ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Δυναμική Διεπαφή Χρήστη Για Διαχείριση Απομακρυσμένης Βάσης Δεδομένων Σε Android

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

των

ΒΑΣΙΛΕΙΟΥ-ΚΑΛΦΑ ΔΗΜΗΤΡΙΟΥ **ΚΑΤΣΑΤΟΥ ΙΩΑΝΝΗ**

Επιβλέπων : **Γ**εώργιος Στασινόπουλος
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2014

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΈΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Δυναμική Διεπαφή Χρήστη Για Διαχείριση Απομακρυσμένης Βάσης Δεδομένων Σε Android

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

των

ΒΑΣΙΛΕΙΟΥ-ΚΑΛΦΑ ΔΗΜΗΤΡΙΟΥ **ΚΑΤΣΑΤΟΥ ΙΩΑΝΝΗ**

Επιβλέπων : Γεώργιος Στασινόπουλος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 18η Ιουλίου 2014.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γεώργιος Στασινόπουλος Καθηγητής Ε.Μ.Π.	Ε. Δ. Συκάς Καθηγητής Ε.Μ.Π.	Μ. Ε. Θεολόγου Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2014

(Υπογραφή)

.....

ΒΑΣΙΛΕΙΟΥ-ΚΑΛΦΑΣ ΔΗΜΗΤΡΙΟΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

(Υπογραφή)

.....

ΚΑΤΣΑΤΟΣ ΙΩΑΝΝΗΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η δημιουργία δυναμικά δημιουργούμενης διεπαφής χρήστη με στόχο την διαχείριση όποιας απομακρυσμένης βάσης δεδομένων του δηλώσουμε ως είσοδο. Στην ουσία μιλάμε για την κατασκευή ενός δυναμικού διαχειριστικού της απομακρυσμένης βάσης δεδομένων σχεδιασμένο και υλοποιημένο σε Android περιβάλλον.

Πιο συγκεκριμένα, η εφαρμογή μας αποτελείται από δύο υποσυστήματα:

- Το Client Side υποσύστημα (Android Εφαρμογή) το οποίο απεικονίζει με γραφικό τρόπο την δομή οποιαδήποτε απομακρυσμένης βάσης δεδομένων και ακόμη δίνει την δυνατότητα στον χρήστη με απλό τρόπο να επιτελέσει τις διάφορες λειτουργίες (π.χ. εισαγωγή, διαγραφή δεδομένων κ.α) που επιθυμεί σε αυτή.

- Το Server Side υποσύστημα το οποίο στην ουσία είναι μία και μόνο απλή διεπαφή με την απομακρυσμένη Βάση Δεδομένων.

Για τους σκοπούς της διπλωματικής μας αναπτύξαμε πιο πολύ το κομμάτι της εισαγωγής δεδομένων στα διάφορα Tables της απομακρυσμένης Βάσης Δεδομένων. Παρόλα αυτά, το σημαντικότερο είναι ότι θέσαμε τα θεμέλια και την όλη μεθοδολογία έτσι ώστε να είναι εφικτό, με πολύ λίγο κόπο, η εφαρμογή να να είναι δυνατόν να εξελιχθεί και να αναπτυχθεί σε ένα ολοκληρωμένο διαχειριστικό περιβάλλον της απομακρυσμένης Βάσης Δεδομένων.

Λέξεις Κλειδιά: Android, Εφαρμογή κινητού, Εξυπηρετητής Βάσης Δεδομένων, Σχήμα Βάσης Δεδομένων, Διαχειριστικό Βάσης Δεδομένων.

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

The scope of this thesis was the design and development of a dynamically created Graphical User Interface (GUI) which could manage - a pointed as input- remote database. Basically, it is created a dynamically created administration tool of the remote database, designed and developed for the Android environment. More specifically , our application consists of two subsystems:

- The Client Side subsystem (Android Application) which illustrates graphically the structure of any remote database pointed to and additionally gives users the opportunity to apply specific actions to the database (such as record insert, delete , update etc.) in a simple and user friendly way.

- The Server Side subsystem which is actually a single interface of the remote database.

For the purpose of this thesis, we focused our efforts in developing only the "insert record" action, for the tables of the remote database. But, the most important thing is that we designed and delivered a whole infrastructure and the methodology in order our application to be transformed, with minimal effort, into a complete remote database administration environment.

Keywords: Android, Mobile App, Database Server, Database Schema, Database Administration Tool. .

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

1 Εισαγωγή....1

1.1 <Τίτλος που έχει σχέση με τον γενικότερο χώρο εφαρμογής της διπλωματικής>....1

1.2 Αντικείμενο διπλωματικής....1

1.2.1 Συνεισφορά....1

1.3 Οργάνωση κειμένου....2

2 Σχετικές εργασίες....3

2.1 <Τίτλος για σχετική θεματική περιοχή 1>....3

2.2 <Τίτλος για σχετική θεματική περιοχή 2>....3

3 Θεωρητικό υπόβαθρο....4

3.1 <Τίτλος τεχνικής, μεθοδολογίας, μοντέλου 1>....4

3.2 <Τίτλος τεχνικής, μεθοδολογίας, μοντέλου 2>....4

4 Ανάλυση Απαιτήσεων Συστήματος....5

4.1 Αρχιτεκτονική 5

4.2 Περιγραφή Λειτουργιών....6

4.2.1 Client Side υποσύστημα....6

4.2.2 Server Side υποσύστημα....11

4.3 Μοντέλο Οντοτήτων Συσχετίσεων....11

5 Σχεδίαση Συστήματος....12

5.1 Αρχιτεκτονική 12

5.1.1.2 ServerConnector....14

5.2 Περιγραφή Κλάσεων....16

5.2.1 <Τίτλος κλάσης 1>....16

5.2.2 <Τίτλος κλάσης 2>....16

5.3 Βάση Δεδομένων....16

5.4 Κωδικοποίηση αρχείων....16

6 Υλοποίηση.....17

6.1 Λεπτομέρειες υλοποίησης.....17

6.1.1 <Τίτλος θέματος 1>.....17

6.1.2 <Τίτλος θέματος 2>.....17

6.2 Πλατφόρμες και προγραμματιστικά εργαλεία 17

7 Έλεγχος 19

7.1 Μεθοδολογία ελέγχου.....19

7.2 Αναλυτική παρουσίαση ελέγχου.....19

8 Επίλογος 20

8.1 Σύνοψη και συμπεράσματα.....20

8.2 Μελλοντικές επεκτάσεις.....20

9 Βιβλιογραφία.....21

1 Ανάλυση Απαιτήσεων Συστήματος

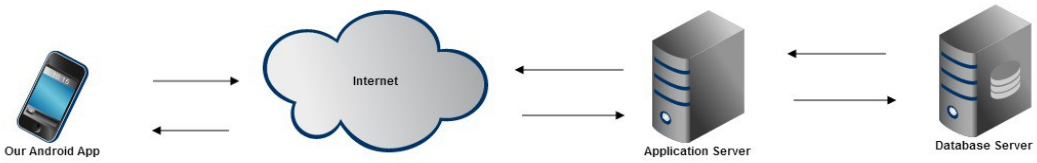
Στο κεφάλαιο αυτό ακολουθεί η περιγραφή της αρχιτεκτονικής του συστήματος και θα γίνει η ανάλυση απαιτήσεων για τις λειτουργίες του.

1.1 Αρχιτεκτονική

Το σύστημα μας, αποτελείται από δύο βασικά επιμέρους κομμάτια (υποσυστήματα). Το Client Side υποσύστημα (το οποίο είναι η εφαρμογή Android) και το Server Side υποσύστημα.

Το Client Side κομμάτι, στην ουσία, αποτελεί μία διεπαφή μεταξύ του τελικού χρήστη και της απομακρυσμένης Βάσης Δεδομένων. Όταν ο χρήστης θέλει να πραγματοποιήσει κάποια ενέργεια στην απομακρυσμένη Βάση, δεν έχει παρά να υλοποιήσει την ενέργεια αυτή στο UI περιβάλλον του Android και αυτό με την σειρά του θα την μεταφέρει στην απομακρυσμένη Βάση Δεδομένων.

Το Server Side υποσύστημα, από την άλλη, αποτελεί μία διεπαφή μεταξύ της Android εφαρμογής και της Βάσης Δεδομένων. Όταν η Android εφαρμογή θέλει να επιτελέσει κάποια λειτουργία στην Βάση Δεδομένων, μεταφέρει το αίτημα της στον Server και αυτός με την σειρά του (γνωρίζοντας πως) το μεταφέρει στην Βάση Δεδομένων.



1.2 Περιγραφή Λειτουργιών

Εδώ περιγράφουμε τις λειτουργίες που απαιτείται να εκτελεί το σύστημα. Αφού έχουμε χωρίσει ήδη το σύστημά μας σε υποσυστήματα, περιγράφουμε το κάθε υποσύστημα ξεχωριστά. Κάθε περιγραφή υποσυστήματος έχει 2 σκέλη: κείμενο και σχήμα. α) Για το κείμενο: με απλά λόγια και πραγματικά παραδείγματα όπου χρειάζεται, πρέπει να εξηγήσουμε στο χρήστη τι είναι αυτό που κάνει το υποσύστημα. Καλό είναι οι εξηγήσεις να δίνονται σύντομα, περιεκτικά και αριθμημένα, π.χ. 1. αυτό, 2. το άλλο, κ.λ.π.. β) Για το σχήμα: μετά από το κείμενο, έχουμε και σχετικό Διάγραμμα Ροής Δεδομένων (DFD) με το οποίο δίνουμε και σχηματικά τί κάνει το υποσύστημα.

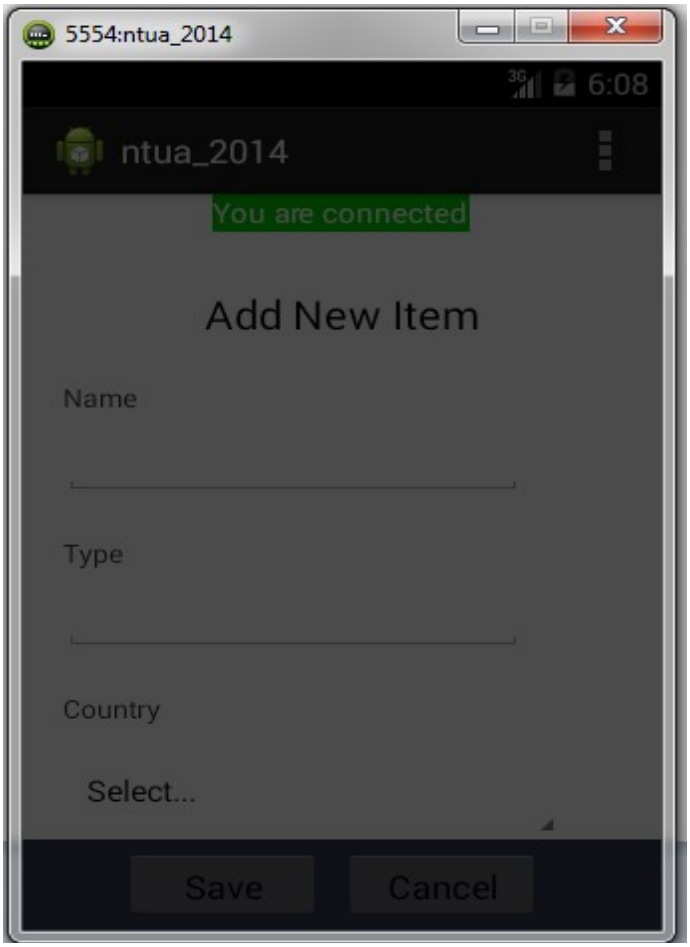
Είναι πολύ βασικό να κατανοήσετε ότι εδώ περιγράφουμε **τί κάνει** το σύστημα **και ΟΧΙ πώς** το κάνει. Για το λόγο αυτό, να χρησιμοποιείτε και αρκετά παραδείγματα.

1.2.1 Client Side υποσύστημα

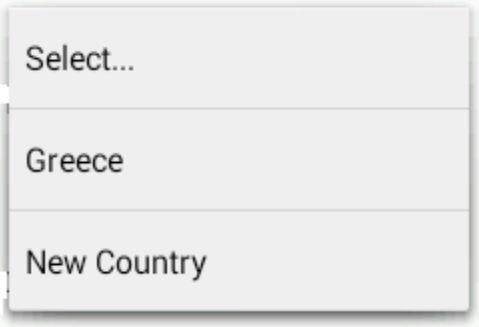
Η παρούσα υλοποίηση της Android εφαρμογής, παρέχει στον χρήστη μόνο την δυνατότητα εισαγωγής δεδομένων στην Βάση. Με λίγα λόγια, η κάθε οθόνη της εφαρμογής αναπαριστά με μορφή UI και έναν πίνακα της βάσης. Με το που εκκινεί η εφαρμογή οδηγεί τον χρήστη στην Edit οθόνη του default πίνακα της Βάσης. Για το παράδειγμα μας, ο default αυτός πίνακας είναι ο **Item,** ο οποίος διαθέτει μία σχέση πολλά προς ένα με τον πίνακα **Country.**

Οι σχέσεις των πινάκων της βάσης, στις οθόνες της Android εφαρμογής απεικονίζονται με την βοήθεια dropdown λιστών. Έτσι, για το παράδειγμα μας, η σχέση του πίνακα Item με τον πίνακα Country εκφράζεται με την παρουσία μίας dropdwon λίστας που περιέχει τις τιμές του πίνακα Country.

Στην συνέχεια ακολουθούν μερικές εικόνες της Android εφαρμογής, οι οποίες απεικονίζουν καλύτερα αυτά που μόλις είπαμε πιο πάνω.



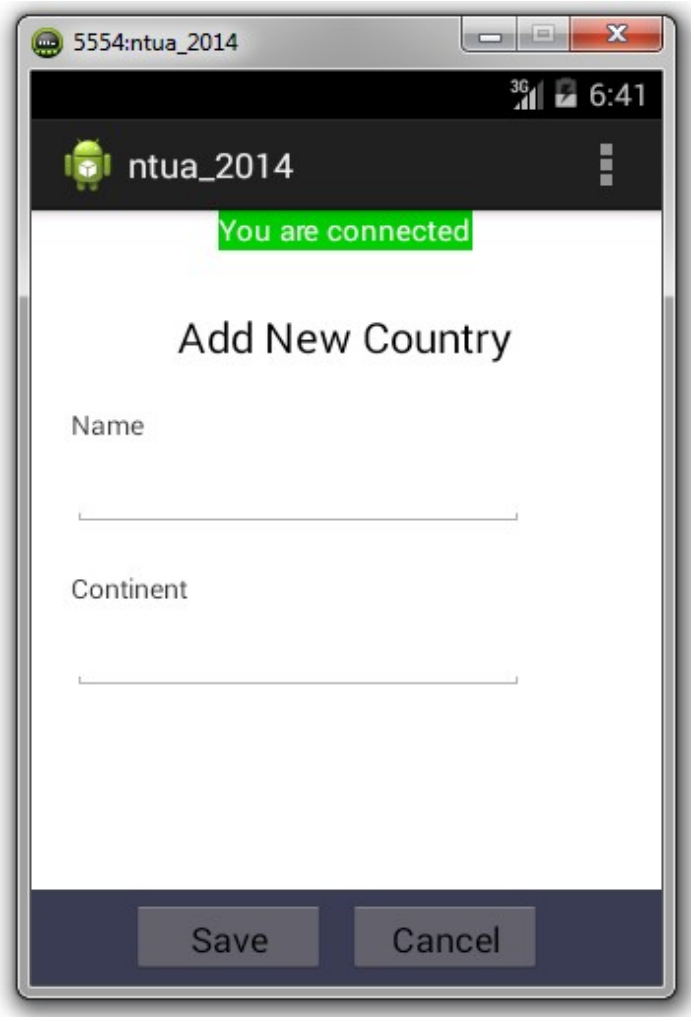
Σχήμα 1.2 Η default οθόνη της Android Εφαρμογής.



Σχήμα 1.3 Η dropdown λίστα, με τις τιμές του πίνακα Country.

Όπως παρατηρούμε, Σχήμα 1.3, η dropdown λίστα εκτός από τα περιεχόμενα του πίνακα που αναπαριστά (τον Counrty στο παράδειγμα μας), περιέχει και άλλες δύο επιλογές. Την επιλογή **Select...** και την επιλογή **New Table_name** (New Country στο παράδειγμα μας).

Από τις δύο αυτές επιλογές σημαντική είναι μόνο η δεύτερη (New Table_name), καθώς η πρώτη μόνο προτρέπει τον χρήστη να επιλέξει μία από τις διαθέσιμες επιλογές της λίστας. Απεναντίας η επιλογή New Table_name, παρέχει κάποια επιπλέον λειτουργικότητα στο UI της εφαρμογής. Δίνει την δυνατότητα στον χρήστη να προσθέσει, αν δεν υπάρχει ήδη, μία ακόμη επιπλέον επιλογή στην dropdown λίστα. Έτσι, αν στο παράδειγμα μας ο χρήστης επιλέξει New Country, τότε θα οδηγηθεί σε μία νέα οθόνη εισαγωγής η οποία αναπαριστά τον πίνακα Country. Η οθόνη αυτή απεικονίζεται στο ακόλουθο σχήμα.



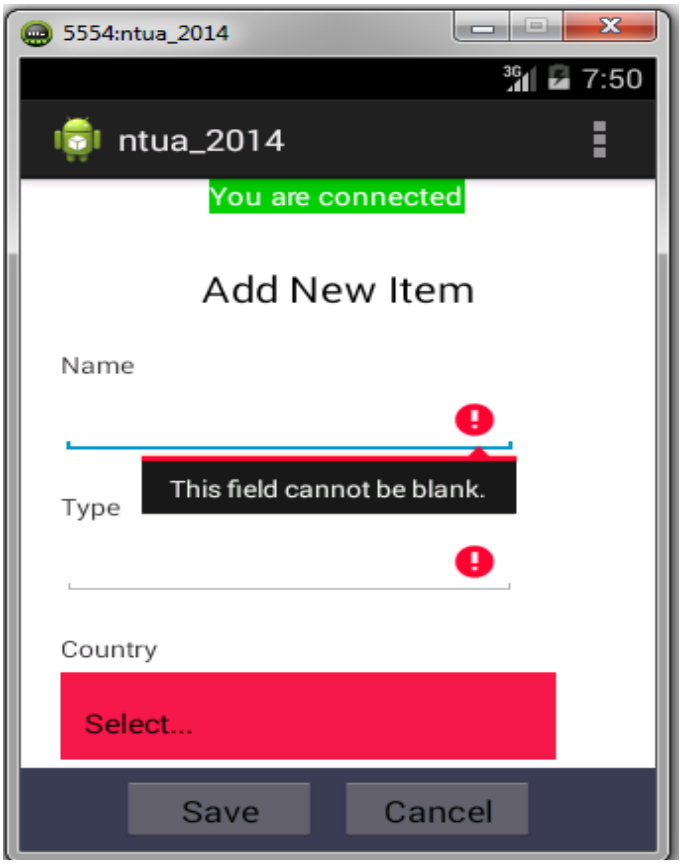
Σχήμα 1.4 Οθόνη εισαγωγής στον πίνακα Country της βάσης.

Στην συνέχεια αναλύουμε την λειτουργία των πλήκτρων **Save** και **Cancel,** που φαίνονται στις παραναπάνω οθόνες εισαγωγής.

Το πλήκτρο Cancel μας μεταφέρει πάντα στην προηγούμενη οθόνη, π.χ. αν από την οθόνη εισαγωγής Country πιέσουμε το πλήκτρο Cancel θα οδηγηθούμε στην οθόνη εισαγωγής του πίνακα item.

Το πλήκτρο Save έχει κάπως μία πιο σύνθετη λειτουργία. Αν βρισκόμαστε στην default οθόνη εισαγωγής (π.χ. την item) και πατήσουμε Save, τότε θα πραγματοποιηθεί validation (π.χ. αν τα πεδία είναι συμπληρωμένα με τιμές, αν το μέγεθος των τιμών τους είναι έγκυρο ή αν περιέχουν έγκυρες τιμές- αριθμητικά κ.α.) στα

πεδία εισαγωγής και αν όλα είναι εντάξει, τα αποτελέσματα θα σταλούν στον Server προκειμένου αυτός να ενημερώσει την Βάση Δεδομένων. Ωστόσο, αν κάποια ή κάποιο πεδίο δεν περάσει με επιτυχία το validation, τότε δεν αποστέλλεται κάτι προς τον Server και εμφανίζεται κατάλληλο μήνυμα στο χρήστη.

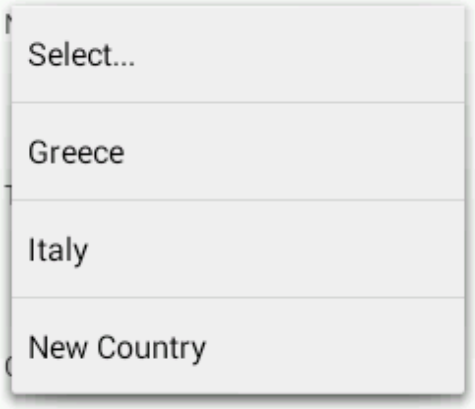


Σχήμα 1.5 Save και αποτυχία στο Validation.

Τα πράγματα είναι διαφορετικά όταν δεν πρόκειται για την default οθόνη εισαγωγής, αλλά για κάποια οθόνη "παιδί". Δηλαδή για κάποια οθόνη η οποία συνδέεται μέσω μίας dropdown λίστας με μία προηγούμενη οθόνη. Για παράδειγμα όπως είναι η οθόνη Country στον παράδειγμα μας, που συνδέεται με την item. Στην περίπτωση αυτή, μόλις πατήσουμε το πλήκτρο Save γίνεται ότι και προηγουμένως, με την διαφορά ότι κάποια ποσότητα πληροφορίας επιστρέφει στην "πατρική" οθόνη. Αυτό γίνεται με σκοπό να ενημερωθούν οι επιλογές της dropdown λίστας που αντιστοιχεί στον πίνακα που μόλις ενημερώσαμε στην Βάση.

Στα σχήματα που ακολουθούν απεικονίζονται αυτά που μόλις είπαμε πιο πάνω. Δηλαδή στο πίνακα Country προσθέτουμε μία νέα χώρα π.χ. την Ιταλία και βλέπουμε πως μετά το Save ενημερώνονται, εκτός από την βάση δεδομένων, και οι διαθέσιμες επιλογές της dropdown λίστας στην οθόνη Item.

Σχήμα 1.6 Εισαγωγή νέας χώρας.



Σχήμα 1.7 Ενημέρωση της dropdown λίστας στην “πατρική” οθόνη.

1.2.2 Server Side υποσύστημα

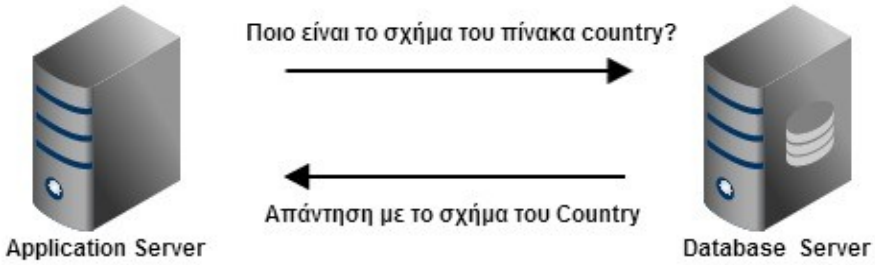
Όσον αφορά τον server, μπορούμε να διακρίνουμε δύο διαφορετικές λειτουργικότητες. Η πρώτη αφορά την λήψη του σχήματος ενός συγκεκριμένου πίνακα της βάσης δεδομένων, ενώ η δεύτερη αφορά κάποιες ενέργειες που θα διεξαχθούν πάνω στον συγκεκριμένο πίνακα. Θα πρέπει να σημειωθεί ότι ο σχεδιασμός των πινάκων της βάσης δεδομένων έχει άμεση σχέση με την κάθε οθόνη που θα εμφανιστεί στο app του smartphone. Ουσιαστικά, κάθε πίνακας της βάσης δεδομένων συνδέεται και με μια οθόνη της εφαρμογής του κινητού τηλεφώνου , όπου τα πεδία του πίνακα (με εξαίρεση το id) εμφανίζονται και ως πεδία εισαγωγής στην αντίστοιχη οθόνη του κινητού

Για να γίνουν πιο καταννοητά τα παραπάνω, θα δώσουμε ένα παράδειγμα για το πως έχει σχεδιαστεί και λειτουργεί το σύστημα μας. Στο παράδειγμα αυτό θα χρησιμοποιήσουμε οθόνη που ήδη παρουσιάσαμε προηγουμένως.

Έστω στην βάση δεδομένων ότι υπάρχει ένας πίνακας που ονομάζεται **country** με πεδία (**id**,**name**,**continent**) και από την εφαρμογή του κινητού έχει ζητηθεί να απεικονιστεί η οθόνη εισαγωγής για την country. Οπότε ακολουθούνται τα παρακάτω βήματα:

i. Η εφαρμογή του κινητού ζητάει από τον application server τα δεδομένα για να σχηματίσει την οθόνη του country.

ii. Ο application server διαβάζει το σχήμα του συγκεκριμένου πίνακα (Country) από την βάση δεδομένων.



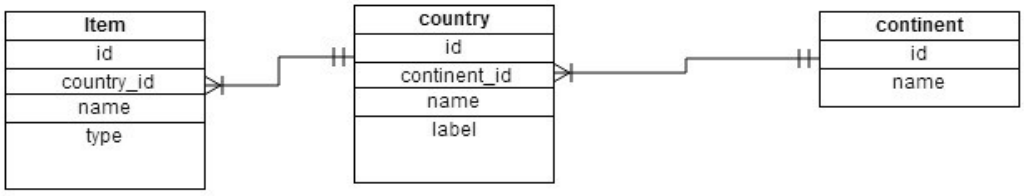
iii. Ο Application Server επεξεργάζεται το σχήμα (πχ αφαιρεί την στήλη id η οποία θεωρούμε ότι υπάρχει σε όλους τους πίνακες, και η τιμή της δεν ενδιαφέρει την εφαρμογή του κινητού), και το επιστρέφει.

iv. Η εφαρμογή του κινητού λαμβάνει το μήνυμα με το σχήμα του πίνακα, και το απεικονίζει. Η οθόνη του κινητού, παρουσιάστηκε στο Σχήμα 1.4

1.3 Μοντέλο Οντοτήτων Συσχετίσεων

Δίνουμε εδώ το μοντέλο Οντοτήτων- Συσχετίσεων της βάσης δεδομένων που θα χρησιμοποιήσουμε.

Σε αυτό το σημείο θα πρέπει να αναφέρουμε ότι στη δομή της βάσης μας , κατά σύμβαση, θα πρέπει ο κάθε πίνακας (εκτός από τον τελικό) να έχει μόνο ένα πεδίο το οποίο να συσχετίζεται με ένα έναν άλλον πίνακα με την σχέση **Πολλά προς -1**, και αυτό να συμβαίνει για όλους τους πίνακες πλην ενός (του τελικού). Στο συγκεκριμένο παράδειγμα θα πρέπει να έχει μια σχέση της παρακάτω μορφής:



Δηλαδή πολλά item μπορεί να πάρουν μία συγκεκριμένη τιμά από τον πίνακα country (Πολλά προς -1) , που ομοίως μπορεί να πάρει μια τιμή από τον πίνακα continent.

Αυτού του είδους η σύμβαση με την οποία δημιουργούμε την βάση,μας βοηθάει στον σχηματισμό των οθόνων στην εφαρμογή του κινητού και συνεπώς στην οθόνη θα εμφανιστούν τα ακόλουθα πεδία:

- Το **id** δεν εμφανίζεται ποτέ, γιατί δεν υπάρχει νόημα να εμφανίζεται
- Τα πεδία **name**,**type** είναι τύπου text
- Το πεδίο **country_id** της βάσης που ουσιαστικά αποτελεί FK (Foreign Key) για τον πίνακα country , ουσιαστικά παρουσιάζεται ως μια dropdown λίστα στο κινητό, με όνομα country, και ως δεδομένα έχει το πεδίο name όλων των records του πίνακα country

2 Σχεδίαση Συστήματος

Στο σημείο αυτό θα παρουσιάσουμε, με μία σύντομη περιγραφή, τα συστατικά μέρη από τα οποία αποτελείται η εφαρμογής μας. Η ανάλυση θα χωριστείς σε δύο σκέλη. Το πρώτο θα αφορά το Client Side υποσύστημα και το δεύτερο το Server Side. Επίσης, για το κάθε υποσύστημα, θα παρουσιάσουμε διαγραμματικά τον τρόπο με τον οποίο σχετίζονται και αλληλεπιδρούν μεταξύ τους τα διάφορα μέρη τα οποία το συνθέτουν.

2.1 Αρχιτεκτονική

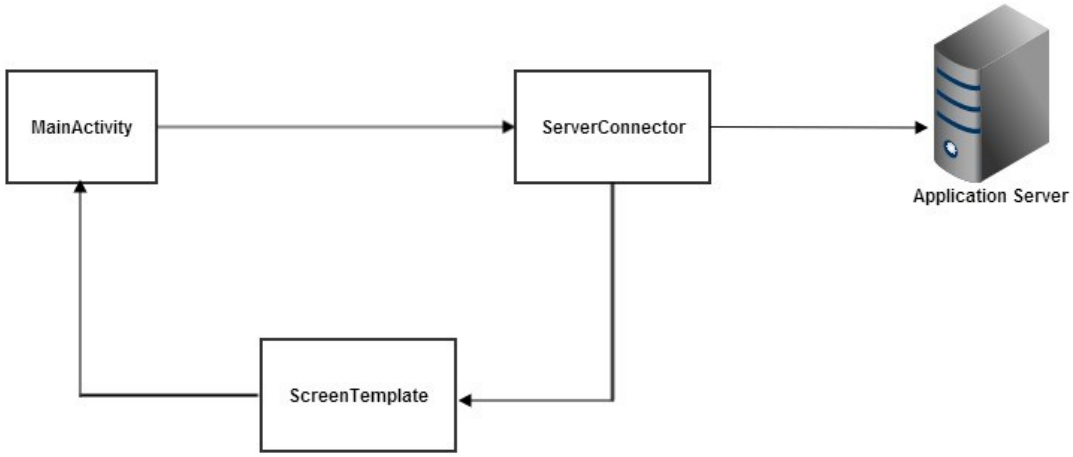
Εδώ παρουσιάζουμε τα επιμέρους κομμάτια από τα οποία θεωρούμε ότι έχει κτιστεί ο κώδικάς μας. Συνήθως, επειδή οι περισσότεροι γράφετε σε αντικειμενοστρεφή γλώσσα προγραμματισμού (C++/JAVA), τα κομμάτια είναι στην ουσία οι κλάσεις της εφαρμογής. Για κάθε κλάση γράψτε μια σύντομη περιγραφή (η αναλυτική περιγραφή των μεθόδων/λειτουργιών/συναρτήσεων του θα ακολουθήσει στην επόμενη ενότητα). Τέλος, δίνουμε ένα γενικό σχήμα που δείχνει τις κλάσεις και πώς αυτές επικοινωνούν μεταξύ τους. Το σχήμα αυτό π.χ. αρκεί να είναι ένα απλό block diagram κλάσεων, όπου θα φαίνεται η κληρονομικότητα και οι συνδέσεις μεταξύ των κλάσεων. Μια κλάση συνδέεται με μια άλλη αν μια μέθοδός της χρησιμοποιεί αντικείμενο από την άλλη ως παράμετρο.

2.1.1 Client Side

Η Android εφαρμογή αποτελείται από τρία βασικά μέρη. Το πρώτο μέρος είναι επιφορτισμένο με την όλη διαχείριση της εφαρμογής (π.χ. Ποια οθόνη να ζητήσει από τον Server, ποιο Screen Template να διαλέξει, ποιο Listener να εκτελέσει κ.α.),

το δεύτερο αφορά την όλη επικοινωνία με τον Server (π.χ. ελέγχει αν υπάρχει κάποιο έγκυρο connection, διαχειρίζεται τα μηνύματα που θα στείλει και θα λάβει, κάνει την κωδικοποίηση και αποκωδικοποίηση των δεδομένων κ.α.) και το τρίτο μέρος ασχολείται με την δημιουργία και κατασκευή των οθονών της Android εφαρμογής.

Ακολουθεί ένα διάγραμμα το οποίο συνδυάζει συνοπτικά τα τρία μέρη από τα οποία συνίσταται η Android εφαρμογή και στην συνέχεια αναλύουμε λεπτομερώς το κάθε μέρος ξεχωριστά.

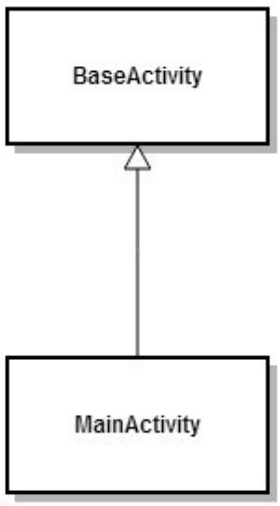


2.1.1.1 MainActivity

Η βασική λειτουργία της κλάσης MainActivity είναι ο καθορισμός και η διαχείριση της οθόνης που θα σχεδιαστεί στο κινητό μας. Η MainActivity κληρονομεί βασική λειτουργικότητα από την BaseActivity. Κληρονομεί, δηλαδή, μεθόδους που καθορίζουν το action (ή URL) του Server με το οποίο η Android εφαρμογή θα αλληλεπιδράσει. Κληρονομεί μεθόδους που δηλώνουν τον πίνακα της βάσης τον οποίο η εφαρμογή θα προσπελάσει, μεθόδους που επιλέγουν το ScreenTemplate της οθόνης (για τις περιπτώσεις που πρόκειται να σχεδιαστεί κάποια νέα οθόνη), μεθόδους που θέτουν τα listeners στα διάφορα συστατικά στοιχεία του view (button, dropdown λίστες κ.α), μεθόδους που υλοποιούν τα listeners και

τέλος μεθόδους που εκτελούνται σε διαφορετική φάση του κύκλου ζωής του MainActivity.

Στην συνέχεια, παραθέτουμε σχηματικά την σχέση των δύο κλάσεων που μόλις περιγράψαμε.



2.1.1.2 ServerConnector

Η κλάση ServerConnector κληρονομεί από την AsyncTask, προκειμένου να αποκτήσει δυνατότητα ασύγχρονης εκτέλεσης. Έτσι εκτελούνται οι χρονοβόρες εργασίες που λαμβάνουν χώρα σε ένα Activity του Android, καθώς με αυτό τον τρόπο η διεπαφή χρήστη παραμένει responsive και δεν "παγώνει" μέχρι να ολοκληρωθεί η εργασία. Η κλάση αυτή, στην πραγματικότητα, είναι ένα νέο thread που ξεκινάει από την MainActivity.

Η ServerConnector κρατάει αναφορές προς την κλάση MainActivity (η οποία και δημιούργησε την ServerConnector) και προς την ScreenTemplate (την οθόνη που είχε αρχικοποιήσει για σχεδίαση η MainActivity). Αυτό συμβαίνει, γιατί όταν η ServerConnector

ολοκληρωθεί, θέλουμε όλες οι λειτουργίες που θα κάνει να γίνουν στο thread του αρχικού MainActivity.

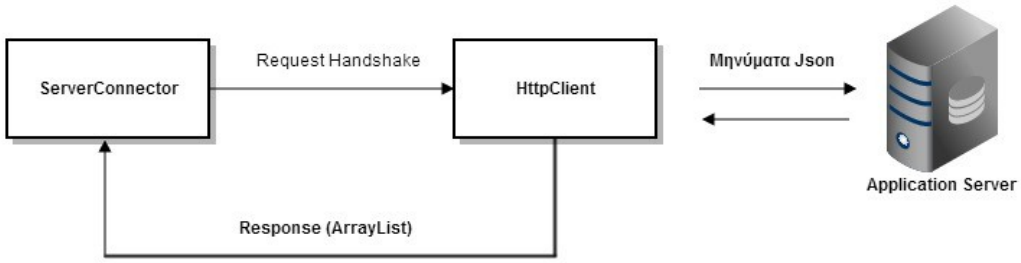
Η Βασικές λειτουργίες της κλάσης αυτής, είναι η επικοινωνία με τον Server και η διαμόρφωση των αποτελεσμάτων (που λαμβάνει από αυτών) σε μορφή κατάλληλη για την περαιτέρω επεξεργασία από την Android εφαρμογή.

Κλάση HttpClient

Η κλάση ServerConnector δημιουργεί ένα αντικείμενο HttpClient, το οποίο είναι επιφορτισμένο με την αποστολή και την λήψη των δεδομένων προς τον Server. Η κλάση HttpClient δέχεται, σε μορφή Hashtable, το request που στέλνει η Android εφαρμογή στον Server, το μετατρέπει σε JSON, το κωδικοποιεί σε Base64 μορφή (με την βοήθεια της κλάσης **EncoderDecoder**) και το αποστέλλει.

Η κλάση HttpClient, στην συνέχεια, λαμβάνει την απόκριση από τον Server και κάνει ακριβώς την αντίστροφη διαδικασία. Δηλαδή, αποκωδικοποιεί το μήνυμα από Base64 σε κανονική μορφή και μετατρέπει το response του Server σε μορφή κατάλληλη για την Android εφαρμογή (με την βοήθεια της κλάσης **DbSchemaParser**).

Στο σημείο αυτό αξίζει να σημειώσουμε, ότι η Android εφαρμογή αναμένει την απόκριση του Server σαν ένα ArrayList από Hashtables.



2.1.1.3 ScreenTemplate

Το τρίτο μέρος της Android εφαρμογής ασχολείται με την σχεδίαση των οθονών. Στο σημείο αυτό καταβλήθηκε ιδιαίτερη μέριμνα, έτσι ώστε ο προγραμματιστής να έχει ευελιξία ως προς τον τύπο και την ποικιλία των οθονών που μπορεί να σχεδιάσει. Όλες οι οθόνες επεκτείνουν την abstract κλάση ScreenTemplate. Η κλάση αυτή παρέχει κάποια κοινή λειτουργικότητα, καθώς και κάποιες προδιαγραφές τις οποίες πρέπει όλες οι οθόνες να ικανοποιούν.

Τώρα, επεκτείνοντας την κλάση ScreenTemplate μπορούμε να φτιάξουμε τους διαφόρους τύπους οθονών που θα διαθέτει η εφαρμογή μας. Για τις ανάγκες της διπλωματικής μας, χρειάστηκε να φτιάξουμε μόνο έναν τύπο οθόνης. Τον τύπο EditScreen, καθώς η εφαρμογή μας θα κάνει μόνο εισαγωγή δεδομένων στην Βάση. Με τον ίδιο τρόπο θα μπορούσε, κάποιος άλλος, να φτιάξει έναν τύπο οθόνης π.χ. ListScreen, ο οποίος θα απεικόνιζε τα δεδομένα που θα ήταν αποθηκευμένα στους πίνακες της Βάσης Δεδομένων. Επίσης, με τον τρόπο αυτό θα μπορούσε κάποιος να ορίσει, όσες διαφορετικές οθόνες επιθυμούσε να έχει η εφαρμογή του π.χ. μία HomeScreen, ErrorScreen ή μία διαφορετική οθόνη για τον κάθε πίνακα της Βάσης Δεδομένων.

Κλάση EditScreen

Η κλάση EditScreen επεκτείνει την abstract κλάση ScreenTemplate, η οποία όπως έχουμε αναφέρει και πιο πάνω είναι το πρότυπο μίας οθόνης, και στην ουσία αποτελεί μία υλοποίηση οθόνης.

Η EditScreen στην ουσία αποτελεί μια γραφική αναπαράσταση κάποιου Table της απομακρυσμένης Βάσης, στο οποίο θέλουμε να κάνουμε εισαγωγή δεδομένων. Συνεπώς, καθίσταται σαφές ότι κάθε οθόνη της Android εφαρμογής αντιστοιχεί σε κάποιο Table της απομακρυσμένης Βάσης και ότι επεκτείνοντας την abstract κλάση ScreenTemplate μπορούμε να σχεδιάσουμε διαφορετικούς τύπους

οθόνης για τις διαφορετικές ενέργειες που θέλουμε εκτελέσουμε στο Table της Βάσης (π.χ EditScreen για εισαγωγή δεδομένων στο Table, ListScreen για απεικόνιση των περιεχομένων του Table κ.α). Επίσης, στις περιπτώσεις που θέλουμε κάτι πιο εξειδικευμένο, θα μπορούσαμε να σχεδιάσουμε για το κάθε Table της απομακρυσμένης Βάσης και μία διαφορετική οθόνη, η οποία θα το περιγραφεί πλήρως.

Δύο είναι τα βασικά σημεία για τον ορισμό μίας νέας οθόνης:

1. Ο καθορισμός του Layout της οθόνης. Δηλαδή η υλοποίηση των abstract μεθόδων header, body, footer οι οποίες αντίστοιχα σχετίζονται με το πάνω, το μεσαίο και το κάτω μέρος της οθόνης. Κάποια ή κάποιες από αυτές μπορεί να έχουν κενή υλοποίηση. Ωστόσο, αρκεί να έχει υλοποιηθεί τουλάχιστον μία.
2. Ο καθορισμός των events που αντιστοιχούν σε κάποια στοιχεία της οθόνης π.χ. Buttonw, dropdown λίστες κ.α.

Στο σημείο αυτό, σαν επίλογος αυτής της ενότητας, κρίνεται απαραίτητο να εξηγήσουμε τον λόγο που μας ώθησε να σχεδιάσουμε με αυτόν τρόπο την υλοποίηση των οθονών.

Το βασικό εμπόδιο που συναντήσαμε ήταν το γεγονός ότι έπρεπε, κάθε φορά που δημιουργούσαμε μία νέα οθόνη, να την δηλώνουμε στο AndroidManifest.xml αρχείο. Όπως όλοι αντιλαμβάνεστε, κάτι τέτοιο μας περιόριζε αρκετά και επίσης κατάστρεφε την γενίκευση του κώδικα μας. Συνεπώς, για να μπορέσουμε να κάνουμε την εφαρμογή μας όσο πιο generic γινόταν. Αποφασίσαμε να έχουμε στο AndroidManifest.xml μία οθόνη δηλωμένη, την οποία κάθε φορά θα διαμορφώνουμε δυναμικά, έτσι ώστε να προσαρμόζεται στις εκάστοτε ανάγκες μας.

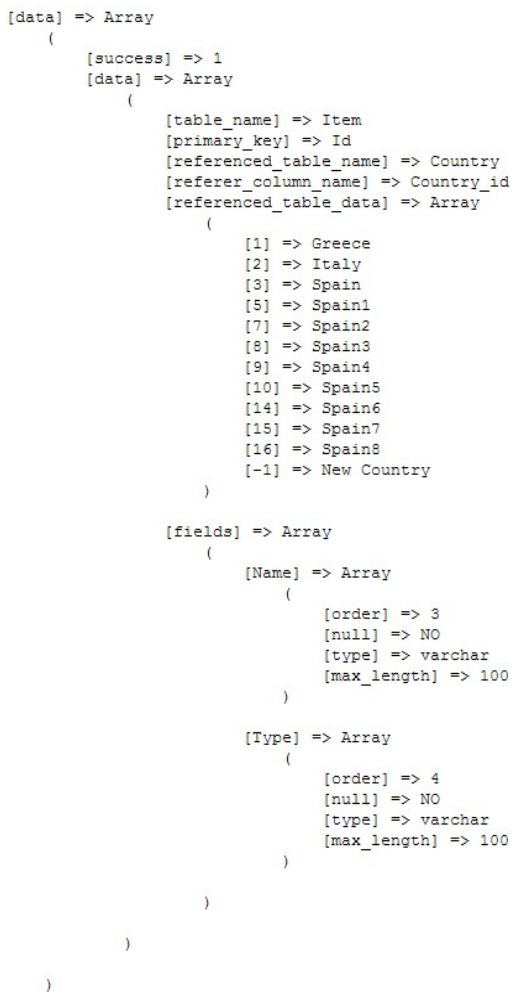
2.1.2 Server Side

Όπως αναφέραμε και παραπάνω, όσον αφορά τον server, μπορούμε να διακρίνουμε δύο διαφορετικές λειτουργικότητες - **την λήψη του σχήματος της κάθε ζητούμενης οθόνης** από τον αντίστοιχο πίνακα της βάσης δεδομένων και την **εφαρμογή της ζητούμενη ενέργειας (action) στην βάση** (πχ insert record, delete record κτλ.). Ακολούθως θα περιγράψουμε της δύο κύριες κλάσεις που ευθύνονται για την διαχείριση των δύο προαναφερθέντω λειτουργειών.

2.1.2.01 Κλάση DbSchemaCreatorController

Η κλάση αυτή χρησιμοποιείται είτε για να πάρει το σχήμα από όλα τους πίνακες της βάσης δεδομένων - με την getTables() - , είτε για να για να διαβάσει και έπειτα να αποστείλει το σχήμα και τα επεξεργασμένα δεδομένα από την βάση στην εφαρμογή του κινητού . Η διαδικασία αυτή επιτυγχάνεται μέσω δύο μεθόδων την getTableStructure (table_name) και της getTableData(table_name). Όπως εξηγούμε και στην παράγραφο 1.3, η getTableStructure επιστρέφει στην εφαρμογή του κινητού μια δομή με όλη την πληροφορία για την συγκεκριμένη οθόνη (που ιουσιαστικά αντιστοιχεί σε πίνακα της βάσης δεδομένων), διαγράφοντας το πεδίο id , σε όλα τα πεδία πλην του FK επιστρέφει το όνομα και στοιχεία που χρησιμοποιούνται γοα το validation, ενώ για το FK επιστρέφει επιπρόσθετα και μια δομή με την μορφή πίνακα, η οποία ως κλειδί έχει το id του αναφερόμενου πίνακα (πατέρα) και ως τιμή το πεδίο name του πίνακα πατέρα. Με την παραπάνω δομή , επιστρέφονται όλες οι εγγραφές του πίνακα - πατέρα με τον τρόπο που περιγράψαμε (id, name).

Ένα τυπικό μύνημα που επιστρέφεται απεικονίζεται στο παρακάτω σχήμα:



Θα πρέπει να αναφέρουμε ότι η getTableData, καλείται από την getTableStructure για να φέρει τα δεδομένα από τον πίνακα πατέρα (σε αυτόν που αναφέρεται το FK) και αυτό το πεδίο ουσιαστικά θα εμφανιστεί στην οθόνη του κινητού ως dropdown list με τα δεδομένα του πίνακα πατέρα.

2.1.2.02 Κλάση DatabaseController

Η κλάση αυτή είναι υπεύθυνη για τον χειρισμό της βάσης δεδομένων. Με τον όρο χειρισμό εννοούμε την εισαγωγή/ενημέρωση/διαγραφή εγγραφών σε ένα συγκεκριμένο πίνακα της βάσης, την δημιουργία ή διαγραφή πεδίων σε έναν συγκεκριμένο πίνακα στην βάση, αλλά και την δημιουργία/διαγραφή ολόκληρων πινάκων από την βάση.

2.1.2.03 Κλάση BaseController

Η κλάση αυτή κληρονομείται τόσο από την DbSchemaCreatorController όσο και τη DatabaseController, και είναι ουσιαστικά υπεύθυνη για την λήψη των συμπιεσμένων και κωδικοποιημένων δεδομένων , την αποκωδικοποίηση και αποσυμπίεση τους, και την μετατροπή τους από την μορφή JSON σε μορφή php πίνακα.

2.2 Περιγραφή Κλάσεων

Στην συνέχεια ακολουθεί μία σύντομη και περιεκτική περιγραφή των μεθόδων των κλάσεων. Και σε αυτή την ενότητα, η ανάλυση θα χωριστεί σε δύο μέρη. Το πρώτο μέρος θα αφορά το Client Side κομμάτι, ενώ το δεύτερο το Server Side.

2.2.1 Client Side

2.2.1.01 BaseActivity

Η κλάση BaseActivity παρέχει την στοιχειώδη λειτουργικότητα των Activities. Για τον λόγο αυτό περιέχει τις ακόλουθες μεθόδους:

- **setActionURL:** η μέθοδος αυτή καθορίζει το Action του Server, με το οποίο θα αλληλεπιδράσει το Activity.
- **setActionDBTable:** η μέθοδος αυτή καθορίζει το Table της βάσης, στο οποίο θα ενεργήσει το Action του Server.
- **setScreenTemplate:** η μέθοδος αυτή καθορίζει το είδος της οθόνης (π.χ. EditScreen, ListScreen κ.α) που θα υλοποιήσει το Activity.
- **setListeners :** η μέθοδος αυτή σετάρει τους ακροατές των διαφόρων στοιχείων από τα οποία αποτελείται η οθόνη.
- **getURL:** η μέθοδος αυτή επιστρέφει το URL του Action με το οποίο αλληλεπιδρά το Activity.
- **getActionDBTable:** η μέθοδος αυτή επιστρέφει το Table της βάσης στο οποίο ενεργεί το Action του Server.

- **getScreenTemplate:** η μέθοδος αυτή επιστρέφει την οθόνη που υλοποιεί το Activity.

2.2.1.02 MainActivity

Η MainActivity επεκτείνει την κλάση BaseActivity και είναι στην ουσία αυτή που χτίζει και διαχειρίζεται την οθόνη. Για τον λόγο αυτό αποτελείται από τις ακόλουθες μεθόδους:

- **onCreate:** η οποία στην ουσία χτίζει την οθόνη. Καθορίζει το Action του Server, το Table της βάσης δεδομένων και τον τύπο της οθόνης.

- **onItemSelected:** η μέθοδος αυτή υλοποιεί τον Listener των dropdown λιστών. Στην πραγματικότητα, προκειμένου να έχουμε μεγαλύτερη ευελιξία, εκτελεί έναν Adapter της abstract κλάσης ScreenTemplate.

- **onClick:** η μέθοδος αυτή υλοποιεί τον onClick Listener. Όπως και πριν, για χάρη της ευελιξίας, και αυτή εκτελεί έναν Adapter της abstract κλάσης ScreenTemplate.

- **onStart:** η μέθοδος εκτελείται όταν εκκινεί η οθόνη.

- **onRestart:** η μέθοδος αυτή εκτελείται όταν η οθόνη ξανά ξεκινάει από την κατάσταση Stop.

- **onPause:** η μέθοδος αυτή εκτελείται όταν η οθόνη σταματάει προσωρινά για να εκκινήσει κάποια άλλη.

- **onResume:** η μέθοδος αυτή εκτελείται όταν η οθόνη ξανά αποκτάει το focus και είναι έτοιμη να επανεκκινήσει.

- **onStop:** η μέθοδος αυτή εκτελείται όταν η οθόνη χάνει το focus από κάποια άλλη.

- **onDestroy:** η μέθοδος αυτή εκτελείται όταν η οθόνη τερματίζει εντελώς την λειτουργία της.

- **onActivityResult:** η μέθοδος αυτή εκτελείται όταν κάποιο Activity, το οποίο έχει ενεργοποιηθεί από το τρέχον Activity, χρειάζεται να επιστρέψει κάποια αποτελέσματα στο αρχικό.

2.2.1.03 ServerConnector

Η κλάση Server Connector επεκτείνει την AsyncTask, για να δημιουργήσει ένα καινούργιο thread και να "τρέξει" σε αυτό. Αυτή είναι μια τεχνική που χρησιμοποιεί το Android για τις χρονοβόρες εργασίες, προκειμένου να διατηρεί το περιβάλλον του Activity κάθε ώρα και στιγμή responsive απέναντι στον χρήστη. Η κλάση αυτή στην ουσία δημιουργεί το κατάλληλο περιβάλλον το οποίο θα αλληλεπιδράσει με τον Server. Καθορίζει το Action του Server με το οποίο θα αλληλεπιδράσει η Android εφαρμογή, περνάει τις κατάλληλες παραμέτρους σε αυτό και τέλος (αναλόγως με το Action) διαχειρίζεται το Response του Server.

Το είδος της λειτουργίας που επιτελεί κάθε φορά η κλάση Server Connector ορίζεται από τις σταθερές `GET_TABLE_SCHEMA` (παίρνει το σχήμα κάποιου καθορισμένου πίνακα της βάσης δεδομένων και σχεδιάζει την οθόνη του) και `SAVE_TABLE_DATA` (αποθηκεύει δεδομένα σε κάποιον καθορισμένο πίνακα της βάσης δεδομένων). Στην συνέχεια παρουσιάζουμε συνοπτικά τις μεθόδους από τις οποίες αποτελείται η κλάση Server Connector:

- **server Connector:** η κλάση Server Connector έχει μία μέθοδο κατασκευαστή για την καθεμία από τις δύο διαφορετικές ενέργειες που υποστηρίζει.
- **onPreExecute:** η μέθοδος αυτή εκτελείται πριν εκκινήσει η κυρία λειτουργία του thread που δημιουργεί η κλάση ServerConnector. Η βασική λειτουργία της μεθόδου αυτή είναι να επιτελεί κάποιες προπαρασκευαστικές λειτουργίες (getTebleSchemaPreExecute, saveTableDataPreExecute) πριν την κυρία εκτέλεση του thread.
- **doInBackground:** στην μέθοδο αυτή εκτελείται η κυρίως λειτουργία του thread. Όπου στην περίπτωση μας καλεί, αναλόγως με την λειτουργία που εκτελεί η κλάση Server Connector, την κατάλληλη μέθοδο (getTableSchemaRequest, saveTableDataRequest).

- **onPostExecute:** όταν η μέθοδος doBackground ολοκληρώσει την λειτουργίας της περνάει το response που λαμβάνει από τον Server σε αυτή την μέθοδο για περαιτέρω επεξεργασία. Άρα η onPostExecute εκτελείται αφού το thread ολοκληρώσει την κυρία λειτουργία του.
- **checkConnectionStatus:** η μέθοδος αυτή τσεκάρει την σύνδεση με το Internet και αναλόγως αν υπάρχει κάποια ενεργή σύνδεση επιστρέφει true, διαφορετικά επιστρέφει false.
- **getTableSchemaPreEcexcute:** αυτή είναι η onPreExecute στην περίπτωση που ο Server Connector είναι σε mode λειτουργίας `GET_TABLE_SCHEMA` (λαμβάνει το σχήμα ενός Table της βάσης και σχεδιάζει την οθόνη του). Αυτό που κάνει η μέθοδος αυτή, είναι να ρυθμίζει το Status του Connection (μήνυμα πάνω- πάνω στην οθόνη, δείτε προηγούμενες εικόνες) στην οθόνη που θα σχεδιαστεί και επίσης την εμφάνιση κατάλληλου μηνύματος στον Loader της οθόνης.
- **saveTableDataPreExecute:** αυτή είναι η onPreExecute στην περίπτωση που ο Server Connector είναι σε mode λειτουργίας `SAVE_TABLE_DATA` (αποθηκεύει δεδομένα σε κάποιον καθορισμένο πίνακα της βάσης δεδομένων). Αυτό που κάνει η μέθοδος αυτή είναι, σε περίπτωση που υπάρχει Connection, να σετάρει κατάλληλο μήνυμα (σχετικά με την ενέργεια που επιτελείται εκείνη την στιγμή) στον Loader της οθόνης.
- **getTableSchemaRequest:** η μέθοδος αυτή εκτελείται από την doInBackground στην περίπτωση που ο ServerConnector είναι σε mode `GET_TABLE_SCHEMA`. Η λειτουργία που επιτελεί η μέθοδος αυτή είναι απλά η επικοινωνία με τον Server. Με λίγα λόγια περνάει το Request της Android εφαρμογής στον Server και στην συνέχεια λαμβάνει την απόκριση από αυτόν την οποία και προωθεί στην onPostExecute για περαιτέρω επεξεργασία.
- **saveTableDataRequest:** η μέθοδος αυτή είναι η ίδια με την getTableSchemaRequest, αλλά εκτελείται από την doInBackground όταν ο ServerConnector είναι σε mode `SAVE_TABLE_DATA`.
- **getTableSchemaResponse:** η μέθοδος αυτή καλείται στην onPostExecute όταν ο ServerConnector είναι σε mode `GET_TABLE_SCHEMA`. Αυτό που κάνει η μέθοδος αυτή είναι να χτίζει την οθόνη που προκύπτει από το Table για το οποίο η Android εφαρμογή έκανε Request στον Server.
- **saveTableDataResponse:** η μέθοδος αυτή καλείται στην onPostExecute όταν ο ServerConnector είναι σε mode `SAVE_TABLE_DATA`. Αυτό που κάνει αυτή η μέθοδος είναι να επιστρέφει το αποτέλεσμα που αποθηκεύτηκε στο Table της βάσης στην

αρχική οθόνη, από την οποία εκκίνησε το εκάστοτε Activity. Για να καταλάβουμε την ανάγκη αυτού, θα παρουσιάσουμε ένα παράδειγμα. Έστω ότι είμαστε σε μία οθόνη εισαγωγής η οποία διαθέτει ένα dropdown στοιχείο με διάφορες χώρες και έστω ότι θέλουμε να εισάγουμε μία καινούργια χώρα σε αυτή την λίστα. Τότε επιλέγουμε New Country και οδηγούμαστε σε μία νέα οθόνη εισαγωγής (η οποία αντιστοιχεί στο Table με τις χώρες της απομακρυσμένης βάσης). Από την οθόνη αυτή εισάγουμε τις πληροφορίες για την νέα χώρα και πατάμε Save. Με την ενέργεια αυτή θέλουμε, κατά πρώτων, να ενημερωθεί η απομακρυσμένη βάση δεδομένων και κατά δεύτερων να ενημερωθεί η λίστα με τις διαθέσιμες χώρες της προηγούμενη οθόνης (έτσι ώστε να περιλαμβάνεται η νέα επιλογή που προσθέσαμε).

- **setActionURL:** με την μέθοδο αυτή καθορίζεται το action του Server με το οποίο θα αλληλεπιδράσει η κλάση ServerConnector.
- **setServerConnectorAction:** με την μέθοδο αυτή καθορίζεται το mode λειτουργίας (για το αν πρόκειται για το χτίσιμο μίας νέας οθόνης με βάση το σχήμα ενός δεδομένου Table της βάσης ή για το αν πρόκειται για αποθήκευση δεδομένων σε κάποιο Table της βάσης) της κλάσης ServerConnector.

2.2.1.04 HttpClient

Η κλάση HttpClient ουσιαστικά είναι το σημείο επικοινωνίας της Android εφαρμογής με τον απομακρυσμένο Server. Η κλάση αυτή δημιουργεί την σύνδεση με τον Server, μεταφέρει σε αυτόν το Request της Android εφαρμογής και τέλος λαμβάνει το Response από αυτόν.

Ακολουθεί μία σύντομη περιγραφή των μεθόδων της κλάσης HttpClient:

- **HttpClient:** αυτή η μέθοδος κατασκευαστή δέχεται σαν παράμετρο εισόδου το URL της επικοινωνίας με το Server και το αναθέτει στην private μεταβλητή url.
- **getHttpResponse:** αυτή η μέθοδο δέχεται σαν είσοδο το Request (σε μορφή Hashtable) της Android εφαρμογής προς τον Server, το μετατρέπει σε JSON μορφή και μέσω της μεθόδου sendHttpPost το στέλνει στον Server. Στην συνέχεια, λαμβάνει σε JSON μορφή το Response του Server, το μετατρέπει σε String και το περνάει σαν παράμετρο στην κλάση DbSchemaParser (η οποία μετατρέπει το Response του Server σε μορφή κατάλληλη για την Android εφαρμογή) και τέλος επιστρέφει το κατάλληλα αυτό φορμαρισμένο Response στη Android εφαρμογή.
- **SendHttpPost:** η μέθοδος αυτή δέχεται σαν είσοδο το Request της Android εφαρμογής, το στέλνει στον Server και τέλος λαμβάνει σε JSON μορφή ξανά το Response του Server.

2.2.1.05 EncoderDecoder

Όλα τα μηνύματα που στέλνονται ή λαμβάνονται από τον Server είναι συμπιεσμένα (zip) και κωδικοποιημένα σε Base64 μορφή. Συνεπώς, η κλάση αυτή διαθέτει μεθόδους που συμπιέζουν/ αποσυμπιέζουν και κωδικοποιούν/ αποκωδικοποιούν τα μηνύματα επικοινωνίας μεταξύ Android εφαρμογής και Server.

Ακολουθεί μία σύντομη περιγραφή των βασικότερων μεθόδων της κλάσης EncoderDecoder:

- **encodeBase64String:** η μέθοδος αυτή κωδικοποιεί τα μηνύματα επικοινωνίας μεταξύ Android και Server σε μορφή base64 και στην συνέχεια τα συμπιέζει. Επίσης, η μέθοδος αυτή έχει δύο υλοποιήσεις, μία για επεξεργασία μηνυμάτων σε String μορφή και μία άλλη για επεξεργασία μηνυμάτων σε JSON μορφή.
- **decodeBase64String:** η μέθοδος αυτή κάνει την αντίστροφη διαδικασία από την μέθοδο encodeBase64String. Αποσυμπιέζει, δηλαδή, τα μηνύματα επικοινωνίας μεταξύ Android και Server και στην συνέχεια τα αποκωδικοποιεί από base64 σε κανονική μορφή String. Επίσης, και αυτή η μέθοδος έχει δύο υλοποιήσεις. Μία για επεξεργασία μηνυμάτων σε String μορφή και μία άλλη για επεξεργασία μηνυμάτων σε JSON μορφή.

2.2.1.06 DbSchemaParser

Η κλάση αυτή μετατρέπει το Response του Server σε μορφή κατάλληλη για την Android Εφαρμογή. Ακολουθεί μία σύντομη περιγραφή των μεθόδων της κλάσης αυτής. Ωστόσο, σε επόμενο κεφάλαιο, θα δούμε μερικά πράγματα (π.χ την μορφή των δεδομένων της Android εφαρμογής) με μεγαλύτερη λεπτομέρεια.

- **DbSchemaParser:** αυτή η μέθοδος κατασκευαστή δέχεται στην είσοδο της το Response του Server, σε μορφή String, το οποίο το μετατρέπει σε JSON μορφή και το αναθέτει στην μεταβλητή input της κλάσης DbSchemaParser. Επίσης, ο κατασκευαστής αυτός αρχικοποιεί και το Hashtable για τα errors.
- **parse:** η μέθοδος αυτή καλεί την μέθοδο Iterate για να φορμάρει τα δεδομένα σε μορφή κατάλληλη για την Android εφαρμογή.
- **Iterate:** η μέθοδος αυτή λουπάρει το JSON με το Response του Server και το διαμορφώνει σε μορφή κατάλληλη (θα περιγραφεί με περισσότερη λεπτομέρεια σε επόμενο κεφάλαιο) για την Android εφαρμογή.
- **addElementd:** διαμορφώνει στην μορφή που τα θέλει η Android εφαρμογή τα dropdown στοιχεία.

- **addElement:** η μέθοδος αυτή μετατρέπει σε Android format τα δεδομένα του JSON. Διαθέτει δύο υλοποιήσεις, μία για την περίπτωση που το Server Response επιστρέφει το σχήμα ενός Table της απομακρυσμένης Βάσης Δεδομένων (προκειμένου να κατασκευαστεί η οθόνη της) και μία άλλη για την περίπτωση που το Server Response επιστρέφει τα αποτελέσματα από την λειτουργία αποθήκευσης δεδομένων σε κάποιο Table της απομακρυσμένης Βάσης Δεδομένων.
- **sortedIterator:** την μέθοδο αυτή την χρησιμοποιούμε για να ταξινομήσουμε τα δεδομένα του dropdown στοιχείου, με βάση το id τους.
- **setErrors:** με την μέθοδο αυτή σετάρουμε τα διάφορα errors που συμβαίνουν κατά το parsing του JSON.

2.2.1.07 ScreenTemplate

Η abstract αυτή κλάση αποτελεί το πρότυπο που πρέπει να κληρονομούν όλες οι κλάσεις που υλοποιούν κάποια οθόνη, καθώς περιέχει κοινή λειτουργικότητα και κάποιες μεθόδους/ προδιαγραφές τις οποίες οφείλουν όλες οι οθόνες να ικανοποιούν.

Στην συνέχεια παρουσιάζονται όλες οι μέθοδοι αυτής της κλάσης:

- **ScreenTemplate:** η μέθοδος αυτή δέχεται σαν είσοδο το Activity στο οποίο θα δημιουργηθεί η τρέχουσα οθόνη και κρατάει μία αναφορά σε αυτό. Επίσης, αρχικοποιεί και το Object του ListenerHandler.
- **createScreen:** πρόκειται για μία Template μέθοδο η οποία δημιουργεί τα διάφορα μέρη από τα οποία αποτελείται μια οθόνη (header, body, footer). Πρέπει να έχει υλοποιηθεί τουλάχιστον μία από τις μεθόδους header, body και footer, αλλιώς η εκτέλεση της μεθόδους createScreen δεν θα έχει κάποιο εμφανές αποτέλεσμα.
- **setDBTableName:** με την μέθοδο αυτή ορίζεται το όνομα του Table της απομακρυσμένης βάση δεδομένων, στο οποίο αντιστοιχεί η τρέχουσα οθόνη.
- **setScreenElements:** με την μέθοδο αυτή ορίζεται σε ένα ArrayList τα στοιχεία από τα οποία αποτελείται η οθόνη.
- **getListenerHandler:** η μέθοδος αυτή επιστρέφει μία αναφορά του Object ListenerHandler τον οποίο αρχικοποιήσαμε στον κατασκευαστή της κλάσης ScreenTemlate.
- **setConnected:** η μέθοδος αυτή θέτει σε Status True το Connection με τον Server.
- **setDisconnected:** η μέθοδος αυτή θέτει σε Status False το Connection με τον Server.

- **getConnectionStatus:** η μέθοδος αυτή επιστρέφει το τρέχων Status του Connection με τον Server.
- **getActivity:** η μέθοδος αυτή επιστρέφει μία αναφορά του Activity στο οποίο έχει σχεδιαστεί ή στο οποίο πρόκειται να σχεδιαστεί η τρέχουσα οθόνη.
- **clearScreen:** η μέθοδος αυτή μας δίνει την δυνατότητα, αν την υλοποιήσουμε, να κάνουμε clear ή reset της οθόνης μας σε μία αρχική κατάσταση (π.χ μετά την διαδικασία του Save, επιθυμούμε κενά όλα τα EditText και σεταρισμένη η default επιλογή στα dropdown στοιχεία).
- **setRootViewGroup:** η μέθοδος αυτή ορίζει το ViewGroup στο οποίο τοποθετείται το δυναμικό κομμάτι της οθόνης.
- **header:** στην μέθοδο αυτή προτείνεται η υλοποίηση του πάνω μέρους της οθόνης.
- **body:** στην μέθοδο αυτή προτείνεται η υλοποίηση του κυρίως μέρους της οθόνης.
- **footer:** στην μέθοδο αυτή προτείνεται η υλοποίηση του κάτω μέρους της οθόνης.
- **getRootViewGroup:** η μέθοδος αυτή επιστρέφει μία αναφορά του ViewGroup που ορίστηκε στην μέθοδο setRootViewGroup.
- **onClick:** στην μέθοδο αυτή υλοποιούμε τα onClick events (τις ενέργειες που γίνονται με το πάτημα των διαφόρων πλήκτρων της εφαρμογής).
- **onItemSelected:** στη μέθοδο αυτή υλοποιούμε τα onItemSelected events (τις ενέργειες που γίνονται με την επιλογή ενός αντικειμένου μίας dropdown λίστας).
- **onActivityResult:** στην μέθοδο αυτή εκτελείται όταν το focus περνάει ξανά στην παρούσα οθόνη από μία οθόνη παιδί της (π.χ αν η παρούσα οθόνη διαθέτει ένα dropdown με επιλογές και επιλέξουμε να πάμε σε μία άλλη οθόνη για να προσθέσουμε μία νέα τιμή σε αυτό το dropdown, τότε όταν ξανά γυρίσουμε στην αρχική θέλουμε η νέα τιμή να συμπεριλαμβάνεται σε αυτό).
- **onStart:** η μέθοδος αυτή εκτελείται όταν η οθόνη ξεκινάει τον κύκλο ζωής της.
- **onRestart:** η μέθοδος αυτή εκτελείται όταν η οθόνη ξανά ξεκινάει από την κατάσταση Stop.
- **onPause:** η μέθοδος αυτή εκτελείται όταν η οθόνη σταματάει προσωρινά για να εκκινήσει κάποια άλλη.
- **onResume:** η μέθοδος αυτή εκτελείται όταν η οθόνη ξανά αποκτάει το focus και είναι έτοιμη να επανεκκινήσει.
- **onStop:** η μέθοδος αυτή εκτελείται όταν η οθόνη χάνει το focus από κάποια άλλη.

- **onDestroy:** η μέθοδος αυτή εκτελείται όταν η οθόνη τερματίζει εντελώς την λειτουργία της.

2.2.1.08 ScreenParser

Η κλάση αυτή δημιουργεί το δυναμικό μέρος της εκάστοτε οθόνης, με βάση πάντα την πληροφορία που έχει ληφθεί από τον Server. Στην συνέχεια, παρουσιάζονται οι μέθοδοι από τις οποίες αποτελείται η κλάση ScreenParser:

- **ScreenParser:** αυτή η μέθοδος κατασκευαστή έχει σαν βασική λειτουργία τον ορισμό κάποιων default layout παραμέτρων. Επίσης, υπάρχει σε δύο υλοποιήσεις, στην μία εκ των οποίων πέραν του ορισμού του default layout, παίρνει σαν παράμετρο και σετάρει τα στοιχεία από τα οποία αποτελείται η οθόνη.
- **setScreenElements:** με την μέθοδο αυτή σετάρουμε το ArrayList με τα στοιχεία από τα οποία αποτελείται η οθόνη και τα οποία έχουν προκύψει από το σχήμα κάποιου Table της απομακρυσμένης βάσης δεδομένων.
- **setLabelParams:** με την μέθοδο αυτή σετάρουμε τις layout παραμέτρους όλων των label της οθόνης.
- **setEditParams:** με την μέθοδο αυτή σετάρουμε τις layout παραμέτρους όλων των EditText της οθόνης.
- **setSpinnerParams:** με την μέθοδο αυτή σετάρουμε τις layout παραμέτρους όλων των dropdown της οθόνης.
- **setListenerHandler:** με την μέθοδο αυτή σετάρουμε τον ListnerHandler της οθόνης.
- **createScreen:** η μέθοδος αυτή παρσάρει το ArrayList με τα στοιχεία της οθόνης και δημιουργεί την οθόνη.

2.2.1.09 EditScreen

Η κλάση αυτή στην ουσία αποτελεί μία οθόνη κάποιου Table της απομακρυσμένης Βάσης Δεδομένων στο οποίο επιθυμούμε να κάνουμε εισαγωγή δεδομένων. Κληρονομεί την abstract κλάση ScreenTemplate για να αποκτήσει την βασική λειτουργικότητα των οθονών και επίσης να την επεκτείνει. Οι μέθοδοι από τις οποίες αποτελείται η EditScreen αναλύονται στην συνέχεια:

- **EditScreen:** ο κατασκευαστής αυτός κρατάει μία αναφορά στο Activity στο οποίο θα δημιουργηθεί η τρέχουσα οθόνη.

- **setRootViewGroup:** η μέθοδος αυτή ορίζει το root ViewGroup, με λίγα λόγια το container στο οποίο θα τοποθετηθούν τα στοιχεία της οθόνης που έχουν ληφθεί από το σχήμα κάποιου Table της απομακρυσμένης βάσης Δεδομένων.
- **getRootViewGroup:** η μέθοδος αυτή επιστρέφει στην έξοδο της μία αναφορά του root ViewGroup που ορίστηκε στην μέθοδο setRootViewGroup.
- **header:** αυτή η μέθοδος ορίζει το πάνω μέρος της οθόνης. Όπως φαίνεται, το τμήμα αυτό είναι σταθερό και ίδιο για όλες τις EditScreen.
- **body:** αυτή η μέθοδος ορίζει το κυρίως μέρος της οθόνης. Όπως φαίνεται, το τμήμα αυτό είναι δυναμικό και σχεδιάζει τα στοιχεία της οθόνης που έχουν ληφθεί από το σχήμα κάποιου Table της απομακρυσμένης Βάσης Δεδομένων.
- **footer:** αυτή η μέθοδος ορίζει το κάτω μέρος της οθόνης. Όπως φαίνεται, το τμήμα αυτό είναι σταθερό και ίδιο για όλες τις EditScreen.
- **onClick:** αυτή η μέθοδος περιέχει τις λειτουργίες που λαμβάνουν χώρα κατά το onClick event (που ενεργοποιείται με το πάτημα ενός πλήκτρου).
- **onItemSelected:** αυτή η μέθοδος περιέχει τις λειτουργίες που λαμβάνουν χώρα κατά το onItemSelected event (που ενεργοποιείται με την επιλογή ενός στοιχείου κάποιας dropdown λίστας).
- **onActivityResult:** αυτή η μέθοδος διαχειρίζεται τα αποτελέσματα που προέρχονται από μία οθόνη παιδί της τρέχουσας οθόνης (π.χ ενημέρωση μιας dropdown λίστας με την νέα επιλογή που προστέθηκε σε αυτήν σε κάποια οθόνη παιδί της τρέχουσας).
- **clearScreen:** αυτή η μέθοδος επαναφέρει την οθόνη στην αρχική της μορφή (π.χ καθαρισμός όλων των EditText και επαναφορά των dropdown λιστών στις default τιμές τους) μετά από μία διαδικασία Save.
- **OnStop:** αυτή η μέθοδος εκτελείται όταν η παρούσα οθόνη χάνει το focus από μία οθόνη παιδί της.

2.2.1.10 ListenerHandler

Η κλάση αυτή είναι ένας διαχειριστής των listeners. Γνωρίζει για κάθε στοιχείο της οθόνης σε ποιον listener να το αναθέσει. Η κλάση ListenerHandler αποτελείται από τις ακόλουθες μεθόδους:

- **ListenerHandler:** ο κατασκευαστής αυτός δημιουργεί ένα ArrayList για τον κάθε Listener. Είναι ευνόητο ότι ο αριθμός των Listeners θα μπορούσε να είναι μεγαλύτερος. Απλά για τους σκοπούς της διπλωματικής μας επαρκούν οι δύο που χρησιμοποιήσαμε.

- **addOnItemSelectedListener:** με την μέθοδο αυτή, εισάγουμε ένα αντικείμενο στο ArrayList που κρατάει τα στοιχεία τα οποία θα διαθέτουν το onItemSelected Listener.
- **addOnClickListener:** με την μέθοδο αυτή, εισάγουμε ένα αντικείμενο στο ArrayList που κρατάει τα στοιχεία τα οποία θα διαθέτουν το onClick Listener.
- **getOnItemSelectedListener:** η μέθοδος αυτή επιστρέφει το ArrayList με τα αντικείμενα που θα ανατεθούν στο onItemSelected Listener.
- **getOnClickListener:** η μέθοδος αυτή επιστρέφει το ArrayList με τα αντικείμενα που θα ανατεθούν στο onClick Listener.

2.2.1.11 Validator

Η κλάση αυτή διαθέτει μεθόδους για το validation των δεδομένων μίας οθόνης κατά την διαδικασία που θέλουμε να αποθηκεύσουμε δεδομένα σε κάποιο Table της απομακρυσμένης βάσης δεδομένων. Η κλάση Validator αποτελείται από τις ακόλουθες μεθόδους:

- **valid:** η μέθοδος αυτή εκτελεί το validation του στοιχείου που δέχεται σαν παράμετρο. Διαθέτει δύο υλοποιήσεις, μία για τα στοιχεία EditText και άλλη μία για τα dropdown στοιχεία. Στην περίπτωση που η valid εκτελείται στα EditText περνάμε σαν επιπλέον παράμετρο μία λίστα με μεταπληρόφοριες από την βάση σχετικές με το συγκεκριμένο πεδίο (π.χ τύπο δεδομένων, μέγιστο ή ελάχιστο μήκος τιμών κ.α) ενώ στην περίπτωση που εκτελείται για τα dropdown στοιχεία, περνάμε σαν επιπλέον παράμετρο μία λίστα με τις invalid επιλογές.
- **varcharValidation:** η μέθοδος αυτή είναι private και χρησιμοποιείται για το Validation των πεδίων της απομακρυσμένης βάση δεδομένων που είναι τύπου Varchar.
- **integerValidation:** η μέθοδος αυτή είναι private και χρησιμοποιείται για το Validation των πεδίων της απομακρυσμένης βάση δεδομένων που είναι τύπου Integer.
- **getErrorMessage:** η μέθοδος αυτή επιστρέφει το μήνυμα σφάλματος που προέκυψε από το Validation του εκάστοτε πεδίου.

2.2.1.12 StringWithTag

Η κλάση αυτή δημιουργεί ένα βοηθητικό Object με στόχο την δημιουργία ενός ζεύγους id, value για τις dropdiwn λίστες. Στην συνέχεια παρουσιάζουμε τις μεθόδους της κλάσης αυτής:

- **StringWithTag:** η μέθοδος αυτή σετάρει το id και το value του dropdown στοιχείου.

- **ToString:** η μέθοδος αυτή επιστρέφει το value του εκάστοτε dropdown στοιχείου.

2.2.2 Server Side

2.2.2.01 D

Αυτή η κλάση είναι υπεύθυνη για την απεικόνιση μηνυμάτων debugging.

- **dd :** Η μέθοδος αυτή χρησιμοποιείται για την απεικόνιση μηνυμάτων debugging, και μας πληροφορεί, επιπρόσθετα με την μεταβλητή που της έχουμε δώσει ώς ότρισμα να απεικονίσει, και την κλάση στην οποία βρίσκεται , αλλά και την γραμμή στον κώδικα.

2.2.2.02 DatabaseConnection

Η κλάση αυτή είναι υπέυθυνη για την διαχείριση της βάσης δεδομένων. Είναι ουσιαστικά ένας wrapper για την βιβλιοθήκη mysqli της php, και μας παρέχει πληθώρα βελτιώσεων, όπως πιο απλοποιημένη σύνταξη των sql ερωτημάτων. Επιπρόσθετα υλοποιησεί το σχεδιαστικό μοντέλο singletron, σύμφωνα με το οποίο για κάθε session του χρήστη, δεν ανοίγεται καινούργιο connection στην βάση κάθε φορά που εκτελούμε ένα query, αλλά ανοίγεται μόνο την πρώτη φορά, και έπειτα η σύνδεση αυτή αποθηκεύεται. Συνεπώς σε όλες τις μετέπειτα διαδοχικές κλήσεις χρησιμοποιείται το ήδη υπάρχον connection με την βάση. Το αποτέλεσμα είναι πιο γρήγορη πρόσβαση στην βάση.

- **loadCredentials:** Φορτώνει τα διαπιστευτήρια σύνδεσης με μια συθγκεκριμένη βάση δεδομένων από ένα config file
- **getDBHandler:** Εξετάζει αν υπάρχει ήδη ανοιχτό connection. Αν υπάρχει, τότε επιστρέφει αυτό, αλλιώς δημιουργεί ένα καινούργιο και το επιστρέφει.

2.2.2.03 EncoderDecoder

- **decode:** Αποκωδικοποιεί και έπειτα αποσυμπιέζει το μήνυμα που έχει ληφθεί από την εφαρμογή του κινητού
- **encode:** Πρώτα συμπιέζει και έπειτα κωδικοποιεί το μήνυμα που θα σταλεί στην εφαρμογή του κινητού.

2.2.2.04 Query

Μαζί με την DatabaseConnection είναι υπεύθυνες για την πρόσβαση στην βάση. Αυτή κλάση είναι ουσιαστικά υπεύθυνη για την απλοποίηση της μορφής των queries σε σχέση με την mysqli.

2.2.2.05 constants.php

Πρόκειται ουσιαστικά για ένα αρχείο configuration στο οποίο δηλώνουμε το ip της βάσης δεδομένων, τα διαπιστευτήρια (username/password) , το όνομα της βάσης καθώς και τον αρχικό πίνακα/view το οποίο θέλουμε να εμφανίζεται όταν ξεκινάει η εφαρμογή.

2.2.2.06 Router

Πρόκειται ουσιαστικά για μια βοηθητική κλάση που μας επιτρέπει να χρησιμοποιήσουμε σχεδιαστικά μια ενιαία μορφή των urls για να εκτελέσουμε κάποια συγκεκριμένη μέθοδο μια κλάσης. Για παράδειγμα το παρακάτω url:

http://localhost/ntua_diplomatiki/server/**index.php?r=Database/insertRecord**

υποδεικνύει να εκτελεστεί ο κώδικας της μεθόδου **insertRecord** της κλάσης **Database**Controller.

Επιπρόσθετα η Router, έχει κώδικα ο οποίος κάνει auto-include on demand κάθε κλάση που πρόκειται κα χρησιμοποιήσουμε. Με λιγα λόγια κάθε φορά που δημιουργείται ένα καινούργιο αντικείμενο με την **new** (πχ new **Database**Controller) ,υπάρχει κώδικας που κάνει "include **Database**Controller.php"

3 Υλοποίηση

Στο κεφάλαιο αυτό θα αναλύσουμε με μεγαλύτερη λεπτομέρεια κάποια σημεία της υλοποίησης που αξίζει να γίνει ειδική μνεία σε αυτά είτε γιατί είναι κάπως πιο πολύπλοκα και δυσνόητα, είτε γιατί παρουσιάζουν κάποιο αλγοριθμικό ενδιαφέρον. Επίσης, και στο κεφάλαιο αυτό η παρουσίαση θα γίνει σε δύο σκέλη, το Client Side και το Server Side.

3.1 Λεπτομέρειες υλοποίησης Client Side

Στις ενότητες που ακολουθούν θα αναλύσουμε με μεγαλύτερη λεπτομέρεια τα πιο σημαντικά σημεία της υλοποίησης της Android Εφαρμογής.

3.1.1 MainActivity αρχικοποίηση μίας νέας οθόνης

Στην συνέχεια παραθέτουμε το σημείο του κώδικα που ορίζει το URL του Server Action με το οποίο θα αλληλεπιδράσει η Android εφαρμογή, καθώς και το Table της Βάσης που μας ενδιαφέρει.

```
// Set Action's URL
setActionURL("http://10.0.2.2/ntua_diplomatiki/server/index.php?
r=dbSchemaCreator/getTableStructure");


// Set Action's Database Table
Bundle extras = getIntent().getExtras();
if(extras != null){
   setActionDBTable(extras.getString("ActionDBTable"));
}else{
   setActionDBTable("");
}
```

Στις γραμμές κώδικα που ακολουθούν αρχικοποιείται η οθόνη η οποία θα απεικονίσει με γραφικό τρόπο το Table της απομακρυσμένης βάσης δεδομένων.

```
// Screen Initialization
screen = new EditScreen(this);
setScreenTemplate(screen);
```

Ενώ στις παρακάτω γραμμές κώδικα βλέπουμε την αρχικοποίηση του Server Connector. Του στοιχείου δηλαδή που θα αλληλεπιδράσει με τον Server και σε ένα νέο Thread που θα ξεκινήσει, όπως έχουμε πει και πιο πάνω, σχεδιάσει την οθόνη. Στο αντικείμενο Server Connector περνάμε σαν παράμετρο στο κατασκευαστή του όλο το Activity, έτσι είναι δυνατόν να λάβει μία αναφορά στην οθόνη που θα σχεδιάσει.

```
// Server Connector Initialization
ServerConnector server = new ServerConnector(this);
server.setServerConnectorAction(ServerConnector.GET_TABLE_SCHEMA);
            server.execute(getActionDBTable());
```

3.2 Λεπτομέρειες υλοποίησης Server Side

Σε γενικές γραμμές ο κώδικας στο κομμάτι του server είναι μικρός και κατανοητός. Το μόνο σημείο που ίσως να θέλει μια επιπρόσθετη ανάλυση είναι η getTableStructure.

Καταρχάς αν δεν έχει περαστεί ως όρισμα το όνομα του πίνακα της βάσης που θέλουμε , τότε επιστρέφουμε το default πίνακα, που ορίζεται από τον configuration file constants.php

```
if (!empty($table)) {
    } else if (!empty($this->post['table_name'])) {
       $table = $this->post['table_name'];
    } else {
       $table = init_table_name;
    }
    $structure = array();
    $fields = array();
```

Έπειτα εκτελούμε το παρακάτω query:

```
$sql = "SELECT sc.*,kcu.referenced_table_name,referenced_column_name FROM
INFORMATION_SCHEMA.COLUMNS AS sc LEFT JOIN
INFORMATION_SCHEMA.key_column_usage AS kcu ON
```

```
sc.TABLE_SCHEMA=kcu.TABLE_SCHEMA AND sc.TABLE_NAME=kcu.TABLE_NAME AND sc.COLUMN_NAME=kcu.COLUMN_NAME AND KCU.referenced_table_name is not null
    WHERE  sc.TABLE_SCHEMA =? AND sc.TABLE_NAME =?";
    $query = new Query($sql, array($this->dbname, $table), '', 'S');
```

Από όπου μας επιστρέφονται και αποθηκεύονται στην μεταβλητη **$rows** τα αποτελέσματα

```
    $rows = $query->getResults(); //D::dd($rows);
```

Στο σημείο αυτό επεξεργαζόμαστε τα δεδομένα, αγνοείται η στήλη id, και στην περίπτωση που υπάρχει FK περιορισμός, καλείται η getTableData, η οποία γεμίζει με της εγγραφές του referenced_table_name (πίνακας πατέρας) τα δεδομένα στην μορφή πίνακα με κλειδί το id, και τιμή το πεδίο name, για όλες τις εγγραφές , οπως αναφέραμε και σε προηγούμενο κεφάλαιο.

```
    if (is_array($rows) && count($rows) > 0) {
      foreach ($rows as $row) {
        $fields[ucfirst($row['COLUMN_NAME'])] = array(
          "order" => $row['ORDINAL_POSITION'],
          "null" => $row['IS_NULLABLE'],
          "type" => $row['DATA_TYPE'],
          "max_length" => $row['CHARACTER_MAXIMUM_LENGTH'],
        );
        $structure['table_name'] = ucfirst($row['TABLE_NAME']);
        if ($row['COLUMN_KEY'] == 'PRI') {
          $structure['primary_key'] = ucfirst($row['COLUMN_NAME']);
        }
        if (!empty($row['referenced_table_name']) && !empty($row['referenced_column_name'])) {
          if (isset($row['referenced_table_name'])) {
            $structure['referenced_table_name'] = ucfirst($row['referenced_table_name']);
          }
          $structure['referer_column_name'] = ucfirst($row['COLUMN_NAME']);
          $structure['referenced_table_data'] = $this->getTableData($row['referenced_table_name']);
        }
      }
    }
```

Στην παρακάτω γραμή κώδικα αφαιρείτη το πεδίο id, και μετα επιστρέφεται το αποτέλεσμα

```
    unset($fields[$structure['primary_key']]);
    if (isset($structure['referer_column_name'])) {
```

```
            unset($fields[ucfirst($structure['referer_column_name'])]);
        }
        $structure['fields'] = $fields;
        $this->results = $structure;
        return $structure;
    }
```

4 Παράρτημα

Στο σημείο αυτό παρατίθεται ο κώδικας της εφαρμογής μας. Πάλι η παρουσίαση θα χωριστεί σε δύο μέρη, στο Client Side μέρος και στο Server Side.

4.1 Client Side κώδικας

4.1.1 BaseActivity

```
package com.example.ntua_2014;

import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Iterator;

import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Spinner;
import android.widget.AdapterView.OnItemSelectedListener;

public abstract class BaseActivity extends ActionBarActivity implements OnItemSelectedListener, OnClickListener {

	// Action URL
	protected String url		= null;
```

```
// Action Table
protected Hashtable<String, String> actionDBTable = null;

// Screen Template
protected ScreenTemplate screen   = null;

// Set Action's URL
protected void setActionURL(String url){
        this.url = url;
}

// Set Action's Table
protected void setActionDBTable(String name){
        this.actionDBTable = new Hashtable<String, String>();
        if(name != null){
                this.actionDBTable.put("table_name", name);
        }else{
                this.actionDBTable.put("table_name", "");
        }
}

// Set Screen Template
protected void setScreenTemplate(ScreenTemplate screen){
        this.screen = screen;
}
// Get URL
public String getURL(){
        return this.url;
}
// Get DB Table Name
public Hashtable getActionDBTable(){
        return this.actionDBTable;
}
```

```
	// Get Screen Template
	public ScreenTemplate getScreenTemplate(){
		return this.screen;
	}

	// set Action's Listeners
	protected void setListeners(ListenerHandler listener){
		// set buttons onClick Listeners
		ArrayList onClickObj   = listener.getOnClickListener();
		Iterator onClickObjItr = onClickObj.iterator();
		while(onClickObjItr.hasNext()){
			Button btnObj = (Button)onClickObjItr.next();
			btnObj.setOnClickListener(this);
		}

		// set spinners onItemSelected Listeners
		ArrayList onItemSelectedObj   = listener.getOnItemSelectedListener();
		Iterator onItemSelectedObjItr = onItemSelectedObj.iterator();
		while(onItemSelectedObjItr.hasNext()){
			Spinner spinner = (Spinner)onItemSelectedObjItr.next();
			spinner.setOnItemSelectedListener(this);
		}
	}
}
```

4.1.2 MainActivity

```
package com.example.ntua_2014;
import java.util.ArrayList;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.content.Intent;
import android.os.Bundle;

public class MainActivity extends BaseActivity  {



	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		// Set Action's URL
		setActionURL("http://10.0.2.2/ntua_diplomatiki/server/index.php?
r=dbSchemaCreator/getTableStructure");

		// Set Action's Database Table
		Bundle extras = getIntent().getExtras();
		if(extras != null){
			setActionDBTable(extras.getString("ActionDBTable"));
		}else{
			setActionDBTable("");
		}

		// Screen Initialization
	    screen = new EditScreen(this);
```

```
            setScreenTemplate(screen);

            // Server Connector Initialization
            ServerConnector server = new ServerConnector(this);
            server.setServerConnectorAction(ServerConnector.GET_TABLE_SCHEMA);
            server.execute(getActionDBTable());
        }

        @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int position, long id) {
                // onItemSelected Adapter Method
                screen.onItemSeletced(adapterView, position);
    }

        @Override
        public void onClick(View v) {

                // onClick Adapter Method
      screen.onClick(v);
        }

        @Override
        public void onStart(){
                super.onStart();

                // onStart Adapter Method
        screen.onStart();
    }
@Override
public void onRestart(){
      super.onRestart();
      // onRestart Adapter Method
      screen.onRestart();
    }
```

```
	@Override
	public void onPause(){
		super.onPause();

		// onPause Adapter Method
	screen.onPause();
}

	@Override
	public void onResume(){
		super.onResume();

		// onResume Adapter Method
	screen.onResume();
}

	@Override
	public void onStop(){
		super.onStop();

		// onStop Adapter Method
	screen.onStop();
}

	@Override
	public void onDestroy(){
		super.onDestroy();

		// onDestroy Adapter Method
	screen.onDestroy();
}

	@Override
```

```
	protected void onActivityResult(int requestCode, int resultCode, Intent data) {

		// onActivityResult Adapter Method
		screen.onActivityResult(requestCode, resultCode, data);
	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {

		// Inflate the menu; this adds items to the action bar if it is present.
		getMenuInflater().inflate(R.menu.main, menu);
		return true;
	}


@Override
public void onNothingSelected(AdapterView<?> arg0) {
   //do something
}

	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		// Handle action bar item clicks here. The action bar will
		// automatically handle clicks on the Home/Up button, so long
		// as you specify a parent activity in AndroidManifest.xml.
		int id = item.getItemId();
		if (id == R.id.action_settings) {
			return true;
		}
		return super.onOptionsItemSelected(item);
	}
```

```
	/**
	 * A placeholder fragment containing a simple view.
	 */
	public static class PlaceholderFragment extends Fragment {

		public PlaceholderFragment() {
		}

	/*	@Override
		public View onCreateView(LayoutInflater inflater, ViewGroup container,
				Bundle savedInstanceState) {
			View rootView = inflater.inflate(R.layout.fragment_main, container,
					false);
			return rootView;
		}*/
	}
}
```

4.1.3 Server Side

```
package com.example.ntua_2014;

import java.util.ArrayList;
import java.util.Hashtable;

import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.widget.Toast;

public class ServerConnector extends AsyncTask<Hashtable, Void, ArrayList>{
```

```
// Set Server Connector Action
public static final int GET_TABLE_SCHEMA = 0;

public static final int SAVE_TABLE_DATA  = 1;

// Current Server Connection Action
private int currentAction;

// Activity
private BaseActivity activity = null;

// Context
private Context context       = null;

// URL
private String url            = null;

// Screen Template
private ScreenTemplate screen = null;

// Progress dialog
private ProgressDialog pDialog;

public ServerConnector(BaseActivity activity) {

        // Activity and Context Initialization
        if(activity != null){
                this.activity = activity;
                this.context = activity;
        }else{
                System.out.println("You must define Activity");
                System.exit(-1);
        }
```

```
        // URL Initialization
        if(activity.getURL() != null){
                this.url = activity.getURL();
        }else{
                System.out.println("You must define URL Action");
                System.exit(-1);
        }

        // Screen Template Initialization
        if(activity.getScreenTemplate() != null){
                this.screen = activity.getScreenTemplate();
        }else{
                System.out.println("You must define Screen Template");
                System.exit(-1);
        }

        // Set Likely Server Connector Action
        currentAction = GET_TABLE_SCHEMA;
}

public ServerConnector(ScreenTemplate screen, String url) {

        // Activity and Context Initialization
   if(screen != null){
        this.screen   = screen;
        this.activity = screen.getActivity();
                this.context  = this.activity;
   }else{
        System.out.println("You must define Screen");
                System.exit(-1);
   }

        // URL Initialization
        if(url != null && url.length() > 0){
```

```
                this.url = url;
        }else{
                System.out.println("You must define URL Action");
                System.exit(-1);
        }

        // Set Likely Server Connector Action
        currentAction = SAVE_TABLE_DATA;
}

@Override
protected void onPreExecute() {
        super.onPreExecute();

        // Initialize Progress Dialog
        pDialog = new ProgressDialog(activity);
        pDialog.setIndeterminate(false);
        pDialog.setCancelable(true);

        // Check current action and Call appropriate method
        switch(currentAction){
    case GET_TABLE_SCHEMA :
        getTableSchemaPreExecute();
             break;
    case SAVE_TABLE_DATA :
        saveTableDataPreExecute();
           break;
        }

        // Show Progress Dialog
        pDialog.show();
}

@Override
```

```
protected ArrayList<Hashtable<String, Object>> doInBackground(Hashtable... request) {
        // Getting Array List from URL
    ArrayList<Hashtable<String, Object>>  response = new ArrayList<Hashtable<String, Object>>();

  // Check current action and Call appropriate method
        switch(currentAction){
          case GET_TABLE_SCHEMA :
                D.log("GET_TABLE_SCHEMA");
                response = getTableSchemaRequest(request[0]);
                  break;
          case SAVE_TABLE_DATA :
                D.log("SAVE_TABLE_DATA");
                response = saveTableDataRequest(request[0]);
                break;
        }

        return response;
}

@Override
protected void onPostExecute(ArrayList response) {
        // Check current action and Call appropriate method
        switch(currentAction){
          case GET_TABLE_SCHEMA:
                getTableSchemaResponse(response);
                  break;
          case SAVE_TABLE_DATA:
                saveTableDataResponse(response);
                  break;
        }

        // Close progress Dialog
        pDialog.dismiss();
```

```
        }

        // check internet connection
        private boolean checkConnectionStatus(){
                ConnectivityManager connectivity = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
                 if (connectivity != null) {
                        NetworkInfo[] info = connectivity.getAllNetworkInfo();
           if (info != null){
            for (int i = 0; i < info.length; i++){
                 if (info[i].getState() == NetworkInfo.State.CONNECTED) {
                return true;
              }
            }
           }
        }
        return false;
        }

        // Implement Pre Execute get Table Schema
        private void getTableSchemaPreExecute(){
                if(screen != null){
                        if(checkConnectionStatus() ){
                                // Set Screen's Connection Status true
                                screen.setConnected();

                                // Set Progress Dialog message
                                pDialog.setMessage("Getting data ...");
                        }else{
                                // Set Screen's Connection Status Disconnected
                    screen.setDisconnected();
                        }
                }else{
                        System.out.println("You must define some Screen Template");
```

```
                }
        }

        // Implement Pre Execute saveTable data
        private void saveTableDataPreExecute(){
                if(checkConnectionStatus()){
                        // Set Progress Dialog message
                        pDialog.setMessage("Save Record ...");
                }else{
                        Toast.makeText(context, "You are NOT connetced",
Toast.LENGTH_LONG).show();
                }
        }

        // Implement request get table schema mehtod
        private ArrayList getTableSchemaRequest(Hashtable request){
                // Getting Array List from URL
                ArrayList<Hashtable<String, Object>> response = new
ArrayList<Hashtable<String, Object>>();
                HttpClient client   = new HttpClient(url);

                // Getting response from Server
                response          = client.getHttpResponse(request);

                return response;
        }

        private ArrayList saveTableDataRequest(Hashtable request){
                // Getting Array List from URL
                ArrayList<Hashtable<String, Object>> response = new
ArrayList<Hashtable<String, Object>>();
                HttpClient client   = new HttpClient(url);

                response          = client.getHttpResponse(request);
```

```
        return response;
    }


    // Implement response get table schema method
    private void getTableSchemaResponse(ArrayList response){

        // Checks from response data
        Hashtable error = (Hashtable)response.remove(0);

        if(error.get("success").equals("1")){
            // Get Header Info
            Hashtable header = (Hashtable)response.remove(0);

            // Set Screen's Table Name
            screen.setDBTableName(header.get("table_name").toString());

            // Set Screen Elements
          screen.setScreenElements(response);

            // Create Screen
            screen.createScreen();

            // Set Listeners
            activity.setListeners(screen.getListenerHandler());

            // Set Screen Visible
          activity.setContentView(screen.getRootViewGroup());
        }else{
            Toast.makeText(context,error.get("error").toString()                    ,
Toast.LENGTH_LONG).show();
        }
    }
```

```
// Implement response save table data method
private void saveTableDataResponse(ArrayList response){ D.log(response.toString());

        // Checks from response data
        Hashtable error = (Hashtable)response.remove(0);

        if(error.get("success").equals("1")){
                if(activity.getIntent().getExtras() != null){
                        Intent intent = new Intent();
                        intent.putStringArrayListExtra("serverResponse", response);
                   activity.setResult(activity.RESULT_OK, intent);
                   activity.finish();
                }

                Toast.makeText(context, "Database updated with success!", Toast.LENGTH_LONG).show();

                // clear screen
                screen.clearScreen();
        }else{
                Toast.makeText(context, error.get("error").toString(), Toast.LENGTH_LONG).show();
        }
}

// Set action url
public void setActionURL(String url){
        if(url != null && url.length() > 0){
                this.url = url;
        }else{
                System.out.println("Wrong arguments!");
                System.exit(-1);
        }
}
```

```
        // Set Server Connector Action
        public void setServerConnectorAction(int action){
                switch(action){
                   case GET_TABLE_SCHEMA :
                           currentAction = action;
                              break;
                   case SAVE_TABLE_DATA :
                           currentAction = action;
                              break;
                           default:
                                   System.out.println("There is not such choice");
                                   System.exit(-1);
                }
        }
}
```

4.1.4 EncoderDecoder

```
package com.example.ntua_2014;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.util.zip.GZIPInputStream;
import java.util.zip.GZIPOutputStream;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.annotation.SuppressLint;
import android.util.Base64;

public class EncoderDecoder {

	public static String decodeBase64String(JSONObject jsonobj){
		try{
			return EncoderDecoder.decodeBase64String(jsonobj.toString());
		}catch(Exception e ){
			return "";
		}
	}


//	TODO DECODE STRING
//	@SuppressLint("NewApi")
	public static String decodeBase64String(String in){
		try{
			byte[] bytes = Base64.decode(in,Base64.NO_WRAP) ;

			ByteArrayInputStream in_bytes= new ByteArrayInputStream(bytes);
```

```
        GZIPInputStream gzip = new GZIPInputStream(in_bytes);
        byte[] buffer = new byte[4096];
        int c = 0;
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        while (( c = gzip.read(buffer, 0, 4096)) > 0) {
            out.write(buffer, 0, c);
        }
        out.close();
gzip.close();

return out.toString("UTF-8");
            }catch(Exception e){
                    e.printStackTrace();
            }
    return "";
  }

  @SuppressLint("NewApi")
  public static String encodeBase64String(String in){
            try{
                    ByteArrayOutputStream out= new ByteArrayOutputStream();

GZIPOutputStream gzip = new GZIPOutputStream(out);
gzip.write(in.getBytes("UTF-8"));
gzip.close();

out.close();
//D.log(out.toString());
byte[] bytes = out.toByteArray();

return Base64.encodeToString(bytes,Base64.NO_WRAP );

            }catch(Exception e){
                    e.printStackTrace();
```

```
                }
            return "";
        }

        public static String encodeBase64String(JSONObject jsonobj){
                try{
                        return EncoderDecoder.encodeBase64String(jsonobj.toString());
                }catch(Exception e ){
                        return "";
                }
        }

        public static boolean isJSONValid(String test){
                try{
                        new JSONObject(test);
                }catch(JSONException ex){
                        // e.g. in case JSONArray is valid as well...
              try{
                    new JSONArray(test);
              }catch(JSONException e){
                    return false;
              }
                    }
            return true;
        }
        public static boolean isJSONValid(JSONObject jsonobj){
                try{
                        return EncoderDecoder.isJSONValid(jsonobj.toString());
                }catch(Exception e ){
                        return false;
                }
        }
}
```

4.1.5 HttpClient

```
package com.example.ntua_2014;

import com.example.ntua_2014.D;

import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.zip.GZIPInputStream;
import java.util.zip.GZIPOutputStream;

import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.annotation.SuppressLint;
import android.util.Base64;
```

```
import android.util.Base64OutputStream;
import android.util.Log;

public class HttpClient {

	private static final String TAG = "HttpClient";

	private String url=null;


	public  HttpClient(String url){
		this.url=url;
	}

	public ArrayList<Hashtable<String, Object>> getHttpResponse(Hashtable request) {
		ArrayList<Hashtable<String, Object>> response = new ArrayList<Hashtable<String,
Object>>();

		try{
			// Getting Server Response to JSON Format
			JSONObject reSjsonObj = this.SendHttpPost(new JSONObject(request));

			// Convert JSON to String
			String base64_decoded = reSjsonObj.toString();

			// Pass Server Response to DB Schema Parser
		   DbSchemaParser db_parser = new DbSchemaParser(base64_decoded);

		   // Getting Response From DB Schema Parser
		   response = db_parser.parse();

		}catch (JSONException e) {
			// Set Error
			Hashtable<String, Object> error = new Hashtable<String, Object>();
```

```
                error.put("success", "0");
                error.put("error", e.getMessage());
                response.add(error);
        }

        return response;
    }

    public JSONObject SendHttpPost(JSONObject jsonObjSend) {
        try {
            DefaultHttpClient httpclient = new DefaultHttpClient();
            HttpPost httpPostRequest     = new HttpPost(this.url);

            // Encoding Base64
            String
base64_encoded=EncoderDecoder.encodeBase64String(jsonObjSend.toString());

            StringEntity se = new StringEntity(base64_encoded);

            // Set HTTP parameters
            httpPostRequest.setEntity(se);
            httpPostRequest.setHeader("Accept", "application/json");
            httpPostRequest.setHeader("Content-type", "application/json");
            httpPostRequest.setHeader("Accept-Encoding", "gzip"); // only set this
parameter if you would like to use gzip compression

            // Getting Server Response
            HttpResponse response = (HttpResponse)
httpclient.execute(httpPostRequest);

            // Get hold of the response entity (-> the data):
            HttpEntity entity = response.getEntity();

            if (entity != null) {
                // Read the content stream
```

```
                    InputStream instream = entity.getContent();

                    // convert content stream to a String
                    String resultString = convertStreamToString(instream);
                    instream.close();

                    // Decoding Base64
                    String base64_decoded =
EncoderDecoder.decodeBase64String(resultString);

                    // Convert Server Response to JSON Format
                    JSONObject jsonObjRecv = new JSONObject(base64_decoded);

                    return jsonObjRecv;
                }

            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
        } catch (ClientProtocolException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        } catch (JSONException e) {
                e.printStackTrace();
        }

                return null;
        }

        private static String convertStreamToString(InputStream is) {
                /*
                 * To convert the InputStream to String we use the BufferedReader.readLine()
                 * method. We iterate until the BufferedReader return null which means
                 * there's no more data to read. Each line will appended to a StringBuilder
```

```
                * and returned as String.
                *
                  * (c) public domain: http://senior.ceng.metu.edu.tr/2009/praeda/2009/01/11/a-simple-restful-client-at-android/
                */
                BufferedReader reader = new BufferedReader(new InputStreamReader(is));
                StringBuilder sb = new StringBuilder();

                String line = null;
                try {
                        while ((line = reader.readLine()) != null) {
                                sb.append(line + "\n");
                        }
                } catch (IOException e) {
                        e.printStackTrace();
                } finally {
                        try {
                                is.close();
                        } catch (IOException e) {
                                e.printStackTrace();
                        }
                }
                return sb.toString();
        }
}
```

4.1.6 ScreenTemplate

```
package com.example.ntua_2014;

import java.util.ArrayList;

import android.app.ActionBar.LayoutParams;
import android.content.Context;
import android.content.Intent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;

abstract public class ScreenTemplate {

	// DB Table Name
	protected String tableName	= null;

	// Activity
	protected BaseActivity activity = null;

	// Context
	protected Context context	= null;

	// constants
	protected int match_parent	= LayoutParams.MATCH_PARENT;
	protected int wrap_content	= LayoutParams.WRAP_CONTENT;

	// Spinner data
	protected ArrayList<String> spinnerArray = null;

	// Screen's Elements
	protected ArrayList screenElements	= null;

	// Listener Handler
```

```
protected ListenerHandler listener       = null;

// Internet Connection Status
protected boolean connectionStatus       = false;


public ScreenTemplate(BaseActivity activity){

      // activity initialization
      this.activity = activity;

      // context initialization
      this.context  = activity;

      // listener handler initialization
      listener      = new ListenerHandler();
}

// Creating New Screen
public void createScreen(){

      // Set RootViewGroup
      setRootViewGroup();

      // Creating Screen's Header
      header();

      // Creating Screen's Main Body
      body();

      // Creating Screen's footer
      footer();
}

// Set DB Table Name
```

```
  public void setDBTableName(String tableName){
        this.tableName = tableName;
  }



  // Set Screen Elements ArrayList
  public void setScreenElements(ArrayList screenElements){
        this.screenElements = screenElements;
  }

  // Get Listener Handler
  public ListenerHandler getListenerHandler(){
        return listener;
  }

  // Set Internet Connection Status true
  public void setConnected(){
        this.connectionStatus = true;
  }

  // Set Internet Connection Status false
  public void setDisconnected(){
        this.connectionStatus = false;
  }

  // Get Internet Connection Status
public boolean getConnectionStatus(){
  return this.connectionStatus;
}

  public BaseActivity getActivity(){
        return this.activity;
  }
```

```
    // Clear form's fields after Save command
    public void clearScreen(){ /* Empty method */ }

    // Set Root Layout
    abstract protected void setRootViewGroup();

    // Screen's Header
    abstract protected void header();

    // Screen's Body
    abstract protected void body();

    // Screen's Footer
    abstract protected void footer();

    // Return
    abstract protected ViewGroup getRootViewGroup();

    // Define onclick Listener Method
    abstract public void onClick(View v);

    // Define onItemSelected Listener Method
    abstract public void onItemSeletced(AdapterView<?> adapterView,
int position);

    // Define onActivityResult Method
    abstract public void onActivityResult(int requestCode, int
resultCode, Intent data);

    // Implement onRestart method
    public void onRestart(){ /* Empty method */ }

    // Implement onStart method
    public void onStart()  { /* Empty method */ }
```

```
      // Implement onResume method
    public void onResume() { /* Empty method */ }

      // Implement onPause method
      public void onPause()  { /* Empty method */ }

      // Implement onStop method
      public void onStop()   { /* Empty method */ }

      // Implement onDestroy method
      public void onDestroy(){ /* Empty method */ }
}
```

4.1.7 ScreenParser

```
package com.example.ntua_2014;

import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.List;

import android.app.ActionBar.LayoutParams;
import android.content.Context;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.TextView;




public class ScreenParser {
```

```
	private ArrayList screenElements                = null;

	private LinearLayout.LayoutParams labelParams   = null;

	private LinearLayout.LayoutParams editParams    = null;

	private LinearLayout.LayoutParams spinnerParams = null;

	private ListenerHandler listener                = null;

	public ScreenParser(ArrayList elements){
		this.screenElements = elements;

		// Default LayoutParams
		// set Label common LayoutParams
	  this.labelParams  = new
LinearLayout.LayoutParams(LayoutParams.WRAP_CONTENT,
LayoutParams.WRAP_CONTENT);
	  labelParams.topMargin = 22;

	  // set EditView common LayoutParams
	  this.editParams   = new
LinearLayout.LayoutParams(LayoutParams.WRAP_CONTENT,
LayoutParams.WRAP_CONTENT);
	  editParams.topMargin  = 5;

	  // set Spinner common LayoutParams
	  this.spinnerParams = new
LinearLayout.LayoutParams(LayoutParams.WRAP_CONTENT,
LayoutParams.WRAP_CONTENT);
	  spinnerParams.topMargin  = 5;
	  spinnerParams.width      = 250;
	}

	public ScreenParser(){
		// Default LayoutParams
```

```
            // set Label common LayoutParams
        this.labelParams  = new
LinearLayout.LayoutParams(LayoutParams.WRAP_CONTENT,
LayoutParams.WRAP_CONTENT);
        labelParams.topMargin = 22;

        // set EditView common LayoutParams
        this.editParams   = new
LinearLayout.LayoutParams(LayoutParams.WRAP_CONTENT,
LayoutParams.WRAP_CONTENT);
        editParams.topMargin  = 5;

        // set Spinner common LayoutParams
        this.spinnerParams = new
LinearLayout.LayoutParams(LayoutParams.WRAP_CONTENT,
LayoutParams.WRAP_CONTENT);
        spinnerParams.topMargin  = 5;
        spinnerParams.width      = 250;
      }

      // Set Screen's Elements
      public void setScreenElements(ArrayList elements){
            this.screenElements = elements;
      }

      // Set Label LayoutParams
      public void setLabelParams(LinearLayout.LayoutParams
labelParams){
            this.labelParams = labelParams;
      }

      // Set EditText LayoutParams
      public void setEditParams(LinearLayout.LayoutParams editParams)
{
            this.editParams = editParams;
      }
```

```
	// Set Spinner LayoutParams
	public void setSpinnerParams(LinearLayout.LayoutParams
spinnerParams){
		this.spinnerParams = spinnerParams;
	}

	// Set Listener Handler Object
	public void setListenerHandler(ListenerHandler listener){
		this.listener = listener;
	}

	// Create Screen
	public void createScreen(Context context, LinearLayout
scrollView){
		if(screenElements != null){
			int dropDownID = 1;

			Iterator<Hashtable> itr =
screenElements.iterator();

		  while(itr.hasNext()){
			Hashtable hash = itr.next();

			// Set New Screen's Element
			if(hash.containsKey("type")){
				// Creating New View

	if(((String)hash.get("type")).equals("textView")){
					TextView object = new
TextView(context);
					object.setLayoutParams(labelParams);

					// Set text
				if(hash.containsKey("text")){
```

```
      object.setText((String)hash.get("text"));
                        }

                        scrollView.addView(object);
                    }else
if(((String)hash.get("type")).equals("editView")){
                        EditText object = new
EditText(context);

                        // set field name
                        if(hash.containsKey("text")){
                              object.setTag(hash);
                        }

                        object.setLayoutParams(editParams);
                        object.setEms(11);
                        scrollView.addView(object);
                    }else {
                        Spinner object = new Spinner(context);
                        object.setId(dropDownID++);
                        object.setLayoutParams(spinnerParams);

                        // Set DropDown List's Data
                      if(hash.containsKey("data")){
                        ArrayList<StringWithTag> dropDownData =
(ArrayList)hash.remove("data");

                        // add spinner's data
                        ArrayAdapter<StringWithTag>
spinnerArrayAdapter = new ArrayAdapter<StringWithTag>(context,
android.R.layout.simple_spinner_dropdown_item, dropDownData);
                        object.setAdapter(spinnerArrayAdapter);

                        // set table name
                        object.setTag(hash);
```

```
                                // add spinner's listener

listener.addOnItemSelectedListener(object);
                            }
                            scrollView.addView(object);
                        }
                    }
                }
            }
        }
}
```

4.1.8 EditScreen

```
package com.example.ntua_2014;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Hashtable;
import java.util.Iterator;

import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.Spinner;
import android.widget.TextView;
```

```
import android.widget.Toast;

public class EditScreen extends ScreenTemplate{

	// Root ViewCroup Element
	private	LinearLayout rootLinLayout = null;

	// Element's ViewGroup
	private LinearLayout scrollLinLayout = null;

	public EditScreen(BaseActivity activity){

		// call parent's constructor
		super(activity);
	}

	// set LinearLayout as root element of the screen
	protected void setRootViewGroup(){
		// *********** creating root LinearLayout ***********
	  rootLinLayout = new LinearLayout(context);

	  // specifying vertical orientation
	  rootLinLayout.setOrientation(LinearLayout.VERTICAL);

	  // set LayoutParams
	  LinearLayout.LayoutParams rootParams = new
LinearLayout.LayoutParams(match_parent, match_parent);
	  rootLinLayout.setLayoutParams(rootParams);
	}

	protected ViewGroup getRootViewGroup(){

		return rootLinLayout;
	}
```

```
    protected void header(){
        // ********* creating header LinearLayout ***********
      LinearLayout headerLinLayout = new LinearLayout(context);

      // specifying vertical orientation
      headerLinLayout.setOrientation(LinearLayout.VERTICAL);

      // set LayoutParams
      LinearLayout.LayoutParams headerParams = new
LinearLayout.LayoutParams(match_parent, wrap_content);
      headerLinLayout.setLayoutParams(headerParams);
      headerLinLayout.setGravity(Gravity.CENTER);
      headerLinLayout.setWeightSum(0);

      // add header LinearLayout to root LinearLayout
      rootLinLayout.addView(headerLinLayout);

      // creating header LinearLayout elements

      // Is Connected Label
      TextView isConnected = new TextView(context);
      isConnected.setLayoutParams(new
LinearLayout.LayoutParams(wrap_content, wrap_content));
      isConnected.setTextSize(15);
      isConnected.setTextColor(Color.parseColor("#FFFFFF"));
      if(connectionStatus){
          isConnected.setText("You are connected");

    isConnected.setBackgroundColor(Color.parseColor("#FF00CC00"));
      }else{
          isConnected.setText("You are NOT connected");

    isConnected.setBackgroundColor(Color.parseColor("#FF0000"));
      }
      headerLinLayout.addView(isConnected);
```

```
        // set Main Label LayoutParams
        LinearLayout.LayoutParams mainParams = new
LinearLayout.LayoutParams(wrap_content, wrap_content);

        // Screen's Main Label
        TextView mainLabel = new TextView(context);
        mainParams.topMargin = 30;
        mainLabel.setLayoutParams(mainParams);
        mainLabel.setTextAppearance(context,
android.R.style.TextAppearance_Large);
        mainLabel.setText("Add New "+tableName);
        headerLinLayout.addView(mainLabel);
        // ************ Ending header LinearLayout **********
      }

      protected void body(){
            // set Label common LayoutParams
        LinearLayout.LayoutParams labelParams  = new
LinearLayout.LayoutParams(wrap_content, wrap_content);
        labelParams.topMargin = 22;

        // set EditView common LayoutParams
        LinearLayout.LayoutParams editParams   = new
LinearLayout.LayoutParams(wrap_content, wrap_content);
        editParams.topMargin  = 5;

        // set Spinner common LayoutParams
        LinearLayout.LayoutParams spinnerParams = new
LinearLayout.LayoutParams(wrap_content, wrap_content);
        spinnerParams.topMargin  = 5;
        spinnerParams.width      = 250;

        // ************** creating ScrollLayout *************
        ScrollView scrollView = new ScrollView(context);
```

```
        // set LayoutParams
        LinearLayout.LayoutParams scrollParams = new
LinearLayout.LayoutParams(match_parent, wrap_content);
        scrollParams.weight = 1;
        scrollView.setLayoutParams(scrollParams);

        // add ScrollView to root LineraLayout
        rootLinLayout.addView(scrollView);

        // ********* creating ScrollVew LinearLayout ********
        scrollLinLayout = new LinearLayout(context);

        // specifying vertical orientation
        scrollLinLayout.setOrientation(LinearLayout.VERTICAL);

        // set LayoutParams
        scrollLinLayout.setLayoutParams(new
LinearLayout.LayoutParams(match_parent, wrap_content));
        scrollLinLayout.setPadding(20, 0, 5, 0);

        // add scroll LinearLayout to ScrollView
        scrollView.addView(scrollLinLayout);



          // Screen Parser Initialization
        ScreenParser screen = new ScreenParser(this.screenElements);
        screen.setEditParams(labelParams);
        screen.setEditParams(editParams);
        screen.setSpinnerParams(spinnerParams);

        // Set Screen Parser ListenerHandler
        screen.setListenerHandler(listener);
```

```
        // Screen Creation
        screen.createScreen(context, scrollLinLayout);
    }

    protected void footer(){
        // ********** creating bottom LinearLayout **********
        LinearLayout bottomLinLayout = new LinearLayout(context);

        // specifying horizontal orientation
        bottomLinLayout.setOrientation(LinearLayout.HORIZONTAL);

        // set LayoutParams
        LinearLayout.LayoutParams bottomParams = new
LinearLayout.LayoutParams(match_parent, wrap_content);
        bottomParams.topMargin = 5;
        bottomLinLayout.setBackgroundColor(Color.parseColor("#3F3C53"
));
        bottomLinLayout.setLayoutParams(bottomParams);
        bottomLinLayout.setPadding(5, 5, 5, 5);
        bottomLinLayout.setGravity(Gravity.CENTER_HORIZONTAL);
        bottomLinLayout.setWeightSum(0);

        // add bottom LinearLayout to root LinearLayout
        rootLinLayout.addView(bottomLinLayout);

        // creating bottom LinearLayout elements

        // set bottom commons LayoutParams
        LinearLayout.LayoutParams commonBottomParams = new
LinearLayout.LayoutParams(100, 40);
        commonBottomParams.rightMargin = 10;

        // creating save button
        Button saveBtn = new Button(context);
        saveBtn.setText("Save");
```

```
        saveBtn.setLayoutParams(commonBottomParams);
        listener.addOnClickListener(saveBtn);
        bottomLinLayout.addView(saveBtn);

        // creating cancel button
        Button cancelBtn = new Button(context);
        cancelBtn.setText("Cancel");
        cancelBtn.setLayoutParams(commonBottomParams);
        listener.addOnClickListener(cancelBtn);
        bottomLinLayout.addView(cancelBtn);
        // ************ Ending bottom LinearLayout **********
    }

    // Implement onClick Listener Method
    public void onClick(View v){
        //do something
        Button btn = (Button)v;
        String btnText = btn.getText().toString();

        if(btnText == "Cancel"){
            activity.finish();
        }else if(btnText == "Save"){
            if(scrollLinLayout != null){
                // set action url
                String url =
"http://10.0.2.2/ntua_diplomatiki/server/index.php?
r=Database/insertRecord";

                // validation flag
                boolean valid = true;

                // posted data hashtable
                Hashtable request = new Hashtable();

                // set action's table name
```

```
                        request.put("table_name", tableName);

                        // posted fields
                        Hashtable<String, String> fields = new
Hashtable<String, String>();

                        // validator
                        Validator validator = new Validator();

                        ViewGroup group = (ViewGroup)scrollLinLayout;
                        for (int i = 0, count =
group.getChildCount(); i < count; ++i) {
                            View view = group.getChildAt(i);

                            if(view instanceof EditText){
                              EditText editText = (EditText)view;
                              Hashtable tag =
(Hashtable)editText.getTag();

                              if(validator.valid(editText, tag)){
                                    if(tag.containsKey("text"))

      fields.put(tag.get("text").toString(),
editText.getText().toString());
                              }else{

      editText.setError(validator.getErrorMessage());
                                    valid = false;
                              }
                            }else if(view instanceof Spinner){
                              Spinner spinner = (Spinner)view;

                              StringWithTag strTag = (StringWithTag)
spinner.getSelectedItem();
                                  Object id = strTag.tag;
```

```
                                String excluded_values[] = {"-1",
"-2"};
                                if(validator.valid(spinner,
excluded_values)){

                                  if(spinner.getTag() != null){

fields.put(((Hashtable)spinner.getTag()).get("referer_column_name").t
oString(), strTag.tag.toString());
                                  }
                                }else{

                                  valid = false;
                                }
                            }
                        }

                        if(valid){
                              // add fields data into request
Hashtable
                              request.put("fields", fields);

                              // initialize server connector and set
its action type
                              ServerConnector server = new
ServerConnector(this, url);

server.setServerConnectorAction(ServerConnector.SAVE_TABLE_DATA);
                              server.execute(request);
                        }else{
                              Toast.makeText(context, "Please correct
your errors and try again.", Toast.LENGTH_SHORT).show();
                        }
                    }else{
```

```
                        Toast.makeText(activity, "You must define
Element's ViewGroup", Toast.LENGTH_LONG).show();
                        System.out.println("You must define Element's
ViewGroup");
                        System.exit(-1);
                  }
            }
      }

      // Implement onItemSelected Listener Method
      public void onItemSeletced(AdapterView<?> adapterView, int
position){
            Spinner spinner = (Spinner) adapterView;
            StringWithTag strTag = (StringWithTag)
adapterView.getItemAtPosition(position);
          Object id = strTag.tag;

          // If selected add new item
            if(id.equals("-1")){
                  Intent intent = new Intent(activity,
MainActivity.class);
                  intent.putExtra("ActionDBTable", (String)
((Hashtable)spinner.getTag()).get("table_name"));
                  activity.startActivityForResult(intent, 1);
            }
      }

      // Implement onActivityResult Method
      public void onActivityResult(int requestCode, int resultCode,
Intent data){
            if (data == null) {return;}

            ViewGroup group = (ViewGroup)scrollLinLayout;
            for (int i = 0, count = group.getChildCount(); i < count;
++i) {
                View view = group.getChildAt(i);
```

```
                if(view instanceof Spinner){

                  Spinner spinner = (Spinner)view;
                  ArrayList<HashMap> serverResponse =
(ArrayList)data.getExtras().getStringArrayList("serverResponse");

                  // Add new record one position before end
                      if(serverResponse != null){
                        Iterator<HashMap> itr =
serverResponse.iterator();
                        while(itr.hasNext()){
                              HashMap refreshData = itr.next();
                              if(spinner.getTag() != null &&

((Hashtable)spinner.getTag()).get("table_name").toString().equals(ref
reshData.get("table_name").toString())){
                                    ArrayAdapter<StringWithTag>
spinnerAdapter = (ArrayAdapter<StringWithTag>)spinner.getAdapter();
                                    spinnerAdapter.insert(
                                          new
StringWithTag(refreshData.get("name").toString(),
refreshData.get("id").toString()),

      spinnerAdapter.getCount() - 1);

      spinnerAdapter.notifyDataSetChanged();
                              }
                        }
                      }
                }
            }
      }

      // Clear form's fields after Save command
      public void clearScreen(){
```

```
            ViewGroup group = (ViewGroup)scrollLinLayout;

            for (int i = 0, count = group.getChildCount(); i < count;
++i) {
                View view = group.getChildAt(i);

                if(view instanceof EditText){
                    ((EditText)view).setText("");
                }else if(view instanceof Spinner){
                  ((Spinner)view).setSelection(0);
                }
            }
      }

      // Implement onPause method
    public void onStop(){
      ViewGroup group = (ViewGroup)scrollLinLayout;

            for (int i = 0, count = group.getChildCount(); i < count;
++i) {
                View view = group.getChildAt(i);

                if(view instanceof Spinner){
                  ((Spinner)view).setSelection(0);

((Spinner)view).setBackgroundColor(Color.parseColor("#ffffff"));
                }
            }
    }

}
```

4.1.9 ListenerHanlder

```
package com.example.ntua_2014;

import java.util.ArrayList;

import android.view.View;

public class ListenerHandler {

	// onItemSelected Objects
	private ArrayList onItemSelectedObj = null;

	// onClick objects
	private ArrayList onClickObj        = null;

	// ListenerHandler Constructor
	public ListenerHandler(){

		onItemSelectedObj = new ArrayList();

		onClickObj        = new ArrayList();
	}

	// Add Subject To onItemSelected Observer
	public void addOnItemSelectedListener(View v){
		onItemSelectedObj.add(v);
	}

	// Add Subject To onClick Observer
	public void addOnClickListener(View v){
		onClickObj.add(v);
	}

	// Get All onItemSelected Observer Subjects
```

```
	public ArrayList getOnItemSelectedListener(){
		return onItemSelectedObj;
	}

	// Get All onClick Observer Subjects
	public ArrayList getOnClickListener(){
		return onClickObj;
	}
}
```

4.1.10 Validator

```
package com.example.ntua_2014;

import java.util.Hashtable;

import android.graphics.Color;
import android.widget.EditText;
import android.widget.Spinner;

public class Validator {

	// Minimum string length
	public static final int MINIMUM_STRING_LENGTH  = 3;

	private String error;

	public Validator(){

		error = null;
	}

	public boolean valid(EditText edit, Hashtable dbAttr){
		if(edit.getText() == null ||
edit.getText().toString().length() == 0){
			error = "This field cannot be blank.";
```

```
                return false;
            }

            if(dbAttr.containsKey("dataType")){

      if(dbAttr.get("dataType").toString().equals("varchar")){
                        return
varcharValidation(edit.getText().toString(), dbAttr);
                }else
if(dbAttr.get("dataType").toString().equals("int")){
                        return
integerValidation(edit.getText().toString(), dbAttr);
                }
            }

            return true;
      }

      public boolean valid(Spinner spinner, String excluded[]){
            StringWithTag strTag = (StringWithTag)
spinner.getSelectedItem();
          String id = strTag.tag.toString();

          // set background color white
          spinner.setBackgroundColor(Color.parseColor("#ffffff"));

          for(int i=0; i<excluded.length;i++){
                if(id.equals(excluded[i])){
                      // set background color red

      spinner.setBackgroundColor(Color.parseColor("#f2184b"));
                        error = "Invalid Selected Item.";
                        return false;
                }
            }
            return true;
```

```
	}

	private boolean varcharValidation(String data, Hashtable
dbAttr){
		if(data.length() < MINIMUM_STRING_LENGTH){
			error = "Minimum Character Length.";
			return false;
		}

		if(dbAttr.containsKey("maxLength") &&
Integer.parseInt(dbAttr.get("maxLength").toString()) < data.length())
{
			error = "Maximum Character Length Exceeded.";
			return false;
		}
		return true;
	}

	// This method checks if a String contains only numbers
	private boolean integerValidation(String data, Hashtable
dbAttr){
		if(dbAttr.containsKey("maxLength") &&
Integer.parseInt(dbAttr.get("maxLength").toString()) < data.length())
{
			error = "Maximum Character Length Exceeded.";
			return false;
		}

	  for (int i = 0; i < data.length(); i++) {

		//If we find a non-digit character we return false.
		if (!Character.isDigit(data.charAt(i))){
			error = "This field must be numeric.";
D.log(error);
		    return false;
		}
```

```
        }

        return true;
      }

      // return error message
      public String getErrorMessage(){
            return error;
      }
}
```

4.1.11 StringWithTag

```
package com.example.ntua_2014;

public class StringWithTag {
      public String string;
    public Object tag;

    public StringWithTag(String stringPart, Object tagPart) {
        string = stringPart;
        tag = tagPart;
    }

    @Override
    public String toString() {
        return string;
    }
}
```

4.1.12 D

```
class D {
    //put your code here

public static function dd($in,$flag=null){
    $trace=  debug_backtrace();

    $out=array(
        'file'=>$trace[0]['file'],
        'line'=>$trace[0]['line'],
        'data'=>$in
    );
                echo    "<pre    style='color:black;    z-index:1000;
position:relative;'>";
        print_r($out);
    echo "</pre>";
    if(isset($flag) && (string)$flag==='1'){die;}
}

}
```

4.1.13 DatabaseConnection

```
class DatabaseConnection
{
    /**
     * Host name
     * @access private
     * @static
     * @var string
     */
    private static $host;
```

```
/**
 * The MySQL user name
 * @access private
 * @static
 * @var string
 */
private static $username;

/**
 * The MySQL user's password
 * @access private
 * @static
 * @var string
 */
private static $password;

/**
 * The MySQL Database
 * @access private
 * @static
 * @var string
 */
private static $database;

/**
 * Static reference to database connection
 * @access private
 * @static
 */
private static $dbConn = null;

/**
 * Reference to database connection
 * @access private
 * @var mysqli object
```

```
     */
    private $connection;

    /**
     * DatabaseConnection constructor
     *
     * Makes the connection to database and initializes
     * the connection handler
     *
     * @access private
     * @return void
     */
    private function __construct()
    {
        //self::loadCredentials();
                    $this->connection   =   new   mysqli(database_host,
database_username, database_password, database_name);
    }

    /**
     * Credentials loader
     *
     * Loads the right credentials depending the server
     *
     * @access private
     * @static
     * @staticvar string $host a server's hostname
     * @staticvar string $username a username for that database
     * @staticvar string $password a password for that database
     * @staticvar string $database name of the database for the above
credentials
     * @return void
     */
    private static function loadCredentials()
    {
```

```
        self::$host = database_host;
        self::$username = database_username;
        self::$password = database_password;
        self::$database = database_name;
    }
    /**
     * Returns the database handler
     *
     * if a handler is found is returned, otherwise
     * a new one is instantiated
     *
     * @access public
     * @static
     * @staticvar object $dbConn a reference to the database
     * @return mysqli object
     */
    public static function getDBHandler()
    {
        if ( self::$dbConn == null )
        {
            self::$dbConn = new self();
        }
        return self::$dbConn->connection;
    }

}
```

4.1.14 EncoderDecoder

```
class EncoderDecoder {

    //$data: string input
    //Output returns: json  string
    public static function decode($data) {
        $gzipped_msg = base64_decode($data); //D::dd($gzipped_msg);
        if (empty($gzipped_msg)) {
```

```
                return '{"success": 0,"error": "Data not base64
encoded" }';
        }
         $json = gzinflate(substr($gzipped_msg, 10, -8)) . PHP_EOL .
PHP_EOL;

        return $json;
    }

    //$data: json string
    //Output returns: string
    public static function encode($json_message) {
        $gziped = gzcompress($json_message, 9, ZLIB_ENCODING_GZIP);
        $encoded_msg = base64_encode($gziped);
        return $encoded_msg;
    }
}
```

4.1.15 Helper

```
class Helper {
   //put your code here

   public static function array_map_deep($callback,$array) {
      $new = array();
      if (is_array($array))
         foreach ($array as $key => $val) {
            if (is_array($val)) {
               $new[$key] = Helper::array_map_deep( $callback,$val);
            } else {
               $new[$key] = call_user_func($callback, $val);
            }
         } else
         $new = call_user_func($callback, $array);
      return $new;
```

```
    }

}
```

4.1.15 EncoderDecoder

```
class BaseController {

    //put your code here
    protected $options = array();
    protected $post = array();

    public function __construct($in = array()) {
        $this->options = $in;
        $data = file_get_contents('php://input');
        if (isset($data) && $data) {
            $json = EncoderDecoder::decode($data);
            $array = json_decode($json, true);
            if ($array === null || (isset($data['success'])&&(string) $data['success'] === '0')) {
                return;
            }

            $this->post = $array;
//            D::dd($this->post);
        }
    }

}
```

4.1.16 EncoderDecoder

```
class DatabaseController extends BaseController {
```

```
    //put your code here
    private $query = null;
    private $response = array();

    public function insertField() {

            $sql = 'ALTER TABLE `' . $this->post['table_name'] . '` ADD `' . $this->post['field_name'] . '` ';

        if (isset($this->post['type']) && $this->post['type'] && isset($this->post['length']) && $this->post['length']) {
          $sql.=' ' . $this->post['type'] . '( ' . $this->post['length'] . ' )';
        }
        $sql.=' NOT NULL ';
        $this->query = new Query($sql, array(), '');
    }

    public function updateField() {
            //ALTER TABLE `contacts` CHANGE `email` `email` VARCHAR( 50 ) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL ;
            $sql = 'ALTER TABLE `' . $this->post['table_name'] . '` CHANGE `' . $this->post['field_name'] . '` `' . $this->post['field_name'] . '` ' . $this->post['rest_sql'];

        $this->query = new Query($sql, array(), '');
    }

    public function deleteField() {
        //ALTER TABLE `contacts` DROP `email` ;
            $sql = 'ALTER TABLE `' . $this->post['table_name'] . '` DROP `' . $this->post['field_name'] . '` ';
        $this->query = new Query($sql, array(), '');
```

```
    }

    public function insertRecord() {
        //INSERT INTO `ntua_diplomatiki`.`contacts` (`id`, `name`, `email`) VALUES (NULL, 'test', 'hi@ole.gr');
        $sql = 'INSERT INTO `ntua_diplomatiki`. `' . $this->post['table_name'] . '` ';
        $sql .= '(  `' . implode('`,`', array_keys($this->post['fields'])) . '`  ) VALUES';
        $sql .= "(  '" . implode("','", array_values($this->post['fields'])) . "'  );";
        $this->query = new Query($sql, array(), ''); //D::dd($this->query->printQuery());
    }

    public function updateRecord() {
            //UPDATE  `ntua_diplomatiki`.`contacts` SET  `name` =  'mitsos1111',`email` = 'ole@hi.gr' WHERE  `contacts`.`id` =4;
        $sql = 'UPDATE  `ntua_diplomatiki`. `' . $this->post['table_name'] . '`  SET ';
        $fields = array();
        foreach ($this->post['fields'] as $key => $value) {
            array_push($fields, " `$key` =  '$value'   ");
        }
        $sql .= implode(' , ', $fields);
        $sql .= ' WHERE ' . $this->post['where_sql'];
        $this->query = new Query($sql, array(), '');
    }

    public function deleteRecord() {
        //DELETE FROM `ntua_diplomatiki`.`contacts` WHERE `contacts`.`id` = 4;
        $sql = 'DELETE FROM  `ntua_diplomatiki`. `' . $this->post['table_name'] . '`  WHERE ' . $this->post['where_sql'];
        $this->query = new Query($sql, array(), '');
    }

    public function insertTable() {
```

```
        //CREATE TABLE example_autoincrement (id INT NOT NULL AUTO_INCREMENT PRIMARY KEY);
        $sql = 'CREATE TABLE  `ntua_diplomatiki`. `' . $this->post['table_name'] . '(id INT NOT NULL AUTO_INCREMENT PRIMARY KEY);';
        $this->query = new Query($sql, array(), '');
    }

    public function deleteTable() {
        //DROP TABLE table_name ;
        $sql = 'DROP TABLE  `ntua_diplomatiki`. `' . $this->post['table_name'] . ';';
        $this->query = new Query($sql, array(), '');
    }

    private function get_response($success = null) {
        if ((string) $success === '1') {
            $this->response['success'] = '1';
            $this->response['data'] = array(
                'table_name'   => $this->post['table_name'],
                'on_save_data' => array(
                                $this->query->getLastInsertId() => isset( $this->post['fields'][$this->getField('name')]) ? $this->post['fields'][$this->getField('name')] : '',
                ),
            );

            $this->response['error'] = null;
        } else {
            $this->response['success'] = '0';
            $this->response['data'] = array();
                        $this->response['error'] = $this->post['table_name'] . ':' . $this->query->queryFailured();
        }
    }
```

```
    private function get_db_error() {
        $r = $this->query->queryFailured();
        if ((string) $r === '' || !$r) {
            $this->get_response('1');
        } else {
            $this->get_response();
        }
        return EncoderDEcoder::encode(json_encode($this->response));
    }

    private function getField($field) {
        if (isset($this->post['fields'][ucfirst($field)])){
            return ucfirst($field);
        }
        else {
            return $field;
        }
    }

    public function __destruct() {
        echo $this->get_db_error();
    }

}
```

4.1.17 dbSchemaCreatorController

```
class dbSchemaCreatorController extends BaseController {

    private $dbname = null;
```

```
    private $results = array();

    public function __construct($bypass = null) {
        if (empty($bypass)) {
            parent::__construct();
        }
        $this->dbname = database_name;
    }

    public function getTables() {


        $results = array();
        $sql = "SHOW TABLES";
        $query = new Query($sql, array(), '', 'S');
        $rows = $query->getResults();
        if (is_array($rows) && count($rows) > 0) {
            foreach ($rows as $row) {
                $results[current($row)] = $this->getTableStructure(current($row));
            }
        }

        return $results;
    }

    public function getTableStructure($table = null) {
        if (!empty($table)) {

        } else if (!empty($this->post['table_name'])) {
            $table = $this->post['table_name'];
        } else {
//            $this->results=array('success'=>0,'error' => 'Empty table name');//return;
```

```
        $table = init_table_name;
    }
    $structure = array();
    $fields = array();

//          $sql = "SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_SCHEMA =? AND TABLE_NAME = ?";
        $sql = "SELECT sc.*,kcu.referenced_table_name,referenced_column_name FROM INFORMATION_SCHEMA.COLUMNS AS sc LEFT JOIN INFORMATION_SCHEMA.key_column_usage AS kcu
                                ON sc.TABLE_SCHEMA=kcu.TABLE_SCHEMA AND sc.TABLE_NAME=kcu.TABLE_NAME AND sc.COLUMN_NAME=kcu.COLUMN_NAME AND KCU.referenced_table_name is not null
    WHERE  sc.TABLE_SCHEMA =? AND sc.TABLE_NAME =?";
    $query = new Query($sql, array($this->dbname, $table), '', 'S');
    $rows = $query->getResults(); //D::dd($rows);
    if (is_array($rows) && count($rows) > 0) {
        foreach ($rows as $row) {
            $fields[ucfirst($row['COLUMN_NAME'])] = array(
                "order" => $row['ORDINAL_POSITION'],
                "null" => $row['IS_NULLABLE'],
                "type" => $row['DATA_TYPE'],
                "max_length" => $row['CHARACTER_MAXIMUM_LENGTH'],
            );
            $structure['table_name'] = ucfirst($row['TABLE_NAME']);
            if ($row['COLUMN_KEY'] == 'PRI') {
                $structure['primary_key'] = ucfirst($row['COLUMN_NAME']);
            }
                                    if (!empty($row['referenced_table_name']) && !empty($row['referenced_column_name'])) {
                if (isset($row['referenced_table_name'])) {
                    $structure['referenced_table_name'] = ucfirst($row['referenced_table_name']);
                }
```

```
                $structure['referer_column_name'] = ucfirst($row['COLUMN_NAME']);
                $structure['referenced_table_data'] = $this->getTableData($row['referenced_table_name']);
            }
        }
    }
    unset($fields[$structure['primary_key']]);
    if (isset($structure['referer_column_name'])) {
        unset($fields[ucfirst($structure['referer_column_name'])]);
    }
    $structure['fields'] = $fields;
    $this->results = $structure;
    return $structure;
  }

  private function getTableData($table = null) {
    $sql = "SELECT id,name FROM `$table` WHERE 1";
    $query = new Query($sql, array(), '', 'S');
    $rows = $query->getResults();
    $r = array();
    if (is_array($rows) && count($rows) > 0) {
        foreach ($rows as $v) {
//            array_push($r, $v['name']);
            $r[$v['id']] = $v['name'];
        }
    }
    $r[-1] = "New " . ucfirst($table);
    return $r;
  }

  public function __destruct() {
    if (empty($this->results)) {
```

```
            $results = array('success' => 0, 'error' => 'Empty results');
        } else {
            $results = array('success' => 1, 'data' => $this->results);
        }

        echo EncoderDecoder::encode(json_encode($results));
        return;
    }

//$r = $this->getTableStructure(database_name, 'contacts');
//D::dd($r);
}
```

4.1.18 Router.php

```
function __autoload($className) {

    $folder_array = array(
        'models',
        'controllers',
        'Helper',
    );
    $force_include_files = array(
        'D',
        'EncoderDecoder',
    );

    if (!in_array($className, $force_include_files)) {
        array_push($force_include_files, $className);
    }
```

```
    foreach ($force_include_files as $clsname) {
        foreach ($folder_array as $folder) {
            $file = SERVER_ROOT . $folder . '/' . $clsname . '.php';
            if (file_exists($file)) {//include file
                include_once($file);
                break;
            }
        }
    }

    return false;
}

if (isset($_GET['r'])) {
    $router = explode("/", $_GET['r']); //echo "<pre>"; print_r($this->router); echo "</pre>";
    if (array_key_exists(0, $router)) {
        $controller = ucfirst($router[0]) . 'Controller';
        if (array_key_exists(1, $router)) {
            $action = $router[1];
            if (array_key_exists(2, $router)) {
                if (strpos($router[2], '|') !== false) {
                    $action_id = explode('|', $router[2]);
                } else {
                    $action_id = $router[2];
                }
                $action_id = split_by_key($action_id);
            }
        } else {
            $action = 'invoke';
        }
    }
}
```

```
//check if file, class, method exists

$controller_route = array(
    'controllers',
);
foreach ($controller_route as $route) {

    $path = $route . '/' . $controller . '.php';

    if (file_exists($path)) {

        include_once $path;

        if (class_exists($controller)) {
            if (isset($action_id)) {
                $class = new $controller($action_id);
            } else {
                $class = new $controller();
            }
//D::dd(array($class,$action),1);
            if (method_exists($class, $action)) {
                if (isset($action_id)) {
                    $class->$action($action_id);
                } else {
                    $class->$action();
                }
                $page_found = 1;
            }
        }
```

```
        break;
    }
}

//D::dd($path,1);
//redirect if not exists
if (!isset($page_found) || ($page_found !== 1)) {
    header('Location: index.php?r=pagenotfound');
}

//echo "$controller - $action - $action_id - $page_found";



function split_by_key($action_id) {

    $new_action_id = array();

    if (is_array($action_id)) {
        $i = 0;
        foreach ($action_id as $value) {
            if (strpos($value, '=') !== false) {
                $action_id_split = explode('=', $value);
                $new_action_id[$action_id_split[0]] = $action_id_split[1];
            } else {
                $new_action_id[$i] = $value;
            }
            $i++;
        }
    } else {
        if (strpos($action_id, '=') !== false) {
            $action_id_split = explode('=', $action_id);
```

```
            $new_action_id[$action_id_split[0]] = $action_id_split[1];
        }
    }

    if (!empty($new_action_id)) {
        return $new_action_id;
    } else {
        return $action_id;
    }
}
```

4.1.19 index.php

```
session_start();
header("content-type:text/html;charset=utf-8");
//error_reporting(E_ALL ^ E_NOTICE); //error_reporting(1);
error_reporting(E_ALL); //error_reporting(1);

include_once 'config/constants.php';
require_once(SERVER_ROOT . 'Router.php');
session_write_close();
```

5 Βιβλιογραφία

[JS12] Jonathan Simon, Head First Andoid Development, 2012

[EEBK04] Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Siera, Head First Design Patterns, 2004

[Eck06] Bruce Eckel, Thinking in JAVA 4th Edition, 2006

[JJQV98] http://developer.android.com/index.html
http://startandroid.ru/en/lessons/complete-list.html