότερο κόστος γίνεται. Σύμφωνα με τους πελάτες μας υπήρχε ζήτηση της εφαρμογής κυρίως για λειτουργικά iOS και Android.

Η εταιρεία έπρεπε να βρει έναν οικονομικό τρόπο για την υλοποίηση του έργου εφόσον δεν υπήρχαν άτομα τα οποία να ασχολούνται με την δημιουργία εφαρμογών για φορητές συσκευές χωρίς να επεκταθεί υπερβολικά εφόσον δεν ήταν αυτός ο κύριος στόχος της. Μετά από έρευνα ανακαλύψαμε πως η εταιρεία Embarcadero η οποία φροντίζει την ανάπτυξη της Delphi, στις τελευταίες της εκδόσεις δημιούργησε ένα Framework το οποίο δίνει τη δυνατότητα ανάπτυξης εφαρμογών. Κινούμενοι σε αυτή την τεχνολογία που είναι ήδη γνωστή θα ήταν πολύ πιο οικονομικό.

4 Τεχνικές προδιαγραφές

Η υλοποίηση του έργου χρήζει την δημιουργία μιας εφαρμογής η οποία θα τραβάει δεδομένα από Βάση Δεδομένων Microsoft SQL και θα τα απεικονίζει σε πλατφόρμες Android και iOS. Εφόσον χρειάζεται συνεχής επικοινωνία με εξωτερική βάση δεδομένων τότε η εφαρμογή μας θα είναι πολυεπίπεδη και θα προϋποθέτει σύνδεση στο internet.

Το ένα κομμάτι της θα είναι η βάση δεδομένων την οποία θα χρησιμοποιεί το ERP μας στο οποίο θα δημιουργήσουμε Stored Procedures που θα αξιοποιήσουμε στην εφαρμογή μας για να πάρουμε τα στατιστικά που θέλουμε.

Το δεύτερο κομμάτι θα είναι ένας Server ο οποίος ελέγχει τις συνδέσεις των χρηστών και θα πραγματοποιεί την επαλήθευσή χρήστη και την σύνδεση με την εκάστοτε Βάση Δεδομένων. Θα εκθέτει μεθόδους οι οποίες θα αναλαμβάνουν το έργο του να συλλέγουν τα ζητούμενα δεδομένα, θα τα πακετάρει με τρόπο κατάλληλο και τέλος θα τα αποστέλλει στο τρίτο κομμάτι της εφαρμογής μας.

Το τρίτο κομμάτι θα είναι η εφαρμογή την οποία θα έχουν οι χρήστες στο κινητό τους και μέσα από το γραφικό περιβάλλον της εφαρμογής θα μπορούν να επιλέξουν τα στοιχεία για τα οποία θα ήθελαν να ενημερωθούν. Αφού ο χρήστης κάνει την επιλογή του, τα στοιχεία θα εμφανίζονται με φιλικό τρόπο για να λάβει τις πληροφορίες που θέλει.

Το τρίτο κομμάτι της εφαρμογής θα καλύπτει πιο συγκεκριμένα τις παρακάτω ανάγκες.

Θα δίνει δυνατότητα αίτησης δημιουργίας λογαριασμού μέσω mail.

Εφόσον ο λογαριασμός του χρήστη δημιουργηθεί, θα μπορεί να συνδεθεί στην εφαρμογή και να δει το κύριο μενού στο οποίο βρίσκονται όλες οι δυνατότητες.

Θα υπάρχει δυνατότητα ρύθμισης του εύρους ημερομηνιών για τις οποίες ο χρήστης ζητάει να φέρει στατιστικά. Αυτό θα υπάρχει σε δύο σημεία. Το ένα σημείο θα είναι στο κύριο μενού της εφαρμογής. Το δεύτερο σημείο θα είναι σε κάποιο υποπαράθυρο κάθε φόρμας το οποίο θα ανοίγει σαν συρτάρι και θα μπορεί να επιλέγει ο χρήστης προεπιλεγμένα όρια ημερομηνιών.

Τα προεπιλεγμένα όρια ημερομηνιών θα είναι τα εξής.

- **Σήμερα**, οι ημερομηνίες «από» και «έως» ρυθμίζονται στην σημερινή ημερομηνία.

- **Χτες**, οι ημερομηνίες «από» και «έως» ρυθμίζονται στην χθεσινή ημερομηνία.

- **Αρχή της Εβδομάδας**, η ημερομηνία «από» ρυθμίζεται στην πιο κοντινή Δευτέρα του ημερολογίου ενώ η ημερομηνία «έως» ρυθμίζεται στην σημερινή.

- **1 Εβδομάδα Πριν**, η ημερομηνία «από» ρυθμίζεται στην ίδια μέρα της, σημερινής μία εβδομάδα πριν ενώ η ημερομηνία «έως» ρυθμίζεται στην σημερινή.

- **1 Μήνα Πριν**, η ημερομηνία «από» ρυθμίζεται στην ένα μήνα πριν από την σημερινή ημερομηνία ενώ η ημερομηνία «έως» ρυθμίζεται στην σημερινή.

- **Αρχή του Μήνα**, η ημερομηνία «από» ρυθμίζεται στην 1^{η} του τρέχοντος μήνα ενώ η ημερομηνία «εως» ρυθμίζεται στην σημερινή.

Στο κύριο μενού θα υπάρχουν οι εξής δυνατότητες για τον χρήστη.

- Προβολή Τζίρου Πελατών. Με αυτή την επιλογή ο χρήστης μπορεί να δεί το κέρδος που έχει από κάθε κατάστημα με βάση τις πωλήσεις σε πελάτες. Τα στοιχεία που θα προβάλλονται θα περιέχουν το κέρδος με ΦΠΑ και χωρίς ΦΠΑ. Στο τέλος της λίστας επίσης θα προβάλλεται το συνολικό κέρδος από όλα τα καταστήματα του χρήστη με ΦΠΑ και χωρίς ΦΠΑ.

- Προβολή Ταμείου Μετρητών. Με αυτή την επιλογή ο χρήστης μπορεί να δει τα μετρητά που θα έπρεπε να βρίσκονται σε κάθε ταμείο καταστήματος από αγορές πελατών που πραγματοποιήθηκαν με μετρητά. Τα στοιχεία που θα προβάλλονται θα περιέχουν το κέρδος με ΦΠΑ και χωρίς ΦΠΑ. Στο τέλος της λίστας επίσης θα προβάλλεται το συνολικό κέρδος από όλα τα καταστήματα του χρήστη με ΦΠΑ και χωρίς ΦΠΑ.

- Προβολή Ταμείου Projected. Με αυτή την επιλογή ο χρήστης μπορεί να δει τα χρήματα που υπάρχουν στο εικονικό ταμείο κάθε καταστήματος από αγορές πελατών που πραγματοποιήθηκαν με πιστωτική κάρτα.

- Πωλήσεις ανά Κατηγορία. Με αυτή την επιλογή ο χρήστης μπορεί να δει το κέρδος κάθε καταστήματος αθροίζοντας πωλήσεις με βάση την κατηγορία των προϊόντων που έχουν πωληθεί. Ο χρήστης θα οδηγείται πρώτα σε μια φόρμα Φίλτρων όπου θα επιλέγει αν θελήσει να δει αποτελέσματα κάποιου συγκεκριμένου καταστήματος ή και κάποιας συγκεκριμένης κατηγορίας προϊόντος. Σε κάθε κατηγορία που προβάλλεται ο χρήστης μπορεί πιο ειδικά να δει τα παρακάτω στοιχεία ανά Κατάστημα και ανά Κατηγορία:

    - Την **Ποσότητα** των προϊόντων κάθε κατηγορίας που πωλήθηκαν ανά κατάστημα.
    - Την Τελική Αξία των πωληθέντων προϊόντων κάθε κατηγορίας ανά κατάστημα, δηλαδή την Αξία των πωλήσεων συμπεριλαμβανομένου τον ΦΠΑ.
    - Την Καθαρή Αξία των πωληθέντων προϊόντων κάθε κατηγορίας ανά κατάστημα, δηλαδή την Αξία των πωλήσεων χωρίς τον ΦΠΑ.
    - Το Κόστος των πωληθέντων προϊόντων κάθε κατηγορίας, δηλαδή την Αξία του κόστους που είχε το κατάστημα για την αγορά των προϊόντων τις κάθε κατηγορίας.

Τέλος μπορεί να δει την Συνολική Τελική αξία κάθε καταστήματος μέσω των πωλήσεων ανά κατηγορία που προβάλλονται.

- Πωλήσεις ανά Κατασκευαστή. Με αυτή την επιλογή ο χρήστης μπορεί να δει το κέρδος κάθε καταστήματος αθροίζοντας πωλήσεις με βάση τον Κατασκευαστή των προϊόντων που έχουν πωληθεί. Ο χρήστης θα οδηγείται πρώτα σε μια φόρμα Φίλτρων όπου θα επιλέγει αν θελήσει να δει αποτελέσματα κάποιου συγκεκριμένου καταστήματος ή και κάποιου συγκεκριμένου Κατασκευαστή προϊόντος. Σε κάθε Κατασκευαστή που προβάλλεται ο χρήστης μπορεί πιο ειδικά να δει τα παρακάτω στοιχεία ανά Κατάστημα και ανά Επωνυμία Κατασκευαστή:

- Την **Ποσότητα** των προϊόντων κάθε προμηθευτή που πωλήθηκαν ανά Κατάστημα.
- Την Καθαρή Αξία των πωληθέντων προϊόντων κάθε προμηθευτή ανά κατάστημα, δηλαδή την Αξία των πωλήσεων χωρίς τον ΦΠΑ.
- Το Κόστος των πωληθέντων προϊόντων κάθε Προμηθευτή, δηλαδή την Αξία του κόστους που είχε το κατάστημα για την αγορά των προϊόντων του κάθε προμηθευτή.

- Πωλήσεις ανά Προμηθευτή. Με αυτή την επιλογή ο χρήστης μπορεί να δει το κέρδος κάθε καταστήματος αθροίζοντας πωλήσεις με βάση τον Προμηθευτή των προϊόντων που έχουν πωληθεί. Ο χρήστης θα οδηγείται πρώτα σε μια φόρμα Φίλτρων όπου θα επιλέγει αν θελήσει να δει αποτελέσματα κάποιου συγκεκριμένου καταστήματος ή και κάποιου συγκεκριμένου Προμηθευτή προϊόντος. Σε κάθε προμηθευτή που προβάλλεται ο χρήστης μπορεί πιο ειδικά να δει τα παρακάτω στοιχεία ανά Κατάστημα και ανά Επωνυμία Προμηθευτή:

    - Την **Ποσότητα** των προϊόντων κάθε προμηθευτή που πωλήθηκαν ανά Κατάστημα.
    - Την Καθαρή Αξία των πωληθέντων προϊόντων κάθε προμηθευτή ανά κατάστημα, δηλαδή την Αξία των πωλήσεων χωρίς τον ΦΠΑ.
    - Το Κόστος των πωληθέντων προϊόντων κάθε Προμηθευτή, δηλαδή την Αξία του κόστους που είχε το κατάστημα για την αγορά των προϊόντων του κάθε προμηθευτή.

Τέλος μπορεί να δει την Συνολική Τελική αξία κάθε καταστήματος μέσω των πωλήσεων ανα κατηγορία που προβάλλονται.

- Πωλήσεις ανά Brand. Με αυτή την επιλογή ο χρήστης μπορεί να δει το κέρδος κάθε καταστήματος αθροίζοντας πωλήσεις με βάση την μάρκα των προϊόντων που έχουν πωληθεί και μετά ανά κατηγορία. Ο χρήστης θα οδηγείται πρώτα σε μια φόρμα Φίλτρων όπου θα επιλέγει αν θελήσει να δει αποτελέσματα κάποιου συγκεκριμένου καταστήματος ή και κάποιας συγκεκριμένης μάρκας προϊόντος. Σε κάθε μάρκα που προβάλεται ο χρήστης μπορεί πιο ειδικά να δει τα παρακάτω στοιχεία ανά Κατάστημα και ανά Επωνυμία Μάρκας:

- Την Ποσότητα των προϊόντων κάθε μάρκας που πωλήθηκαν ανά Κατάστημα.
- Την Καθαρή Αξία των πωληθέντων προϊόντων κάθε μάρκας ανά Κατάστημα, δηλαδή την αξία των πωλήσεων χωρίς τον ΦΠΑ.
- Το Κόστος των πωληθέντων προϊόντων κάθε Μάρκας, δηλαδή την Αξία του κόστους που είχε το κατάστημα για την αγορά των προϊόντων της κάθε Μάρκας.

- Eshop Παραγγελίες. Με αυτή την επιλογή ο χρήστης μπορεί να δει τις παραγγελίες κάθε καταστήματος που εκρεμμούν. Κάθε παραγγελία θα αναγράφει το Όνομα και το Επώνυμο του πελάτη που πραγματοποίησε την παραγγελία, την Κατάσταση Παραγγελίας, τον Κωδικό της, Ημερομηνία Παραγγελίας και επίσης θα προβάλλονται τα παρακάτω στοιχεία:
    - Την Ποσότητα των προϊόντων που υπάρχουν συνολικά στην παραγγελία.
    - Τυχόν Παρατηρήσεις που σημειώθηκαν από τον πελάτη κατά την πραγματοποίηση της παραγγελίας.
    - Επιλογή για εμφάνιση των προϊόντων που έχει παραγγείλει ο χρήστης.

Η επιλογή Προϊόντα θα εμφανίζει τους τίτλους των προϊόντων που βρίσκονται μέσα στην παραγγελία καθώς και την τιμή τους και την ποσότητά τους. Επιλέγοντας κάποιο από αυτά ο χρήστης μπορεί να δει Γενικές Πληροφορίες για το συγκεκριμένο προϊόν όπως:

- Κωδικό Προϊόντος, BarCode, Περιγραφή, Κατασκευαστή, Τύπο Κατασκευαστή, Κατηγορία, BarCode Προμηθευτή, Τιμή Λιανικής και Τιμή Χονδρικής.
- Ποσότητες ανά αποθήκη, Αναμενόμενα, Υπόλοιπα, Διαθέσιμα, Δεσμευμένα.
- Τις διαθέσιμες τιμές που έχει το προϊόν.

- Παραστατικά Πωλήσεων. Με αυτή την επιλογή ο χρήστης μπορεί να δει τα παραστατικά που έχουν κοπεί από κάθε κατάστημα όσον αφορά τις πωλήσεις. Κάθε παραστατικό θα αναγράφει το Όνομα και το Επώνυμο του πελάτη που πραγματοποίησε το παραστατικό, τον τύπο, τον Κωδικό του, την Ημερομηνία του και επίσης θα προβάλλονται τα παρακάτω στοιχεία:

- Την Ποσότητα των προϊόντων που υπάρχουν συνολικά στο παραστατικό.
- Τυχόν Παρατηρήσεις που σημειώθηκαν από τον πελάτη κατά την πραγματοποίηση του παραστατικού.
- Επιλογή για εμφάνιση των προϊόντων που αγόρασε ο πελάτης.

Η επιλογή Προϊόντα θα εμφανίζει τους τίτλους των προϊόντων που βρίσκονται μέσα στην παραγγελία καθώς και την τιμή τους και την ποσότητά τους. Επιλέγοντας κάποιο από αυτά ο χρήστης μπορεί να δει Γενικές Πληροφορίες για το συγκεκριμένο προϊόν όπως:

- Κωδικό Προϊόντος, BarCode, Περιγραφή, Κατασκευαστή, Τύπο Κατασκευαστή, Κατηγορία, BarCode Προμηθευτή, Τιμή Λιανικής και Τιμή Χονδρικής.
- Ποσότητες ανά αποθήκη, Αναμενόμενα, Υπόλοιπα, Διαθέσιμα, Δεσμευμένα.
- Τις διαθέσιμες τιμές που έχει το προϊόν.

- Παραστατικά Αγορών. Με αυτή την επιλογή ο χρήστης μπορεί να δει τα παραστατικά που έχουν κοπεί από κάθε κατάστημα όσον αφορά τις αγορές. Κάθε παραστατικό θα αναγράφει το Όνομα και το Επώνυμο του προμηθευτή για τον οποίο πραγματοποίηθηκε το παραστατικό, τον τύπο, τον Κωδικό του, την Ημερομηνία του και επίσης θα προβάλλονται τα παρακάτω στοιχεία:
  - Την Ποσότητα των προϊόντων που υπάρχουν συνολικά στο παραστατικό.
  - Τυχόν Παρατηρήσεις που σημειώθηκαν κατά την πραγματοποίηση του παραστατικού.
  - Επιλογή για εμφάνιση των προϊόντων που αγοράστηκαν.

Η επιλογή Προϊόντα θα εμφανίζει τους τίτλους των προϊόντων που βρίσκονται μέσα στην παραγγελία καθώς και την τιμή τους και την ποσότητά τους. Επιλέγοντας κάποιο από αυτά ο χρήστης μπορεί να δει Γενικές Πληροφορίες για το συγκεκριμένο προϊόν όπως:

- Κωδικό Προϊόντος, BarCode, Περιγραφή, Κατασκευαστή, Τύπο Κατασκευαστή, Κατηγορία,

BarCode Προμηθευτή, Τιμή Λιανικής και Τιμή Χονδρικής.

- Ποσότητες ανά αποθήκη, Αναμενόμενα, Υπόλοιπα, Διαθέσιμα, Δεσμευμένα.

Τις διαθέσιμες τιμές που έχει το προϊόν.

- Αγορές ανά Κατηγορία. Με αυτή την επιλογή ο χρήστης μπορεί να δει το κόστος κάθε καταστήματος αθροίζοντας αγορές με βάση την κατηγορία των προϊόντων που έχουν αγοραστεί. Ο χρήστης θα οδηγείται πρώτα σε μια φόρμα Φίλτρων όπου θα επιλέγει αν θελήσει να δει αποτελέσματα κάποιου συγκεκριμένου καταστήματος ή και κάποιας συγκεκριμένης κατηγορίας προϊόντος. Σε κάθε κατηγορία που προβάλλεται ο χρήστης μπορεί πιο ειδικά να δει τα παρακάτω στοιχεία ανά Κατάστημα και ανά Κατηγορία:

    - Την **Ποσότητα** των προϊόντων κάθε κατηγορίας που αγοράστηκαν ανά κατάστημα.
    - Την Τελική Αξία των αγορασθέντων προϊόντων κάθε κατηγορίας ανά κατάστημα, δηλαδή την Αξία των πωλήσεων συμπεριλαμβανομένου τον ΦΠΑ.
    - Την Καθαρή Αξία των αγορασθέντων προϊόντων κάθε κατηγορίας ανά κατάστημα, δηλαδή την Αξία των αγορών χωρίς τον ΦΠΑ.
    - Το Κόστος των αγορασθέντων προϊόντων κάθε κατηγορίας, δηλαδή την Αξία του κόστους που είχε το κατάστημα για την αγορά των προϊόντων τις κάθε κατηγορίας.

Τέλος μπορεί να δει την Συνολική Τελική αξία κάθε καταστήματος μέσω των αγορών ανά κατηγορία που προβάλλονται.

- Αγορές ανά Κατασκευαστή. Με αυτή την επιλογή ο χρήστης μπορεί να δει το κόστος κάθε καταστήματος αθροίζοντας αγορές με βάση τον Κατασκευαστή των προϊόντων που έχουν αγοραστεί. Ο χρήστης θα οδηγείται πρώτα σε μια φόρμα Φίλτρων όπου θα επιλέγει αν θελήσει να δει αποτελέσματα κάποιου συγκεκριμένου καταστήματος ή και κάποιου συγκεκριμένου Κατασκευαστή προϊόντος. Σε κάθε Κατασκευαστή που προβάλλεται ο χρήστης μπορεί πιο ειδικά να δει τα παρακάτω στοιχεία ανά Κατάστημα και ανά Επωνυμία Κατασκευαστή:

- Την **Ποσότητα** των προϊόντων κάθε προμηθευτή που αγοράστηκαν ανά Κατάστημα.
- Την Καθαρή Αξία των αγορασθέντων προϊόντων κάθε προμηθευτή ανά κατάστημα, δηλαδή την Αξία των αγορών χωρίς τον ΦΠΑ.
- Το Κόστος των πωληθέντων προϊόντων κάθε Προμηθευτή, δηλαδή την Αξία του κόστους που είχε το κατάστημα για την αγορά των προϊόντων του κάθε προμηθευτή.

- Αγορές ανά Προμηθευτή. Με αυτή την επιλογή ο χρήστης μπορεί να δει το κόστος κάθε καταστήματος αθροίζοντας αγορές με βάση τον Προμηθευτή των προϊόντων που έχουν αγοραστεί. Ο χρήστης θα οδηγείται πρώτα σε μια φόρμα Φίλτρων όπου θα επιλέγει αν θελήσει να δει αποτελέσματα κάποιου συγκεκριμένου καταστήματος ή και κάποιου συγκεκριμένου Προμηθευτή προϊόντος. Σε κάθε προμηθευτή που προβάλλεται ο χρήστης μπορεί πιο ειδικά να δει τα παρακάτω στοιχεία ανά Κατάστημα και ανά Επωνυμία Προμηθευτή:

    - Την **Ποσότητα** των προϊόντων κάθε προμηθευτή που αγοράστηκαν ανά Κατάστημα.
    - Την Καθαρή Αξία των αγορασθέντων προϊόντων κάθε προμηθευτή ανά κατάστημα, δηλαδή την Αξία των αγορών χωρίς τον ΦΠΑ.
    - Το Κόστος των αγορασθέντων προϊόντων κάθε Προμηθευτή, δηλαδή την Αξία του κόστους που είχε το κατάστημα για την αγορά των προϊόντων του κάθε προμηθευτή.

Τέλος μπορεί να δει την Συνολική Τελική αξία κάθε καταστήματος μέσω των αγορών ανά κατηγορία που προβάλλονται.

- Αγορές ανά Brand. Με αυτή την επιλογή ο χρήστης μπορεί να δει το κόστος κάθε καταστήματος αθροίζοντας αγορές με βάση την μάρκα των προϊόντων που έχουν αγοραστεί και μετά ανά κατηγορία. Ο χρήστης θα οδηγείται πρώτα σε μια φόρμα Φίλτρων όπου θα επιλέγει αν θελήσει να δει αποτελέσματα κάποιου συγκεκριμένου καταστήματος ή και κάποιας συγκεκριμένης μάρκας προϊόντος.

Σε κάθε μάρκα που προβάλλεται ο χρήστης μπορεί πιο ειδικά να δει τα παρακάτω στοιχεία ανά Κατάστημα και ανά Επωνυμία Μάρκας:

- Την Ποσότητα των προϊόντων κάθε μάρκας που αγοράστηκαν ανά Κατάστημα.
- Την Καθαρή Αξία των αγορασθέντων προϊόντων κάθε μάρκας ανά Κατάστημα, δηλαδή την αξία των αγορών χωρίς τον ΦΠΑ.
- Την Θεωρητική Αξία των αγορασθέντων προϊόντων κάθε μάρκας, δηλαδή την Αξία ???????????
- Το Κόστος των αγορασθέντων προϊόντων κάθε Μάρκας, δηλαδή την Αξία του κόστους που είχε το κατάστημα για την αγορά των προϊόντων της κάθε Μάρκας.

- Αναζήτηση Προϊόντων. Με αυτή την επιλογή ο χρήστης μπορεί να δει πληροφορίες για τα προϊόντα τα οποία υπάρχουν καταχωρημένα στην βάση δεδομένων. Ο χρήστης θα μπορεί να πληκτρολογεί τον τίτλο ή μέρος του τίτλου του προϊόντος το οποίο αναζητεί και θα εμφανίζονται σε λίστα από κάτω τα παρακάτω στοιχεία:
  - Ολόκληρος ο τίτλος του προϊόντος.
  - Το όνομα κατασκευαστή του προϊόντος.
  - Ο τύπος του προϊόντος.
  - Η συνολική ποσότητα ελεύθερων τεμαχίων στα καταστήματα.
  - Η τιμή λιανικής που του έχει οριστεί.

Επιλέγοντας κάποιο από αυτά ο χρήστης πατώντας τον τίτλο του μπορεί να δει Γενικές Πληροφορίες για το συγκεκριμένο προϊόν όπως:

- Κωδικό Προϊόντος, BarCode, Περιγραφή, Κατασκευαστή, Τύπο Κατασκευαστή, Κατηγορία, BarCode Προμηθευτή, Τιμή Λιανικής και Τιμή Χονδρικής.
- Ποσότητες ανά αποθήκη, Αναμενόμενα, Υπόλοιπα, Διαθέσιμα, Δεσμευμένα.
- Τις διαθέσιμες τιμές που έχει το προϊόν.

5 Περιγραφή Τεχνολογιών

5.1 Delphi

Για την Υλοποίηση του Project θα χρησιμοποιηθεί η γλώσσα προγραμματισμού Delphi σε συνδυασμό με το framework Firemonkey. Η Delphi είναι μια διάλεκτο Αντικειμενοστραφούς Pascal η οποία αναπτύσσεται πλέον από την εταιρεία Embarcadero.

5.2 RAD Studio XE5

Το IDE το οποίο χρησιμοποιήθηκε είναι το RAD Studio XE5 της εταιρείας Embarcadero.

Το RAD Studio XE5 δίνει την δυνατότητα δημιουργίας παραθυρικών και μη, native εφαρμογών για Windows με το Framework VCL γράφοντας είτε Delphi είτε C++ και multi-platform εφαρμογές με το Framework Firemonkey γράφοντας Delphi. Επίσης δίνει την δυνατότητα δημιουργίας Datasnap Server η οποία είναι μια τεχνολογία που επιτρέπει τη δημιουργία πολυεπίπεδων εφαρμογών με σύνδεση σε βάσεις δεδομένων. Η δυνατότητα που μας δίνει το RAD Studio να κάνουμε compile για τις πλατφόρμες Windows, MacOS, iOS και Android προέρχεται από την αρχιτεκτονική του compiler LLVM.

Το IDE κατά την εγκατάστασή του επίσης εγκαθιστά τα plugins και τα dependencies τα οποία χρειάζονται για το compile και το deployment σε πλατφόρμα Android. Δηλαδή κάνει εγκατάσταση και ρύθμιση του JDK και του Android NDK.

Επίσης παρέχεται ένα πρόγραμμα λεγόμενο PA Server το οποίο πρέπει να εγκατασταθεί σε υπολογιστή Mac για να υπάρξει δυνατότητα compile και deployment σε πλατφόρμα iOS. Ο Mac υπολογιστής θα πρέπει στην συνέχεια να έχει εγκατεστημένο το Xcode, στο οποίο ουσιαστικά το IDE κάνει πάσα τον μεταφρασμένο κώδικα της Delphi ώστε το ίδιο να δημιουργήσει το αρχείο προς deployment σε iOS συσκευές.

Εν λειτουργεία του PA Server και με την ύπαρξη του Xcode, τότε το RAD Studio στο στήσιμο του Connection Profile στον Mac, λαμβάνει το κατάλληλο SDK το οποίο χρειάζεται για την μετάφραση του κώδικα. [1]

5.3 *LLVM Compiler*

Το RAD Studio διαθέτει compilers οι οποίοι μεταφράζουν την γλώσσα που είναι φτιαγμένη μια εφαρμογη σε μία αναπαράσταση κατανοητή για τον Compiler LLVM ο οποίος λειτουργεί ως δεύτερος compiler. Λαμβάνοντας αυτή την αναπαράσταση ο LLVM Compiler χρησιμοποιεί δικά του εργαλεία για μετατρέψει αυτή την αναπαράσταση σε native κώδικα για τον Επεξεργαστή ή σε μια Εκτελέσιμη Ενδιάμεση Αναπαράσταση. Φτιάχνοντας έναν μεταφραστή από μια γλώσσα σε μια τέτοια μορφή Ενδιάμεσης Αναπαράστασης (IR), μπορεί κάποιος να αξιοποιήσει κάθε διαθέσιμη αρχιτεκτονική επεξεργαστών συμπεριλαμβανομένου των ARM.

Τέλος ο Marco Cantu, Delphi Product Manager της Embarcadero μας λέει πως, «η αρχιτεκτονική LLVM αναπτύσσεται συνεχώς και από κόσμο που υποστηρίζει το open source αλλά και τον κόσμο που υποστηρίζει ιδιοτικές τεχνολογίες. Για παράδειγμα η Apple χρησιμοποιεί LLVM στο Xcode για το χτίσημο Mac OS X και iOS εφαρμογές» [2].

5.4 *Firemonkey Framework*

Το Firemonkey είναι ένα Framework της Embarcadero το οποίο επιτρέπει ταυτόχρονη ανάπτυξη εφαρμογών για πολλαπλές πλατφόρμες. Κάθε εφαρμογή που δημιουργείται μέσω του Firemonkey μπορεί να λειτουργήσει σε Windows, OS X, iOS και Android. Επίσης δίνει την δυνατότητα στον προγραμματιστή να χρησιμοποιήσει τα περισσότερες εσωτερικές native λειτουργίες κάθε λειτουργικού και είναι το μόνο μέρος το οποίο χρειάζεται αναγκαστικά εξειδικευμένο κώδικα ανάλογα την πλατφόρμα που θέλουμε.

Οι εφαρμογές που δημιουργούνται από το Firemonkey θεωρούνται native σε κάθε πλατφόρμα. Για την απεικόνιση των στοιχείων διεπαφής κάνει βαριά χρήση της κάρτας γραφικών. Επίσης, η ανάπτυξη εφαρμογών Firemonkey έχουν τα εξής χαρακτηριστικά:

- Όλα τα στοιχεία διεπαφής είναι ζωγραφισμένα διανυσματικά.
- Κάθε στοιχείο διεπαφής μπορεί να είναι παιδί κάθε άλλου στοιχείου διεπαφής το οποίο δίνει την δυνατότητα δημιουργίας νέων στοιχείων διεπαφής που συνδυάζουν λειτουργίες άλλων.
- Υποστηρίζουν Οπτικά Εφέ και Animations.

Τέλος, οι εφαρμογές λειτουργούν μόνο σε επεξεργαστές δομής armV7 και επίσης χρειάζονται κάρτα γραφικών με τεχνολογία NEON λόγω της βαριάς χρήσης που κάνει το Firemonkey.

5.5 *Datasnap Server*

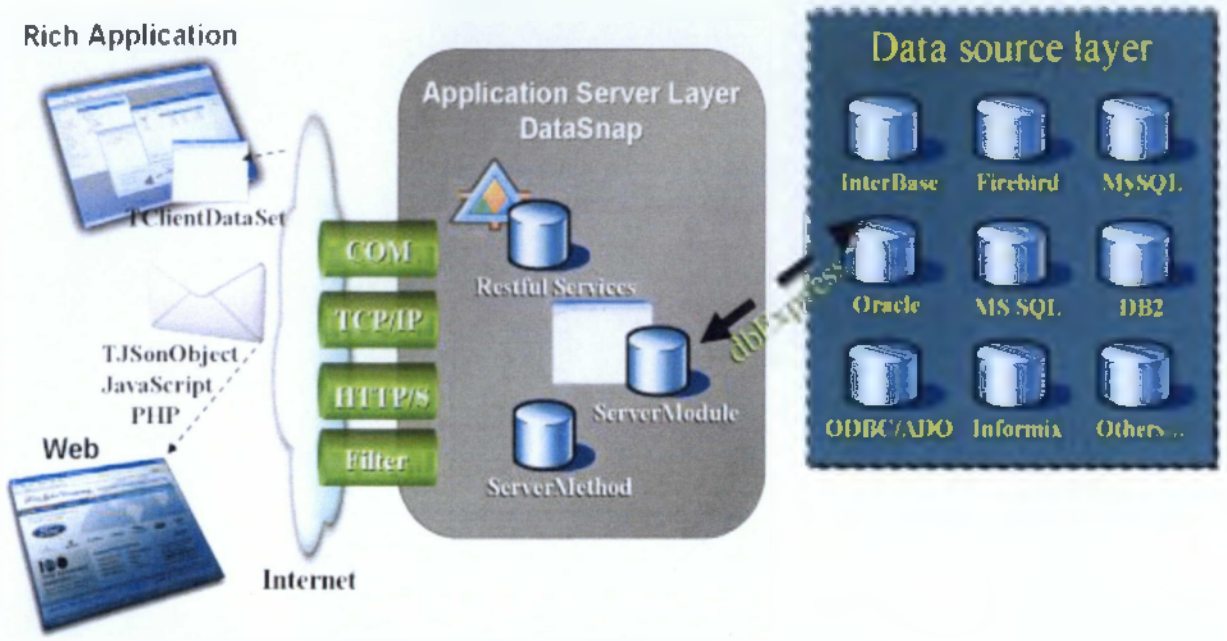
Πέρα από το Firemonkey, χρησιμοποιώντας την Delphi θα φτιάξουμε και τον Server ο οποίος θα γεφυρώνει την επικοινωνία της φορητής εφαρμογής μας με την βάση δεδομένων. Ο Datasnap Server μπορεί να επικοινωνεί με TCP/IP,HTTP ή και Rest.

Μια TCP/IP σύνδεση χρησιμοποιεί το Component (DSTCPServerTransport), το οποίο προσφέρει μια state-full και σταθερή σύνδεση με τους clients. Αυτος ο τρόπος σύνδεσης συνήθως αρμώζει σε ένα εσωτερικό δίκτυο. Οι socket συνδέσεις προσφέρουν συμπίεση και κρυπτογράφηση των δεδομένων τα οποία πρέπει να υλοποιηθούν και στον Server και στον Client.

Μια HTTP σύνδεση χρησιμοποιεί το Component (DSHTTPService), το οποίο προσφέρει μία state-full σύνδεση πάνω σε ένα stateless πρωτόκολλο μέσα από μια custom διαχείριση συνεδρίας και την δυνατότητα να κρατάει τα αντικείμενα του Server ενεργά για κάθε συνεδρία. Είναι δυνατή και η επιλογή HTTPS για περισσότερη ασφάλεια.

Μια REST σύνδεση η οποία είναι βασισμένη είτε στην παραπάνω HTTP σύνδεση είτε στην αρχιτεκτονική WebBroker που επιτρέπει την δημιουργία IIS module ή και stand-alone web servers. Και στις δύο περιπτώσεις ο Datasnap Server έχει την συμπεριφορά ενός stateless server και υπάρχει πολύ μεγαλύτερη ευελιξία στα εργαλεία ανάπτυξης εφαρμογών.

Κάθε ένα από τα παραπάνω μοντέλα μπορεί να υλοποιηθεί στο ίδιο server application δεδομένου ότι ένας Datasnap Server μπορεί να προσφέρει πολλαπλά είδη συνδέσεων (sockets και HTTP) και μια HTTP σύνδεση έχει διπλή διεπαφή (HTTP και REST).



Σχεδιάγραμμα Αρχιτεκτονικής Τεχνολογιών

5.6 DSSERVER Component

Το DSServer είναι το κύριο μέρος της σύνθεσης του του Server το οποίο χρειάζεται για να γεφυρώσει την λειτουργία όλων των υπόλοιπων components του Datasnap μαζί. Αυτό το Component χειρίζεται τις συνδέσεις και τα callbacks και γι αυτό πρέπει να υπάρχει μόνο ένα DSServer Component παρόν.

Στην περίπτωση των Εφαρμογών Web πρέπει να δημιουργεί ένα συγκεκριμένο Data Module το οποίο θα φιλοξενεί ένα DSServer component αλλιώς θα πρέπει να υπάρχει ένας Server για κάθε web module το οποίο θα δημιουργείται με κάθε νέο HTTP request.

5.6.1 DSServerClass Component & LifeTime των Server Objects

Το DSServerClass είναι ένα component το οποίο χρειάζεται για κάθε κλάση που επιθυμούμε να εκθέσουμε σε εφαρμογές Client. Κάθε service εκθέτει μεθόδους και δεδομένα χρησιμοποιώντας μία ή περισσότερες κλάσεις του Server. Το DSServerClass Component είναι το εργοστάσιο το οποίο δημιουργεί αντικείμενα της κλάσης της οποίας θα ήθελε κάποιος να καλέσει απομακρυσμένα από την εφαρμογή Client.

Το δεύτερο σημαντικό χαρακτηριστικό αυτού του Component είναι το ότι ορίζει τον κύκλο ζωής των αντικειμένων των κλάσεων.

5.6.2 Invocation:

Ο κύκλος ζωής Invocation υποδεικνύει πως για κάθε επίκληση σε ένα service θα δημιουργήσει ένα αντικείμενο, θα καλέσει την συγκεκριμένη μέθοδο και αμέσως θα απελευθερώσει το αντικείμενο. Με λίγα λόγια ο Server θα δημιουργεί ένα αντικείμενο για κάθε αίτημα άσχετα από τον χρήστη και την συνεδρία του αποφεύγοντας το caching σε ένα μοντέλο το οποίο ταιριάζει περισσότερο σε μια Stateless αρχιτεκτονική βασισμένη σε HTTP και REST πρωτόκολλα.

5.6.3 Session:

Ο κύκλος ζωής Session υποδεικνύει πως για κάθε συνεδρία που πραγματοποιείται όλα τα αντικείμενα που δημιουργούνται παραμένουν στην μνήμη όσο αυτή είναι ζωντανή. Κάθε αίτημα από τον ίδιο χρήστη που παραμένει στην ίδια συνεδρία θα χρησιμοποιήσει το ίδιο αντικείμενο που έχει δημιουργηθεί στον Server.

5.6.4 Server:

Ο κύκλος ζωής Server υποδεικνύει πως το αντικείμενο είναι ένα Global αντικείμενο το οποίο δημιουργείται στην πρώτη επίκληση και χρησιμοποιείται το ίδιο άσχετα από τον χρήστη ή την συνεδρία. Το αντικείμενο αυτό παραμένει στην μνήμη για πάντα.

Ανάλογα τον τύπο σύνδεσης που μπορεί να διαλέξει κάποιος τότε δεν είναι όλα τα μοντέλα κύκλου ζωής διαθέσιμα. Πιο συγκεκριμένα ο Session κύκλος ζωής δεν είναι διαθέσιμος για τους REST Servers. Αυτό γίνεται επειδή τα αντικείμενα που δημιουργούνται στον Server θα είχαν ένα ογκώδες ίχνος στην μνήμη τα οποία δεν θα έπρεπε να μένουν στην μνήμη για πολύ ώρα σε μια μακρόχρονη συνεδρία όπου ο Client θα αργούσε να κάνει αιτήματα.

Ο δημιουργία της τεχνολογίας Datasnap έγινε με στόχο την εύκολη έκθεση πινάκων βάσεων δεδομένων σε Client Εφαρμογές. Αυτό γινόταν μέσω μια συγκεκριμένης διεπαφής που ονομαζόταν IAppServer η οποία εκτιθόταν από απομακρυσμένα data modules. Οι μέθοδοι για αυτή τη δουλειά δεν χρησιμοποιούνταν άμεσα αλλά μπορούσε κάποιος να εκθέσει ένα Component DataSetProvider το οποίο χρησιμοποιούνταν από το απομακρυσμένο data module για να ανταποκριθεί στις μεθόδους της IAppServer διεπαφής. Από τη μεριά των Clients μπορεί κάποιος να χρησιμοποιήσει συγκεκριμένα Components σύνδεσης (DSProviderConnection) και τα ClientDataSet components για να συνδεθεί στην IAppServer διεπαφή και στους πίνακες που εκθέτουν οι providers.

Επιπλέον ο Datasnap Server προσφέρει τη δυνατότητα να εκθέσει Custom Μεθόδους. Αυτό αρχικά γίνονταν μέσω COM, πλέον όμως η επίκληση απομακρυσμένων μεθόδων χρησιμοποιεί ένα Custom Layer το οποίο βασίζεται σε server κλάσεις, RTTI και μερικούς κανόνες για πέρασμα παραμέτρων. Οι server μέθοδοι μπορούν να χρησιμοποιήσουν όλα τα βασικά data types (ακέραιους, αλφαριθμητικά, χαρακτήρες, αριθμούς κινητής υποδιαστολής κλπ) αλλά και πιο σύνθετους όπως dataset, streams, JSON data δομές, ακόμη και ολόκληρα Delphi αντικείμενα μέσω κάποιων Marshaling κλάσεων.

Η διεπαφή IAppServer ειναι αυστηρά δεμένη με το state-full μοντέλο καθώς υποστηρίζει έννοιες "next data packet". Αυτός είναι ο λόγος που η συγκεκριμένη διεπαφή δεν είναι διαθέσιμη για stateless REST εφαρμογές χτισμένες με Datasnap. Δεδομένου όμως πως ένα REST service μπορεί να εκθέσει πίνακες, μπορεί κάποιος να μετακινήσει δεδομένα βάσεων δεδομένων εύκολα από τον Server στον Client έχει νόημα κανείς να διαλέξει τον συγκεκριμένο τύπο service διότι είναι πιο ανοιχτός, ευέλικτος και επεκτάσιμος.

5.7 Η Έννοια του REST

Στις μέρες μας έχουμε αρχίσει να βλέπουμε αυτόματες αλληλεπιδράσεις μεταξύ διαφορετικών ιστοσελίδων, μεταξύ ιστοσελίδων και Client Εφαρμογές, μεταξύ ιστοσελίδων και βάσεων δεδομένων. Πλέον οι πληροφορίες ανανεώνονται γρηγορότερα από όσο μπορούμε να τις επεξεργαστούμε οπότε υπάρχει έντονη

ανάγκη από Εφαρμογές που βρίσκουν και παρακολουθούν πληροφορίες οι οποίες προέρχονται από διαφορετικές πηγές.

Ως λύση στο παραπάνω υπάρχουν 2 βασικές τεχνολογίες που χρησιμοποιούνται. Το ένα είναι το SOAP (Simple Object Access Protocol) στο οποίο υπάρχει αναφορά στο site [3] . Η άλλη λύση είναι το REST.

Ο όρος REST προέρχεται από το Representational State Transfer και ήταν επινόηση του Roy Fielding το 2000. Στην συνέχεια έγινε συνώνυμο με το να έχει κάποιος πρόσβαση στα δεδομένα μέσω διαδικτύου χρησιμοποιώντας HTTP και URLs αντί να βασίζονται στο SOAP standard.

Η ιδέα ήταν πως όταν κάποιος ανοίγει μια διαδικτυακή πηγή χρησιμοποιώντας έναν Browser ή κάποια συγκεκριμένη Client Εφαρμογή, ο Server θα μας στείλει μια επισκόπηση της πηγής (μια HTML σελίδα, μια εικόνα ή και raw δεδομένα). Ο Client δέχεται αυτή την επισκόπηση σε ένα συγκεκριμένο state. Καθώς ο Client ανοίγει περισσότερες σελίδες και συλλέγει περαιτέρω πληροφορίες αυτό το state θα αλλάξει.

«Σκοπός του Representational State Transfer (REST) είναι να μπορούμε να επικαλεστούμε την εικόνα του πώς μια καλά-σχεδιασμένη Εφαρμογή Web συμπεριφέρεται: ένα διαδίκτυο ιστοσελίδων(σε virtual state) μέσα από το οποίο ο χρήστης προχωράει μέσω μιας εφαρμογής επιλέγοντας links (μετάβαση του state), τα οποία καταλήγουν στην επόμενη σελίδα η οποία μεταφέρεται στον χρήστη(επισκόπηση του επόμενου state του application).» [4]

5.7.1 Σημαντικά σημεία αρχιτεκτονικής του REST

- Το Rest χρησιμοποιεί URLs για να αναγνωρίσει μια πηγή σε ένα server.
- Το Rest χρησιμοποιεί HTTP μεθόδους για να ορίσει ποια λειτουργία να εκτελεστεί (λήψη ή HTTP GET, δημιουργία ή HTTP PUT, ενημέρωση ή HTTP POST, διαγραφή ή HTTP DELETE)
- Το Rest χρησιμοποιεί HTTP παραμέτρους και για Queries αλλά και για POST ώστε να προσφέρει περισσότερες πληροφορίες στον Server.
- Το Rest βασίζεται στο HTTP για αυθεντικοποίηση χρήστη, κρυπτογράφηση και ασφάλεια επικοινωνίας χρησιμοποιώντας HTTPS.

- Το Rest επιστρέφει δεδομένα ως απλά αρχεία τα οποία χρησιμοποιούν πολλαπλά MIME formats όπως XMLJSON, images κλπ.

5.7.2 Απαιτήσεις ενός REST συστήματος

- Να είναι φτιαγμένο με την λογική του Client/Server.
- Να είναι Stateless
- Να είναι Cache-friendly, δηλαδή το ίδιο URL να επιστρέφει τα ίδια δεδομένα αν καλεστεί δύο συνεχόμενες φορές εκτός κι αν τα δεδομένα από την πλευρά του Server έχουν δεχτεί ενημερώσεις.[5]

6 Αρχιτεκτονική του Project

6.1 Datasnap Server

Για αρχή θα δημιουργήσουμε τον Datasnap Server στον οποίο θα υλοποιήσουμε όλες τις Custom μεθόδους τις οποίες θα εκθέτουμε στην εφαρμογή Client.

Δημιουργώντας έναν REST Datasnap Server Project δημιουργούνται 3 βασικά αρχεία .pas . Το ένα είναι το ServerFormUnit.pas το οποίο περιέχει το γραφικό κομμάτι του server. Το δεύτερο είναι το ServerMethodsUnit1.pas το οποίο περιέχει τις Custom μεθόδους που θα υλοποιήσουμε και θα διαχειρίζεται όλο το business logic. Και το τρίτο είναι το WebModuleUnit1.pas το οποίο περιέχει την λειτουργική ραχοκοκαλιά του REST Datasnap Server.

6.1.1 ServerFormUnit.Pas

Το ServerFormUnit είναι το αρχείο το οποίο περιέχει τα στοιχεία της φόρμας από την οποία θα αντιπροσωπεύεται το .exe του Server. Το ServerFormUnit το οποίο παράγεται από τον Wizard του REST Datasnap Server έχει 2 κουμπιά Start,Stop τα οποία ενεργοποιούν/ απενεργοποιούν τον Server, και ενα πεδίο στο οποίο συμπληρώνουμε το Port στο οποίο θα ακούει ο Server. Επίσης έχει το κουμπί "Open Browser" το οποίο φορτώνει στον Default Browser μας μια ιστοσελίδα η οποία εμφανίζει την διεπαφή που παρέχει ο Datasnap Server για όλες τις Custom μεθόδους που έχουμε υλοποιήσει στο 2° αρχείο .pas για το οποίο θα μιλήσουμε μετά. Ο χρήστης μπορεί να κάνει κλήση οποιαδήποτε διαθέσιμη μέθοδο και τα αποτελέσματα της θα επιστρέψουν σε μορφή JSON. Η λειτουργία αυτή είναι πάρα πολύ χρήσιμη στο να τεστάρουμε την καλή λειτουργία των μεθόδων μας. Σε περίπτωση που προκύψει κάποιο σφάλμα κατά την εκτέλεση της μεθόδου τότε μας επιστρέφει μήνυμα που μας καθοδηγεί στο πρόβλημα.

Σε αυτή τη φόρμα προσθέσαμε ένα ακόμη στοιχείο και μία λειτουργία. Θέλαμε να μπορούμε να παρακολουθούμε με κάποιο τρόπο την κίνηση την οποία θα είχε ο

Datasnap Server από πλευράς αιτημάτων μιας και σε REST μοντέλο δεν υπάρχει η έννοια της συνεδρίας.

Έτσι προσθέσαμε ένα ListBox το οποίο καταγράφει κάθε φορά ποια μέθοδο εκτελείται. Αυτό βέβαια κάποια στιγμή θα μπορούσε να μας δημιουργήσει πρόβλημα σε περίπτωση που ο Server αφεθεί ανοιχτός για πάρα πολύ καιρό καθώς το ListBox θα παραγέμιζε και θα κατανάλωνε περισσότερους πόρους από όσους θα έπρεπε. Για να το αντιμετωπίσουμε αυτό, στην διαδικασία καταγραφής όπου καλούνταν από τις μεθόδους προσθέσαμε έλεγχο ώστε ανά 10 εγγραφές να αδειάζει και να καταχωρεί τις εγγραφές αυτές σε ένα .txt αρχείο μαζί με την ώρα και την ημερομηνία που η μέθοδος καλέστηκε.

6.1.2 ServerMethodsUnit1

Το ServerMethodsUnit1 είναι το αρχείο το οποίο περιέχει όλες τις μεθόδους τις οποίες θα εκθέτουμε στους Clients μας. Όλες οι μέθοδοι που περιέχονται σε αυτή την φόρμα θα έχουν κύριο μέλημα να επικοινωνούν με την βάση δεδομένων μας, να δημιουργούν Datasets και να τα στέλνουν στους Clients.

Για την σύνδεση με την βάση δεδομένων θα χρησιμοποιηθεί η τεχνολογία FireDAC η οποία είναι η νεότερη τεχνολογία της Embarcadero για επικοινωνία με βάσεις δεδομένων.

Η σύνδεση με την βάση δεδομένων γίνεται με ένα FDConnection Component στο οποίο ορίζουμε τον κατάλληλο driver ανάλογα την πλατφόρμα Βάσεων Δεδομένων που θα χρησιμοποιήσουμε ο οποίος στην δική μας περίπτωση είναι ο MSSQLDriver. Πέρα από αυτό, πρέπει να ορίσουμε την τοποθεσία της Βάσης Δεδομένων, το όνομα της Βάσης Δεδομένων που θέλουμε καθώς και UserName και Password για την συγκεκριμένη Βάση Δεδομένων.

Η διαλογή των δεδομένων που θα ζητάνε οι Clients θα γίνεται μέσω TFDStoredProc Components τα οποία χρησιμοποιούν το FDConnection που αναφέραμε παραπάνω και θα εκτελούν όποια Stored Procedure τα παραμετροποιήσουμε να χρησιμοποιούν. Εφόσον ένα TFDStoredProc Component έχει δυνατότητα σύνδεσης με τη βάση δεδομένων και μπορεί να εντοπίσει την Stored

Procedure που θέλουμε, αυτόματα δημιουργεί και τις παραμέτρους με τις οποίες τροφοδοτεί τις Stored Procedure μας τις πληροφορίες που χρειάζονται για να εκτελεστούν. Οι περισσότερες είναι παραμετροποιημένες μέσα από το IDE της Delphi εκτός από μερικές περιπτώσεις που τροποποιούνται για να φιλτράρουμε συγκεκριμένες πληροφορίες.

Ένα κοινό σημείο όλων των μεθόδων που εκθέτονται στους Clients είναι ότι η πρώτη τους παράμετρο έχει αποθηκευμένο ένα ConnectionString το οποίο περιέχει πληροφορίες για την βάση δεδομένων στην οποία θέλουμε να αναφερθούμε. Αυτό το ConnectionString θα πρέπει να τροφοδοτείται σε κάθε κάλεσμα των μεθόδων διότι λόγω της τεχνολογίας REST δεν μπορούμε να κρατήσουμε συνεχή συνεδρία με την κάθε βάση δεδομένων.

6.1.3 WebModuleUnit1.pas

Το WebModuleUnit1 είναι από τα πιο σημαντικά μέρη του Datasnap Server καθώς περιέχει τον πυρήνα λειτουργίας του. Το αρχείο αποτελείται από τα Components DSServer και DSServerClass τα οποία ρυθμίζουν τον κύκλο ζωής των αντικειμένων που δημιουργούνται στον Server και το χρονικό όριο ζωής κάθε προσπάθειας επικοινωνίας από κάποιο Client μέχρι να λάβει TimeOut.

Το DSHTTPWebDispatcher Component το οποίο εφαρμόζει το REST πρωτόκολλο επικοινωνίας και το DSAuthenticationManager. Το DSAuthenticationManager είναι το Component που κάνει Authenticate και Authorize τους Clients οι οποίοι προσπαθούν να μιλήσουν στον Datasnap Server.

Το DSAuthenticationManager έχει 2 εξαιρετικά χρήσιμα Events τα οποία εκτελούνται κατά την προσπάθεια σύνδεσης κάποιου Client στον Datasnap Server. Το ένα είναι το OnUserAuthenticate και το δεύτερο είναι το OnUserAuthorize. [6]

Το OnUserAuthenticate είναι το Event το οποίο στο τέλος θα αφήσει τον χρήστη να συνδεθεί στον Datasnap Server ενώ το OnUserAuthorize είναι το Event το οποίο εφόσον πραγματοποιηθεί επιτυχές Authentication, αναθέτει ανάλογο «Ρόλο» στον συγκεκριμένο Client. Αυτοί οι ρόλοι συνήθως είναι του τύπου Administrator, Power User, User κλπ, ανάλογα τι ορίσει ο Developer. Χρησιμοποιώντας τους «Ρόλους» αυτούς μπορούμε να αποκρύψουμε συγκεκριμένες μεθόδους από κάποιον Client ώστε να μην έχει πρόσβαση σε αυτές.

Στο OnUserAuthenticate λαμβάνουμε ένα UserName και ένα Password από τον Client το οποίο ταυτοποιεί με ένα Query σε έναν συγκεκριμένο πίνακα της κεντρικής βάσης δεδομένων μας. Εφόσον υπάρξει ταυτοποίηση τότε το Query τραβάει πίσω ένα ConnectionString το οποίο θα επιστρέψει στον χρήστη ώστε μετά το επιτυχές login θα χρησιμοποιεί για να μπορεί να καλεί τις εκτεθειμένες μεθόδους του Datasnap Server. Το ConnectionString αυτό περιέχει πληροφορίες για την τοποθεσία της βάσης δεδομένων η οποία αφορά τον συγκεκριμένο Client όπως Database Server Address, Database Name, User Name, Password.

6.1.4 Κεντρική Βάση Δεδομένων για Authentication

Για λόγους δυναμικότητας, δημιουργήσαμε έναν μηχανισμό ο οποίος θα μας επιτρέπει εξωτερική ρύθμιση του Datasnap Server για την τοποθεσία της κεντρικής βάσης δεδομένων με την οποία θα πραγματοποιεί το Authentication.

Στον OnUserAuthenticate δημιουργούμε ένα TSQLConnection Component το οποίο παραμετροποιούμε μέσω ενός .txt αρχείου που πρέπει να υπάρχει στον ίδιο φάκελο με το .exe του Datasnap Server με όνομα sql.dat . Αυτό το αρχείο περιέχει τις πληροφορίες που χρειάζεται ο Datasnap Server για να συνδεθεί στην κεντρική βάση δεδομένων και να πραγματοποιήσει το Authentication.

6.2 Firemonkey Client Application

Το κομμάτι υλοποίησης της εφαρμογής του Client είναι αυτό το οποίο χρειάστηκε τον περισσότερο χρόνο κατά την υλοποίηση του Project.

Ξεκινώντας το Multi-Platform Development σε Mobile από τα πρώτα πράγματα που πρέπει να έχουμε στο νού μας είναι το γεγονός ότι η εφαρμογή που θα υλοποιήσουμε θα πρέπει εμφανισιακά να διατηρεί το Native στύλ της ανάλογης πλατφόρμας. Αδυναμία τήρησης του Native στυλ μπορεί να έχει ως αποτέσμα το να μην γίνει αποδεκτή η εφαρμογή μας στο Google Play ή στο i-Tunes. [7]

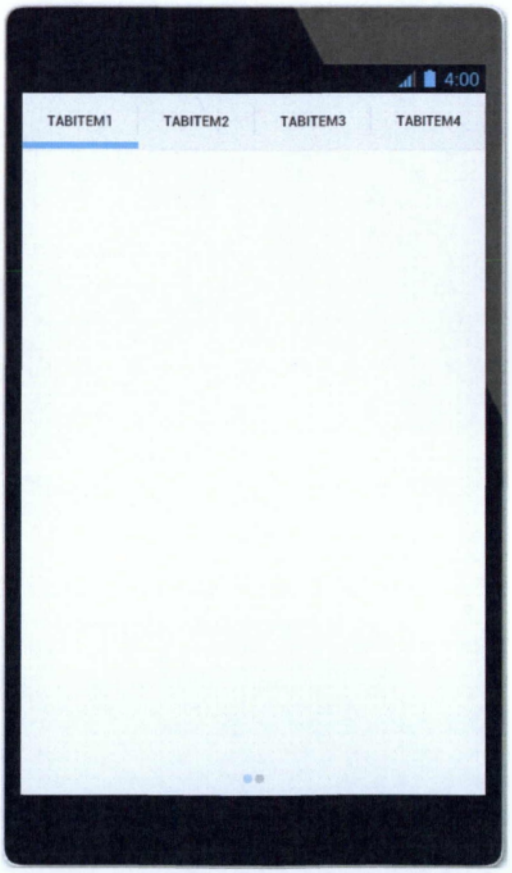
6.2.1 Ανάπτυξη γραφικού περιβάλλοντος για πολλαπλές πλατφόρμες

Στο Γραφικό Περιβάλλον της εφαρμογής μας θα υπάρξουν κομμάτια τα οποία θα μετακινούνται ή θα αλλάζουν στυλ κατά το τρέξιμο της εφαρμογής ανάλογα την πλατφόρμα στην οποία η εφαρμογή μας τρέχει. Μερικές ιδιαιτερότητες που πρέπει να έχουμε στο νου μας είναι :

6.2.2 TabControls:

Στο Android συνηθίζονται τα Tabs να είναι τοποθετημένα στην κορυφή της οθόνης μας ενώ στο iOS θα πρέπει να είναι τοποθετημένα στον πάτο της οθόνης μας και να συνοδεύονται από εικονίδιο (Glyph).

Εμφάνιση του TabControl σε iOS



Εμφάνιση του TabControl σε Android

6.2.3 Buttons:

Στο Android τα συνηθισμένα απλά κουμπιά είναι γκρίζα και έχουν αυστηρές γωνίες και συνήθως χρώμα γκρι, ενώ στο iOS θα πρέπει να μην έχουν πλαίσιο και να έχουν χρώμα ανοιχτό μπλε.

Εμφάνιση των Buttons σε iOS

Εμφάνιση των Buttons σε Android

6.2.4 Τρόπος Πλοήγησης της Εφαρμογής:

Όλες οι φόρμες θα πρέπει να έχουν τρόπους πλοήγησης και τρόπους να επιστρέφουν στην προηγούμενη φόρμα από αυτή που βλέπουν οι χρήστες. Για τις πλατφόρμες Android όμως ο Developer θα πρέπει να φροντίσει και την ανταπόκριση των Hardware Buttons που έχουν όλες οι συσκευές που έχουν εγκατεστημένο Android[8]. Τα Buttons πλοήγησης και για τις δύο πλατφόρμες θα πρέπει να είναι τοποθετημένα μέσα στο toolbar με alignment left αν πρόκειται για κουμπί που πάει τον χρήστη προς τα πίσω και alignment right αν πρόκειται για κουμπί που πάει τον χρήστη προς τα μπροστά. Και στις δύο περιπτώσεις τα κουμπιά θα πρέπει να διατηρούν χώρο των 5 pixel από τα τοιχώματα της οθόνης μας και να έχουν σωστά παραμετροποιημένο το StyleLookup property τους ώστε να μοιάζουν με Native Navigational Button στην ανάλογη πλατφόρμα.

Εμφάνιση του Navigation στο iOS

Εμφάνιση του Navigation στο Android

6.3 *Τρόποι Δόμησης της Εφαρμογής*

Υπάρχουν δύο δρόμοι για τον τρόπο που μπορεί κάποιος να αναπτύξει μία φορητή εφαρμογή χρησιμοποιώντας το Firemonkey Framework όσον αφορά τη βασική του δόμηση. Ο ένας δρόμος είναι το να «σπάσουμε» την λογική της εφαρμογής μας σε αρχεία τα οποία θα περιέχουν την δομή της κάθε φόρμας και τον κώδικα τον οποίο αφορά τα στοιχεία αυτής. Χρησιμοποιώντας αυτό τον τρόπο υπάρχουν τα εξής πλεονεκτήματα και μειονεκτήματα.

Στο Firemonkey όλες οι φόρμες δημιουργούνται κατά την εκκίνηση της εφαρμογής και μένουν φορτωμένες στην μνήμη. Αυτό σημαίνει πως αν έχουμε πάρα πολλές φόρμες, η εφαρμογή μας θα αργεί να ανοίξει δείχνοντας μια μαύρη οθόνη στον χρήστη, πράγμα το οποίο πολλές φορές δίνει την αίσθηση στον χρήστη ότι η εφαρμογή κόλλησε και προχωράει στο κλείσιμό της.

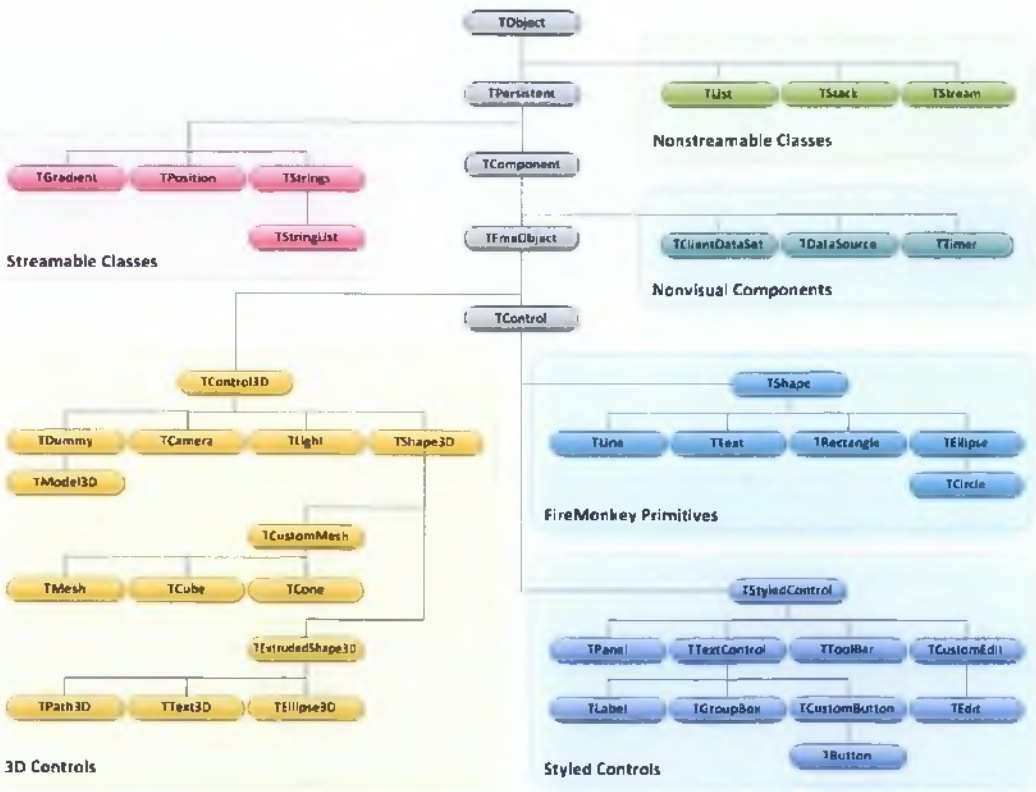
Μπορούμε από τις επιλογές του Project μας μέσα από το IDE να ορίσουμε ποιες φόρμες από τις διαθέσιμες να δημιουργούνται κατά το άνοιγμα της εφαρμογής. Από αυτό όμως προκύπτουν διάφορες δυσκολίες στην ανάπτυξη της εφαρμογής καθώς δεν υπάρχει τρόπος να ορίσουμε ποια φόρμα από τις διαθέσιμες να είναι ενεργή στην οθόνη μας καθώς και είναι αδύνατη η καταστροφή μιας φόρμας αφού δημιουργηθεί και κάθε προσπάθεια καταστροφής της οδηγεί σε Access Violation. Αυτό συμβαίνει διότι θα υπάρχουν ακόμη αναφορές ως προς το αντικείμενο το οποίο πάμε να καταστρέψουμε. Ακόμη και αν γλυτώσουμε το Access Violation, όσο υπάρχουν αναφορές ως προς το αντικείμενο προς καταστροφή η μνήμη που δεσμεύει δεν θα απελευθερωθεί. Η διαχείριση μνήμης που υπαγορεύει το Firemonkey είναι τύπου ARC (Automatic Reference Counting). Ο Compiler κατά το Compile αυτόματα εισάγει καλέσματα καταστροφής αντικειμένων στον κώδικα για την απελευθέρωση μνήμης [9]. Δυστυχώς όμως στο στάδιο ανάπτυξης του Firemonkey ο συγκεκριμένος τομέας είναι προβληματικός και η αδυναμία σωστής διαχείρισης μνήμης μπορεί να οδηγήσει το λειτουργικό σύστημα της φορητής μας συσκευής να διακόψει την λειτουργία της εφαρμογής μας λόγω κατανάλωσης υπερβολικών πόρων.

Για τον παραπάνω λόγω δυσκολίας της διαχείρισης πολλαπλών φορμών η ίδια η εταιρεία προσπαθεί να απομακρύνει τους Developers από τον συγκεκριμένο δρόμο αρχιτεκτονικής ενός Project. Παροτρύνει τους Developers όμως να δομήσουν όλη

τους την εφαρμογή σε μία φόρμα. Η τελευταία τεχνική είναι ο δεύτερος δρόμος ανάπτυξης μιας εφαρμογής.

Προσπαθήσαμε να βρούμε έναν ενδιάμεσο δρόμο εφόσον τα προβλήματα κάθε δρόμου ήταν προφανή και ερευνήσαμε τρόπους παντρέματος των δύο τεχνικών έτσι ώστε η εφαρμογή μας να είναι αποδοτική αλλά και εύκολη στην διαχείριση από πλευράς ανάπτυξης.

Ο τρόπος με τον οποίο λειτουργεί το Firemonkey Framework είναι πως η πρώτη φόρμα που δημιουργείται στο Project είναι η κύρια και οι άλλες φόρμες που μπορεί να δημιουργήσουμε βρίσκονται διαθέσιμες στο παρασκήνιο[10]. Για να καταφέρουμε με σωστό τρόπο να φέρουμε στο προσκήνιο μια διαθέσιμη φόρμα θα πρέπει να έχουμε ένα κενό χώρο στην κύρια φόρμα μας η οποία θα τεθεί γονέας της. Στο Firemonkey κάθε Visual Κλάση έχει ως ρίζα την κλάση TFMXObject και αυτό δίνει την δυνατότητα σε οποιοδήποτε αντικείμενο να είναι γονέας η παιδί οποιουδήποτε άλλου αντικειμένου.



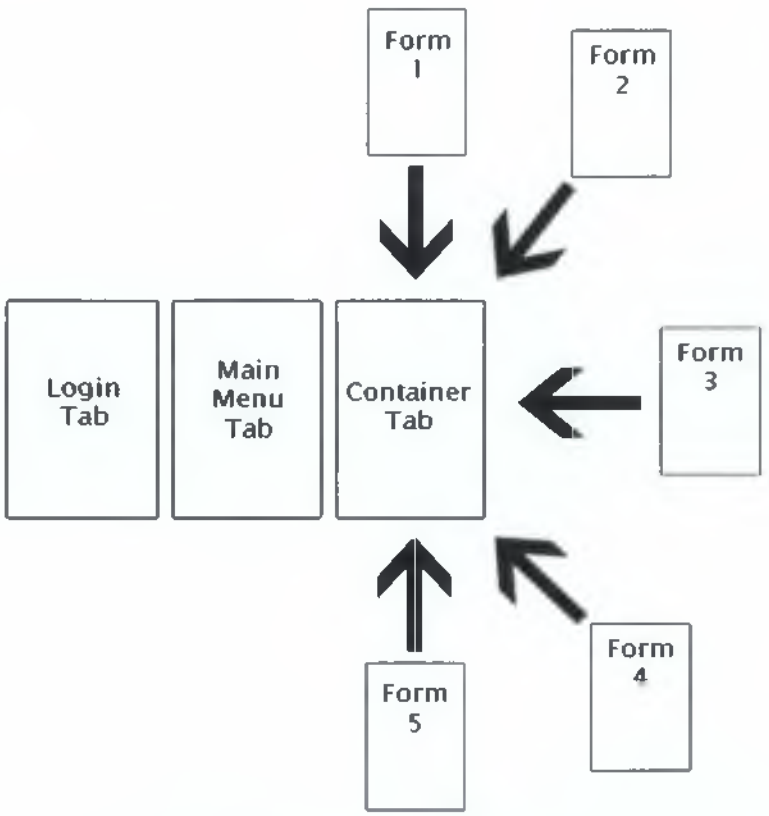
Ιεραρχία κλάσεων του Firemonkey

Έτσι λοιπόν θα δομήσουμε το γραφικό περιβάλλον της εφαρμογής μας ως έχει. Η κυρίος φόρμα θα έχει ένα TabControl το οποίο θα έχει 3 βασικά Tabs.

Το πρώτο Tab θα περιέχει το Login Form μας στο οποίο ο χρήστης θα χρησιμοποιεί τα συνθηματικά του για να κάνει την είσοδο στην εφαρμογή.

Το δεύτερο Tab θα περιέχει ένα TabControl το οποίο θα έχει σε κουμπιά τις επιλογές του χρήστη. Για να χωρέσουν όλα, αποφασίσαμε το 2° TabControl να έχει δύο Tabs για να χωριστούν οι επιλογές ώστε να είναι πιο λειτουργικό. Η πλοήγηση ανάμεσα σε αυτά τα δύο Tabs θα γίνεται με Gestures τα οποία συνηθίζουν να υπάρχουν σε όλες τις εφαρμογές φορητών συσκευών.

Το τρίτο Tab θα περιέχει ένα άδειο TRectangle το οποίο θα υπάρχει ως Container για τις φόρμες που θα δημιουργούνται αναλαμβάνοντας το parentage.



Γενικό πλάνο δομής Γραφικού Περιβάλλοντος

Αυτή θα είναι η ιδέα με την οποία θα δομηθεί το Γραφικό Περιβάλλον της Client Εφαρμογής μας. Στην συνέχεια θα δούμε πώς θα αξιοποιήσουμε τις μεθόδους που δημιουργήσαμε στον Datasnap Server και με ποιόν τρόπο θα επιτευχθεί η σύνδεση με αυτόν.

6.4 *Δημιουργία Interface επικοινωνίας με Datasnap Server*

Μέσα στο Firemonkey Mobile Application που δημιουργήσαμε θα χρειαστεί να προσθέσουμε ένα Datasnap REST Client Module το οποίο δημιουργούμε μέσα από Wizard του IDE της Embarcadero. Κατά την δημιουργία αυτού του Module ο Datasnap Server που έχουμε δημιουργήσει θα πρέπει να είναι ανοιχτός και ενεργοποιημένος καθώς θα χρειαστεί να δώσουμε τα στοιχεία της τοποθεσίας του, όνομα χρήστη και κωδικό. Δίνοντας τα στοιχεία, του Datasnap Server και τεστάροντας την συνδεσιμότητά τους δημιουργούνται δύο αρχεία μέσα στο project μας, το ClientModuleUnit1.pas και το ClientClassesUnit1.pas

Το αρχείο ClientModuleUnit1 μέσα έχει οδηγίες για την δημιουργία του βασικού Component που θα συνδέεται πάνω στον Datasnap Server μας και θα είναι αρχικοποιημένο σύμφωνα με τα στοιχεία που δώσαμε κατά τη δημιουργία του Datasnap REST Client Module.

Το αρχείο ClientClassesUnit1 είναι το αρχείο το οποίο κατά τη δημιουργία του, χάρη στη συνδεσιμότητα με τον Datasnap Server δημιουργεί αυτόματα το αρχέτυπο όλων των μεθόδων που εκθέτει ο Datasnap Server από το αρχείο ServerMethods1.pas προς στον συγκεκριμένο Client.

Μέσα στο ClientClassesUnit1 δημιουργείται η κλάση ServerMethods1Client η οποία έχει μεθόδους που μιλούν στις αντίστοιχες του Datasnap Server. Κάθε μέθοδος μέσα στο ClientClassesUnit1 λειτουργεί με τον εξής τρόπο.

Η μέθοδος δημιουργεί ένα αντικείμενο της κλάσης TDSRestCommand η οποία θα λάβει κάποια στοιχεία για να δημιουργήσει το αίτημα το οποίο θα σταλθεί στον Datasnap Server. Τα στοιχεία τα οποία τροφοδοτούμε το αντικείμενο της TDSRestCommand είναι τα εξής.

RequestType : Εδώ δίνεται ο τύπος του REST αιτήματος το οποίο θα συνταχθεί, στην δική μας περίπτωση όλες οι μέθοδοι θα έχουν RequestType 'GET'.

Text : Εδώ δίνεται το όνομα της κλάσης του Datasnap Server στον οποίο βρίσκεται η μέθοδος η οποία θέλουμε να καλέσουμε μαζί με το όνομα της μεθόδου συνδεδεμένα με τελεία.

Μετά από την τροφοδότηση των 2 παραπάνω μεταβλητών μελών του TDSRestCommand καλείται η μέθοδος Prepare η οποία εν συντομία ελέγχει τι παράμετροι χρειάζονται να τροφοδοτηθούν για σταλθούν ως ορίσματα στην μέθοδο που

θα καλέσουμε στον Datasnap και δημιουργεί ένα πίνακα γι αυτές. Ο πίνακας αυτός θα έχει μία θέση παραπάνω για μια παράμετρο ακόμη η οποία θα είναι για να επιστρέψει το αποτέλεσμα της μεθόδου το οποίο θα πακεταριστεί ως DataSet μετά το Execute.

6.5 Φόρμες

Όταν ένα κουμπί του κύριου μενού πατηθεί, τότε θα ξεκινήσει η μια διαδικασία ελέγχου για το αν είναι δημιουργημένη η φόρμα η οποία θέλουμε να καλέσουμε. Αν η φόρμα δεν υπάρχει διαθέσιμη, τότε την δημιουργούμε και θέτουμε ως γονέα της το ContainerTab αφού βεβαιωθούμε πως δεν υπάρχει άλλη φόρμα υιοθετημένη από το ContainerTab. Σε περίπτωση που είναι ήδη δημιουργημένη τότε απλά θέτουμε τον γονέα της.

Τα δεδομένα τα οποία θέλουμε να απεικονίζουμε θα ήταν βολικό για τον χώρο που μας προσφέρει μια κινητή συσκευή να φορτώνονται σε κάποια λίστα. Υπάρχουνε 2 διαφορετικά είδη λίστας τα οποία μπορούμε να χρησιμοποιήσουμε στο Firemonkey για απεικόνιση δεδομένων.

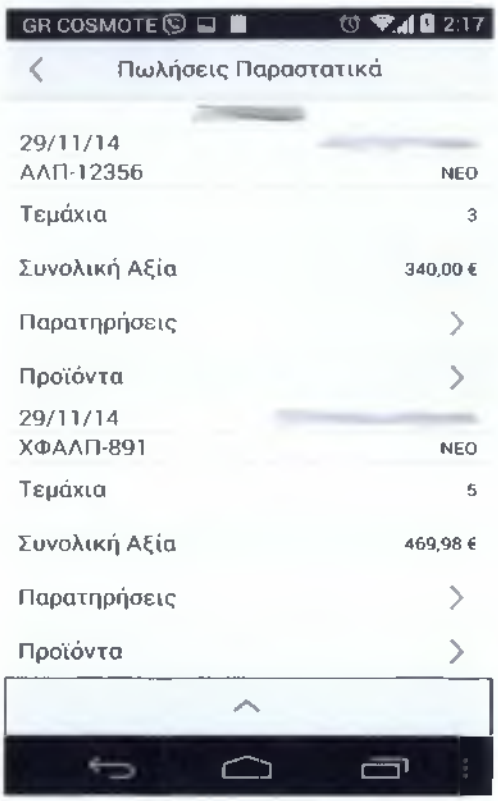
6.5.1 ListBox

Η λίστα ListBox μας δίνει μεγαλύτερη ευκολία στο να δημιουργήσουμε custom κελιά τα οποία μπορούν να φιλοξενούν όσα Components θέλουμε. Αυτό σημαίνει ότι είναι αρκετά ευέλικτη στον τρόπο απεικόνισης πληροφοριών και κυρίως ως προς το εικαστικό το οποίο μπορούμε να εφαρμόσουμε πάνω στα κελιά. Το πρόβλημα με το ListBox όμως είναι ότι καταναλώνει αρκετά περισσότερους πόρους και το scrolling είναι δυσκίνητο. Έτσι λοιπόν θα ήταν σοφή η χρήση του ListBox μόνο στην περίπτωση όπου θέλουμε κάτι εικαστικά πλούσιο και γνωρίζουμε ότι δεν θα χρειαστεί ποτέ να δημιουργήσουμε πολλά κελιά.

Παράδειγμα ενός πλούσιου ListBox

6.5.2 ListView

Η λίστα ListView είναι η κατάλληλη για να φορτώνουμε πάρα πολλά δεδομένα καθώς είναι πολύ πιο ελαφριά και αποδοτική με το scrolling αυτή τη φορά να είναι πολύ smooth[11]. Το πρόβλημα με το ListView όμως είναι ότι καθιστά ιδιαίτερα δύσκολη την διαδικασία του να διαμορφώσει κανείς περαιτέρω το εικαστικό των κελιών και την ποσότητα των πληροφοριών που μπορούμε να απεικονίσουμε. Για να λειτουργεί τόσο αποδοτικά το ListView, όλα τα elements τα οποία μπορεί να φιλοξενήσει ένα κελί ListViewItem είναι πολύ συγκεκριμένα και είναι optimized για το Component αυτό. Τα elements αυτά είναι ένα Accessory, ένα Detail, ένα GlyphButton και ένα Text. Τα συγκεκριμένα είναι τυποποιημένα ως πλήθος μέσα σε ένα κελί και ο Developer μπορεί μόνο να ορίσει το Visibility όλων αυτών πέρα από τις κύριες ιδιότητες τους.



Παράδειγμα ενός γρήγορου ListView

Σε περίπτωση που θα θέλαμε ένα κελί να μπορεί να φιλοξενήσει ένα οποιοδήποτε ακόμη στοιχείο, αυστηρά των προηγούμενων τύπων, θα πρέπει να δημιουργήσουμε μια νέα κλάση ListView Item η οποία θα χρειαστεί να επεκταθεί καταλλήλως ανάλογα με το τι θα θέλαμε να προστεθεί και που. Παρ'όλα αυτά δεν υπάρχει η δυνατότητα του να ορίσουμε ούτε χρώμα στο πίσω μέρος των κελιών ενός ListView.

6.5.3 Images

Άλλο ένα σημαντικό κομμάτι του γραφικού περιβάλλοντος σε φορητές συσκευές είναι οι εικόνες. Στην αγορά υπάρχουν εκατοντάδες διαφορετικές συσκευές με διαφορετικό μέγεθος οθόνης και διαφορετική ανάλυση εικόνας. Αυτό χρειάζεται κατάλληλη αντιμετώπιση έτσι ώστε σε κάθε συσκευή η εφαμοργή μας να φαίνεται το ίδιο όμορφη. Για να επιτευχθεί αυτό θα χρειαστεί να τροφοδοτήσουμε κάθε εικόνα με μερικές διαφορετικές αναλύσεις οι οποίες θα φορτώνονται ανάλογα την συσκευή στην οποία ανοίγει η συσκευή. Υπάρχουν 3 βασικά στοιχεία τα οποία πρέπει να έχουμε στο νου μας ώστε να κατανοήσουμε τον τρόπο με τον οποίο πρέπει να λειτουργούμε στο θέμα των εικόνων[12].

Πυκνότητα οθόνης: Η ποσότητα των pixel μέσα σε μια φυσική περιοχή της οθόνης. Συνήθως αυτό αναφέρεται και ως dpi (τελείες ανά ίντσα). Για παράδειγμα, μια οθόνη "χαμηλής" πυκνότητας έχει λιγότερα pixel σε μια δεδομένη φυσική περιοχή συγκριτικά με μία οθόνη "μεσαίας" ή "υψηλής" πυκνότητας οθόνη.

Ανάλυση οθόνης: Ο συνολικός αριθμός φυσικών pixel σε μία οθόνη. Η πυκνότητα οθόνης και η ανάλυση οθόνης υπολογίζονται σε φυσικές μονάδες – pixels.

Κλίμακα: Το Firemonkey πάντα χρησιμοποιεί λογικές συντεταγμένες και λογικά μεγέθη. Λειτουργώντας με bitmaps πολλαπλών αναλύσεων, το Firemonkey υπολογίζει τις σχέσης μεταξύ λογικών μονάδων και φυσικών μονάδων χρησιμοποιώντας την ιδιότητα της κλίμακας.

Μιλώντας πιο συγκεκριμένα για τις πλατφόρμες iOS και Android... για την πλατφόρμα του iOS είναι αρκετό να χρησιμοποιήσουμε εικόνες με κλίμακες 1 και 2. Για την πλατφόρμα του Android θα χρειαστεί να χρησιμοποιήσουμε εικόνες 1, 1.33, 1.50 και 2. Για παράδειγμα, αν δίναμε μια εικόνα ανάλυσης 100x100 στο iOS για την κλίμακα 1, τότε θα έπρεπε να δώσουμε και μια 200x200 κλίμακας 2 για τις συσκευές iOS με Retina.

Το Timage Component το οποίο θα χρησιμοποιούμε για την εμφάνιση των εικόνων μας, έχει μια ιδιότητα που ονομάζεται MultiResolution. Χρησιμοποιώντας αυτή την ιδιότητα μπορούμε μέσα σε ένα μοναδικό Timage να δώσουμε τις εικόνες μας σε διαφορετικές αναλύσεις και να του ορίσουμε ποια να φορτώνει ανάλογα την κλίμακα που υπαγορεύει η κάθε συσκευή. Η διαδικασία της επιλογής της κατάλληλης κλίμακας εικόνα γίνεται αυτόματα από το Firemonkey κατά την εκκίνηση της εφαρμογής μας.

6.5.4 Λήψη και αναπαράσταση δεδομένων

Σύμφωνα με το μέσο όγκο αποτελεσμάτων το οποίο μαντεύουμε ότι θα λαμβάνουν οι χρήστες αποφασίσαμε να χρησιμοποιήσουμε σε όλες μας τις φόρμες ListView εκτός από τρεις οι οποίες είναι σίγουρο πως δεν θα φορτωθούν υπερβολικά ποτέ τόσο ώστε να δημιουργήσουν αρνητική εμπειρία στον χρήστη.

Η κάθε φόρμα περιέχει μέσα της όλα τα Visual Components και τις μεθόδους για να λειτουργήσει εφόσον καλεστεί και έτσι θέλουμε έναν τρόπο να εκκινηθούν οι κατάλληλες διαδικασίες στο κάλεσμα της. Για τον σκοπό αυτό θα χρησιμοποιήσουμε το Event OnActivate της φόρμας η οποία θα παίρνει στην πλάτη της όλη τη βασική λειτουργία αυτής. Αυτός ο τρόπος θα εφαρμοστεί σε όλες τις φόρμες ανεξαιρέτως και η μόνη διαφορά που θα υπάρξει σε μερικές θα είναι ότι πρώτα θα εμφανίζουν μια φόρμα

φίλτρων στην οποία θα μπορεί ο χρήστης να περιορίσει τα αποτελέσματα που θα επιστρέψουν.

Όλα τα OnActivate των φορμών εκτός των φορμών Παραστατικού, Παραγγελιών Eshop και Αναζήτηση Προϊόντος λειτουργούν με τον εξής τρόπο.

1. Δημιουργία ενός άδειου DataSet.
2. Κάλεσμα τις κατάλληλης μεθόδου
3. Ειδοποίηση του ListView πως θα ξεκινήσουμε να το ενημερώνουμε.
4. Προσπέλαση του DataSet γραμμή γραμμή σε επανάληψη και πέρασμα κάθε στήλη εγγραφής σε ListView Item.
5. Ειδοποίηση του ListView πως τελείωσε η ενημέρωση έτσι ώστε να ανανεώσει το οπτικό του κομμάτι.

Στις φόρμες Παραστατικού, Παραγγελιών Eshop και Αναζήτησης Προϊόντων θα θέλαμε να προβάλουμε μερικά παραπάνω στοιχεία σχετικά με τιν εγγραφή που θα διάλεγε ο χρήστης. Οπότε αυτές οι φόρμες λειτουργούν με τον εξής τρόπο.

1. Δημιουργία ενός άδειου DataSet
2. Κάλεσμα τις κατάλληλης μεθόδου.
3. Ειδοποίηση του ListView πως θα ξεκινήσουμε να το ενημερώνουμε.
4. Προσπέλαση του DataSet και δημιουργία ListView Item ανά γραμμή εγγραφής. Σε συγκεκριμένα ListView Items τα οποία είναι αυτά τα οποία στο πάτημά τους θα φέρνουν κι άλλες πληροφορίες θα αποθηκεύσουμε κάποιος πληροφορίες της εγγραφής σε σημεία του αντικειμένου τα οποία είναι αόρατα όπως το Tag (int) και το ButtonText (Τα buttons είναι αόρατα στα δικά μας ListView). Στο Tag property θα αποθηκεύουμε τον τύπο του ListView Item (Αν είναι να φέρει Παρατηρήσεις ή Προϊόντα ας πούμε) και στο ButtonText θα αποθηκεύουμε το ID της εγγραφής.
5. Ειδοποίηση του ListView Πως τελείωσε η ενημέρωση έτσι ώστε να ανανεώσει το οπτικό του κομμάτι.

6.5.5 Threads

Σύμφωνα με την εξέλιξη των μοντέρνων φορητών συσκευών, πολλές από αυτές λειτουργούν με επεξεργαστές που υποστηρίζουν πολυ-νηματικές εφαρμογές. Η εφαρμογή νημάτων στην εφαρμογή μας είναι αρκετά σημαντική έτσι ώστε το γραφικό μας περιβάλλον να είναι πολύ πιο άμεσο στις αντιδράσεις του. Αν εξαιρέσουμε την αποδοτικότητα που προσφέρουν τα threads και την καλύτερη εμπειρία πλοήγησης, είναι ένα πολύ βασικό feature για την ομαλή συμπεριφορά της εφαρμογής μας. Σε περίπτωση που η εφαρμογή μας προσπαθήσει να εκτελέσει μια εργασία η οποία θα πάρει αρκετή ώρα απασχολώντας το κύριο νήμα, τότε σε μερικά δευτερόλεπτα το λειτουργικό σύστημα της πλατφόρμας θα δείξει μήνυμα πως η εφαρμογή δεν ανταποκρίνεται. Τις περισσότερες φορές σε μια τέτοια περίπτωση ο χρήστης θα νομίζει πως η εφαρμογή κόλλησε και θα προχωρήσει στην επανεκκίνησή της.

Είναι δεδομένο για τις μοντέρνες πλατφόρμες να εκτελούν τις χρονοβόρες διαδικασίες σε δευτερεύον νήματα καθώς το κύριο νήμα μένει απασχολημένο ανανεώνοντας την κίνηση κάποιου Loading Indicator και ανά στιγμές να αναλαμβάνει την ανανέωση των ορατών γραφικών της εφαρμογής μας.

Το κύριο νήμα στο οποίο τρέχει η εφαρμογή είναι το μόνο το οποίο μπορεί με ασφαλή τρόπο να "ζωγραφίζει" στο ορατό γραφικό περιβάλλον. Σε περίπτωση που παραβούμε τον κανόνα αυτόν τότε είναι αρκετά πιθανό να καταλήξουμε σε Access Violation το οποίο θα διακόψει βίαια την ροή της εφαρμογής μας .

Τα παραπάνω προϋποθέτουν κατάλληλη δομή προγραμματισμού ώστε να αποφύγουμε περιπτώσεις για Access Violation και να προστατέψουμε τον χρήστη από το να επηρεάσει την ροή των νημάτων. [13]

Έτσι λοιπόν έχοντας στο νου μας την ροή με την οποία λειτουργούν οι φόρμες όπως εξηγήσαμε προηγουμένως, θα κάνουμε κάποιες τροποποιήσεις έτσι ώστε να μπορούν να λειτουργήσουν σωστά και με threads.

1. Καθαρισμός της λίστας μας σε περίπτωση που έχει αντικείμενα μέσα και ορισμός του visibility της ως False και ειδοποίηση πως θα το ενημερώσουμε.
2. Δημιουργία του παράπλευρου Thread.
3. Εμφάνιση μηνύματος που δηλώνει φόρτωση δεδομένων και ενημερώνει τον χρήστη πως πρέπει να περιμένει. Το μήνυμα αυτό θα περιέχει και έναν Loading Indicator ο οποίος θα ενεργοποιείται.

4. Θέτουμε μια Global μεταβλητή Working ως true η οποία θα "κλειδώνει" την δυνατότητα πλοήγησης του χρήστη μέχρι να ολοκληρωθεί η εργασία.
5. Εκκίνηση του thread το οποίο αναλαμβάνει την εκτέλεση της κατάλληλης μεθόδου για την λήψη των δεδομένων και αναθεσής παρέα με την δημιουργία των listview items.
6. Ειδοποίηση του ListView πως τελείωσε η ενημέρωση και θέτει την ορατότητά του ως True.
7. Θέτουμε την Global μεταβλητή Working ως false για να επιτρέψουμε στον χρήστη να πλοηγηθεί ξανά ελεύθερα.

6.5.6 Slide Transitions

Όταν κάποια δράση του χρήστη τον μεταφέρει σε κάποια άλλη φόρμα, η μετάβασή της σε αυτήν θα πρέπει να γίνεται με συγκεκριμένο οπτικό τρόπο. Αν ο χρήστης ανοίγει μια νέα φόρμα τότε αυτή θα πρέπει να εμφανίζεται συρτά από τα δεξιά της οθόνης προς τα αριστερά. Ενώ όταν ο χρήστης πηγαίνει προς τα πίσω τότε η επαναφορά της προηγούμενης φόρμας θα πρέπει να ξεκινάει από τα αριστερά και να έρχεται προς τα δεξιά.

Αυτό επιτυγχάνεται χρησιμοποιώντας Actions με Slide Transition ανάμεσα στα Tabs. Οι περισσότερες εναλλαγές εξελίσσονται ανάμεσα στα αόρατα tabs στα οποία βρίσκεται το μενού, και τα 2 tabs που περιέχουν την φόρμα με τα φίλτρα και την κενή φόρμα που εξυπηρετεί ως container για τις φόρμες προς δημιουργία.

6.5.7 Exception Handling

Ένα τελευταίο σημαντικό μέρος κάθε εφαρμογής το οποίο βοηθάει στην ομαλή ροή λειτουργίας της είναι το Exception Handling. Έχουμε προβλέψει κάποιες περιπτώσεις στις οποίες κάτι μπορεί να μην πάει καλά και έτσι έχουμε θέσει κατάλληλα μηνύματα σε κάθε είδους σφάλμα έτσι ώστε ο χρήστης να ξέρει τι συμβαίνει.

Οι φορητές συσκευές αρκετές φορές χρησιμοποιούν συνδέσεις στο internet οι οποίες δεν χαρακτηρίζονται για την σταθερότητά τους ή την ταχύτητά τους. Σε περίπτωση λοιπόν όπου ο χρήστης ορίσει ένα αφύσικο εύρος ημερομηνιών για την λήψη πληροφοριών, μετά από ένα ορισμένο χρονικό διάστημα η σύνδεση θα κάνει Timeout το αίτημα. Μέσω του Exception Handling μπορούμε στο Timeout να τροφοδοτήσουμε τον χρήστη με ένα φιλικό μήνυμα έτσι ώστε να καταλάβει πως η σύνδεση δεν είναι αρκετά γρήγορη ή πως ο όγκος δεδομένων που ζήτησε είναι αφύσικα ογκώδης.

Επίσης σε περίπτωση αδυναμίας επικοινωνίας με τον Datasnap Server τροφοδοτεί τον χρήστη με κατάλληλο μήνυμα και τον επαναφέρει στην οθόνη του login έτσι ώστε να ανανεώσει και το connection string του.

7 Στάδια Υλοποίησης

7.1 Project Management

Για αρχή ερευνήσαμε τις πληροφορίες που μπορούμε να αντλήσουμε από την Βάση Δεδομένων την οποία χρησιμοποιεί το ERP. Διαλέξαμε κάποιες οι οποίες παρείχαν πληροφορίες εύπεπτες για να μπορεί να δει κάποιος μέσα από μια φορητή εφαρμογή. Καταλήξαμε στις επιλογές που βάλαμε τελικά στο μενού μας και αναλύσαμε στην αρχή της Εργασίας. Η διαδικασία αυτή μας πήρε 2 μέρες.

Στην συνέχεια ερευνήσαμε ποια προγραμματιστική πλατφόρμα θα μπορούσε να μας βοηθήσει περισσότερο στην υλοποίηση μιας εφαρμογής για πολλές πλατφόρμες. Καταλήξαμε στο RAD Studio της Embarcadero καθώς υπήρχε ήδη εμπειρία με την γλώσσα προγραμματισμού Delphi. Η διαδικασία αυτή μας πήρε 3 μέρες.

Ερευνήσαμε τον τρόπο με τον οποίο υλοποιούνται φορητές εφαρμογές μέσα στο RAD Studio με τις τεχνολογίες που παρέχει και προχωρήσαμε στο να δούμε ποιες από αυτές θα ταίριαζαν καλύτερα στην δική μας περίπτωση. Επιλέξαμε τον Rest Datasnap Server και το Firemonkey Framework ως τεχνολογίες του RAD Studio. Η διαδικασία αυτή πήρε μία εβδομάδα.

Ερευνήσαμε τεχνικές ανάπτυξης γραφικών περιβαλλόντων τις οποίες κάθε πλατφόρμα υπαγορεύει για τις εφαρμογές που επιτρέπει να χρησιμοποιούνται. Αποφασίζοντας το μοτίβο στο οποίο θα δομηθεί το Client κομμάτι της εφαρμογής μας, κάναμε αρκετά Test Projects ώστε να δούμε με ποιο τρόπο μπορούμε να το υλοποιήσουμε όσο πιο αποδοτικά γίνεται. Η διαδικασία αυτή πήρε δύο εβδομάδες.

Έχοντας μαζέψει τις πληροφορίες που χρειάζεται και ελέγχοντας αν είναι αποδοτικές ξεκινήσαμε να φτιάχνουμε πρώτα τον DataSnap Server υλοποιώντας όλες τις μεθόδους τις οποίες θα εξέθετε στην Client Εφαρμογή μας. Η διαδικασία αυτή πήρε δύο εβδομάδες.

Τελειώνοντας τον Datasnap Server, ξεκινήσαμε την δημιουργία του Firemonkey Client Application. Το προγραμματιστικό κομμάτι έγινε παράλληλα με το γραφικό κομμάτι το οποίο δοκιμαζόταν σε αρκετές διαφορετικές οθόνες έτσι ώστε να σιγουρευτούμε το ότι η εφαρμογή θα φαίνεται όπως πρέπει παντού. Η διαδικασία αυτή πήρε δύο μήνες.

Η δημιουργία των εικαστικών για την εφαρμογή, δηλαδή η επιλογή χρωμάτων και η σύνθεση εικόνων καθώς και η βελτιστοποίηση της τοποθεσίας των κουμπιών πήρε δύο

εβδομάδες. Υπήρξε δυσκολία να δημιουργηθεί σωστά το γραφικό περιβάλλον έτσι ώστε να φαίνονται όλα όπως πρέπει σε όλες τις συσκευές.

Οι μετρήσεις αυτές, ειδικά κατά την ανάπτυξη του Datasnap Server και της φορητής εφαρμογής είναι χονδρικές καθώς λόγω ελλιπούς έρευνας και πηγών υπήρξαν αρκετά πισωγυρίσματα και αλλαγή τεχνικών υλοποίησης.

7.2 Testing

Το Τεστάρισμα της εφαρμογής πήρε 3 εβδομάδες παρέα με βελτιστοποιήσεις και διορθώσεις των bugs των οποίων βρέθηκαν στην εφαρμογή. Υπήρχαν αρκετά προβλήματα στον τομέα του γραφικού περιβάλλοντος ανάμεσα στις διαφορετικές συσκευές. Για το μεγαλύτερο μέρος της ανάπτυξης της εφαρμογής το Compile γινόταν για πλατφόρμα Win32 έτσι ώστε να κερδίζουμε χρόνο. Όταν γινόταν compile σε φορητή συσκευή Android το Compile και το Deployment συνολικά έπαιρναν γύρω στο ένα με δύο λεπτά. Σε περιπτώσεις όπου υπήρχαν προσπάθειες "στρωσίματος" κάποιων σφαλμάτων και χρειαζόταν να γίνονται compiles πολύ συχνά τότε η διαδικασία ήταν απίστευτα χρονοβόρα και μη αποδοτική.

Η ανάπτυξη σε πλατφόρμα Win32 όμως δημιούργησε με την σειρά της άλλα απρόοπτα προβλήματα. Συναντήσαμε περιπτώσεις όπου οι φορητές πλατφόρμες συμπεριφέρονταν διαφορετικά σε κάποια στοιχεία της γλώσσας σε σχέση με την πλατφόρμα των Win32. Σε κάποιες περιπτώσεις κατά το γέμισμα των ListViews ερχόμασταν αντιμέτωποι με Access Violations στις φορητές συσκευές τα οποία δεν μπορούσαμε να δικαιολογήσουμε μέχρι που παρατηρήσαμε ότι στις φορητές συσκευές η χρήση μη αρχικοποιημένων μεταβλητών μέσα σε ελέγχους απαγορεύονταν.

Ένα άλλο θέμα το οποίο αντιμετωπίσαμε ήταν το Trimming το οποίο συνέβαινε ανάμεσα στα Elements ενός ListView Item. Τα Default μεγέθη γραμματοσειρών σε Android και iOS ήταν διαφορετικά σε πραγματικές συσκευές και χρειάζονταν συγκεκριμένες ρυθμίσεις για το καθένα καθώς υπήρχαν περιπτώσεις όπου το Text κάλυπτε το πεδίο Detail μέσα σε ένα ListView Item.

Για το τεστάρισμα της εφαρμογής σε πλατφόρμα Android χρησιμοποιήθηκε για αρχή ο Android Emulator της Google αλλά κυρίως χρησιμοποιήθηκε η συσκευή Samsung Galaxy Tab II 7.0 καθώς και οι συσκευές Samsung S3 mini και Samsung S4 για να δούμε πώς η συσκευή συμπεριφέρεται σε ανάμεικτους συνδυασμούς μεγέθους εικόνας και ανάλυσης. Ήταν αδύνατο να χρησιμοποιηθεί Emulator Android διότι χρειαζόμασταν ένα τον οποίο να μιμείται την

αρχιτεκτονική επεξεργαστή ARM. Η μίμηση επεξεργαστή ARM όμως από καθημερινούς επεξεργαστές δεν είναι καθόλου αποδοτική με αποτέλεσμα ο Emulator να είναι υπερβολικά αργός ώσπου σε κάποιο σημείο δεν ήταν καν δυνατό να κάνουμε compile την εφαρμογή μας όταν ξεκίνησε να μεγαλώνει.

Για το τεστάρισμα της εφαρμογής σε πλατφόρμα iOS χρησιμοποιήθηκε MacBook και iPhone 4. Για αρχή χρησιμοποιήθηκε ο Emulator του iOS για πιο γρήγορο τεστάρισμα αλλά εντοπίστηκε διαφορετική συμπεριφορά από την πραγματική συσκευή ως προς τον "ορισμό" και την μέτρηση των pixels, πράγμα το οποίο μπέρδευε.

7.3 Χώρος για Βελτιώσεις

Λόγω χρόνου, υπήρξαν κάποια πράγματα ως εργαλεία τα οποία δεν αξιοποιήθηκαν στην εκπόνηση της εργασίας αλλά θα μπορούσαν να υλοποιηθούν στο μέλλον έτσι ώστε η εφαρμογή να είναι πιο αποδοτική.

Ένα από τα πράγματα τα οποία θα πρέπει να γίνει στο μέλλον είναι να μετατραπεί ο Datasnap Server σε ISAPI έτσι ώστε να μπορεί να φορτωθεί ως Service πάνω στον IIS Web Server τον οποίο έχουν τα Windows. Αυτή τη στιγμή με τον τρέχον τρόπο υλοποίησης του Datasnap Server, σε περίπτωση που η λειτουργία του .exe μας διακοπεί για οποιονδήποτε λόγο, θα πρέπει χειροκίνητα κάποιος System Administrator να ξανατρέξει το .exe του Datasnap Server. Αυτό σημαίνει πως και για να σιγουρευτούμε πως ο Datasnap Server λειτουργεί όπως πρέπει να τον τσεκάρουμε καθημερινά. Μετατρέποντας τον Datasnap Server από .exe σε service τότε ο IIS μπορεί αυτόματα να φροντίζει την αδιάκοπη λειτουργία του, καθώς και το να ξεκινάει αυτόματα κατά την εκκίνηση του υπολογιστή στον οποίο το έχουμε στήσει.

Ένα άλλο χαρακτηριστικό το οποίο θα ήταν αρκετά σημαντικό ως προς την οικονομία του Bandwith και της εταιρείας αλλά και των χρηστών που χρησιμοποιούν δίκτυο 3G ως μέσω σύνδεσης στο internet, είναι το Compression. Όταν μια μέθοδος του Datasnap Server τραβάει δεδομένα από την Βάση Δεδομένων και τα πακετάρει ως DataSet για να τα στείλει στην φορητή εφαρμογή θα μπορούσαμε να χρησιμοποιήσουμε την βιβλιοθήκη Gzip για να συμπιέζουμε τις πληροφορίες αυτές. Η διαδικασία αυτή θα έριχνε περισσότερο βάρος στον Server καθώς θα δούλευε περισσότερο αλλά όμως η ισχύς αυτή είναι αμελητέα μπροστά στο κόστος το οποίο γλιτώνεται στο bandwith. Ο λόγος που δεν υλοποιήθηκε αυτό ήταν ότι θα χρειαζόταν να τροποποιήσουμε όλες τις μεθόδους έτσι ώστε να δημιουργούν Streams αντί για DataSet, τα οποία είναι συμπιέσιμα και να επιστρέφουν αυτά. Τέλος θα έπρεπε να τροποποιηθούν και στο Client

κομμάτι της εφαρμογής μας όλα τα καλέσματα των μεθόδων ώστε να λαμβάνουν το stream και να το αποσυμπιέζουν για να το χρησιμοποιήσουμε.

Υπάρχει άλλο ένα χαρακτηριστικό το οποίο έχουν πολλές φορητές εφαρμογές οι οποίες εμφανίζουν δεδομένα σε λίστες αλλά υπάρχει και στα sites τα οποία επισκεπτόμαστε καθημερινά. Αυτό το χαρακτηριστικό είναι παρόμοιο με το Paging ή το Lazy Loading. Δηλαδή να υπάρχει η δυνατότητα η μέθοδος του Datasnap Server να κομματιάζει το πακέτο που θα στείλει στον χρήστη και να στέλνει μόνο το πρώτο. Αφού ο χρήστης κυλήσει στο τέλος του πακέτου να εμφανίζεται επιλογή "Εμφάνιση περισσότερων" η οποία να ζητάει από τον Datasnap Server το επόμενο πακέτο πληροφοριών. Αυτό είναι ένα χαρακτηριστικό το οποίο θα ήταν ιδιαίτερο ως προς την υλοποίηση λόγω της τεχνολογίας REST που χρησιμοποιούμε όπου δεν δημιουργούνται sessions.

Υπάρχουν και άλλες λειτουργίες του ERP μας οι οποίες θα μπορούσαν να αξιοποιηθούν στο μέλλον από την φορητή μας εφαρμογή όπως είναι για παράδειγμα Ευρετήρια Πελατών και Προμηθευτών. Πέρα από την απλή προβολή στοιχείων Πελατών ή Προμηθευτών θα μπορούσαμε να υλοποιήσουμε και δυνατότητα άμεσης κλήσης μέσω της εφαρμογής μας καθώς το Firemonkey δίνει την δυνατότητα να κάνουμε κλήσεις στις λειτουργίες του Android/iOS API. Αυτό βέβαια θα καθιστούσε την εφαρμογή μας διαθέσιμη μόνο σε κινητές συσκευές οι οποίες έχουν την δυνατότητα να πραγματοποιήσουν κλήσεις το οποίο δεν είναι καλό. Θα μπορούσαμε αντί για άμεση κλήση να δώσουμε την δυνατότητα "αντιγραφής" του στοιχείου που θέλουμε για να μπορούμε να το επικολλήσουμε οπουδήποτε θέλουμε.

Μια άλλη δυνατότητα που θα μπορούσαμε να προσθέσουμε είναι η τροποποίηση των πληροφοριών στις οποίες έχουμε πρόσβαση μέσω της εφαρμογής. Θα μπορούσαμε να δημιουργήσουμε πολλαπλούς χρήστες για κάθε διαθέσιμη λειτουργία και να περιορίζουμε το τι μπορεί να βλέπει/τροποποιεί κανείς. Αυτό βέβαια προϋποθέτει να δημιουργηθούν και άλλες λειτουργίες οι οποίες επιτρέπουν να γίνεται κανονικά εργασία μέσω της φορητής μας εφαρμογής. Τέλος, οι λειτουργίες τροποποίησης θα μπορούσαν να εμφανίζονται κυρίως σε συσκευές οι οποίες έχουν μεγάλη οθόνη καθώς θα καθιστούσε πολύ πιο άνετη την χρήση τους.

8 Συμπεράσματα

Ο στόχος της πτυχιακής εργασίας εκπληρώθηκε καθώς η εφαρμογή κάνει όλα όσα χρειαστήκαμε και έγινε αποδεκτή και στο app store της Apple αλλά και στο Google Play. Η ταυτόχρονη δημιουργία εφαρμογών για πολλαπλές πλατφόρμες είναι δυνατός να υλοποιηθεί με τις τεχνολογίες τις οποίες επιλέξαμε και επιτεύχθηκε η οικονομία πόρων από την εταιρεία.

Από την εμπειρία που κερδίσαμε κατά τη διάρκεια εκπόνησης της πτυχιακής εργασίας φτάσαμε στο συμπέρασμα πως η χρήση εργαλείων για ταυτόχρονη ανάπτυξη εφαρμογών για πολλαπλές πλατφόρμες είναι χρήσιμη σε περιπτώσεις όπου θέλουμε η υλοποίηση του έργου μας να γίνει όσο πιο γρήγορα γίνεται. Αυτό όμως είχε τα μειονεκτήματά του ως προς την απόδοση της εφαρμογής στην κατανάλωση πόρων και ως προς την συνολική εμφάνιση της εφαρμογής. Σε περίπτωση που θα θέλαμε να αναπτύξουμε εφαρμογές οι οποίες θα είχαν μεγαλύτερη συμβατότητα ως προς τις συσκευές ή να περιέχουν πλούσια γραφικά και smooth animations ίσως να ήταν καλύτερα να κινηθούμε σε άλλες τεχνολογίες ανάπτυξης λογισμικού για φορητές συσκευές.

Ένα άλλο εφόδιο το οποίο αποκομίσθηκε μέσα από την εργασία αυτή είναι η εμπειρία ανάπτυξης εφαρμογής σε παραθυρικό περιβάλλον για πρώτη φορά. Η εμπειρία προγραμματισμού σε παραθυρικές εφαρμογές και ειδικότερα σε εφαρμογές κινητών είναι ιδιαίτερα πολύτιμες στις μέρες μας καθώς είναι οι τεχνολογίες που χρησιμοποιούνται περισσότερο στις μέρες μας. Επίσης μάθαμε πως το να χρησιμοποιούμε εργαλεία τα οποία είναι πολύ καινούργια δεν είναι πάντα ο πιο εύκολος δρόμος κρίνοντας μόνο από τις λειτουργίες τις οποίες προσφέρουν καθώς οι πηγές για την σωστή αξιοποίησή τους θα είναι δυσεύρετη. Ακόμη και σήμερα δεν υπάρχουν βιβλία τα οποία έχουν γραφτεί με κύρια θεματολογία τους το Firemonkey. Χρησιμοποιήθηκαν όμως βιβλία για την εκμάθηση των βασικών στοιχείων της Delphi τα οποία είναι σχεδόν ίδια και εκεί.

Επίσης για άλλη μια φορά καταλάβαμε πόσο σημαντικός είναι ο καλός σχεδιασμός αρχιτεκτονικής του έργου που υλοποιούμε. Η περιορισμένη διορατικότητα λόγω έλλειψης εμπειρίας μας ανάγκασε να κάνουμε πισωγυρίσματα στον τρόπο υλοποίησης κάποιων μερών του έργου το οποίο κόστισε πολύ χρόνο. Επιπλέον, τεχνικές που χρησιμοποιήθηκαν λάθος στιγμή στο project ήταν δύσκολο να χωρέσουν λόγω του τρόπου με τον οποίο είχαν ήδη υλοποιηθεί κομμάτια του έργου. Το πιο τρανταχτό παράδειγμα πάνω σε αυτό ήταν η υλοποίηση των Threads.

9 Ηλεκτρονικές Πηγές

[1] DocWiki Embarcadero. PAServer, the Platform Assistant Server Application. [Online]. http://docwiki.embarcadero.com/RADStudio/XE5/en/PAServer,_the_Platform_Assistant_Server_Appli n

[2] Marco Cantu, *Development and Deployment of Delphi Multi-tier Applications.* - http://www.embarcadero.com/rad-in-action/development-and-deployment-of-delphi-multi-tier-applications, Δεκέμβριος 2012.

[3] W3C, SOAP Documentation - http://www.w3.org/TR/soap/

[4] Roy Fielding, *Architectural Styles and the Design of Network-based Software Architectures (Ph.D.).* http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[5] Marco Cantu, The Delphi Language for Mobile Development., Απρίλιος 2013. [Online]. http://www.embarcadero.com/resources/white-papers?download=102

[6] Pawel Glowacki, Delphi Labs: Datasnap XE - Authentication and Authorization - http://edn.embarcadero.com/article/41267

[7] Sarina DuPont, Designing Common User Interfaces for iOS & Android - https://www.youtube.com/watch?v=Oypth7HCjZM

[8] Stack Overflow, Handle Hardware Back Button In Delphi XE5 Firemonkey On Android - http://stackoverflow.com/questions/18774408/how-do-i-handle-a-back-button-press-in-a-delphi-andro app

[9] Freeing and Disposing Dynamically created forms problems. http://www.codenewsfast.com/cnf/thread/0/permalink.thr-ng2027q14159

[10] Stack Overflow, Dynamically create form by name - http://stackoverflow.com/questions/14004373/dynamically-create-form-by-name

[11] Effectively Using List Controls in Mobile Apps - https://www.youtube.com/watch?v=XRj3qjUjBlc&list=PLwUPJvR9mZHiaYvH9Xr7WuFCVYugC4d0w&inde 23

[12] Embarcadero DocWiki, Using Multi-Resolution Bitmaps - http://docwiki.embarcadero.com/RADStudio/XE5/en/Using_Multi-Resolution_Bitmaps

[13] Zarko Gajic. (2014) delphi.about.com. [Online]. http://delphi.about.com/od/kbthread/a/thread-gui.htm

10 Βιβλιογραφία

Coding in Delphi , Nick Hodges - http://www.amazon.com/Coding-Delphi-Nick-Hodges/dp/1941266037

Delphi 2010 Handbook, Marco Cantu - http://www.marcocantu.com/dh2010/

Delphi Cookbook, Daniele Teti - https://www.packtpub.com/application-development/delphi-cookbook

Delphi XE2 Foundations, Chris Rolliston - http://www.amazon.com/Delphi-XE2-Foundations-Chris-Rolliston/dp/1477550895/ref=sr_1_3?s=books&ie=UTF8&qid=1421097105&sr=1-3