ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή εργασία

Ανάπτυξη εφαρμογής αποστολής σύντομων ειδοποιήσεων για την πλατφόρμα Android

NotifyMe

Κοτσυφός Στυλιανός (ΑΜ:1771)
E-mail: stelioskotsifos@gmail.com

Ηράκλειο-Οκτώβριος 2013

Επιβλέπων Καθηγητής : Δρ. Αθανάσιος Μαλάμος

Abstract

The Android platform offers numerous and diverse applications with which users can communicate with each other, using video, text messages or voice and many other ways.

The purpose of this thesis is to create a new application for the Android platform. This application allows users, with the touch of a button, to send short messages to each other via the internet (if the recipient is connected) or through the GSM network, using SMS. (In case that the recipient is not connected) The main administration of data is performed by a Java server, which is responsible for transmitting the requested data to the appropriate Android phone. Finally, the user has the option to make a phone call to the contact of his choice, if he decides to. The approach followed in the current work includes initially an extensive review of the relevant technologies. Afterwards, the proposed system requirements are clearly set and analyzed.

Σύνοψη

Η Android πλατφόρμα προσφέρει πολλές και ποικίλες εφαρμογές με τις οποίες οι χρήστες μπορούν να επικοινωνούν μεταξύ τους. Είτε με χρήση video, είτε μέσω μηνυμάτων ή φωνής, αλλά και με πολλούς άλλους τρόπους.

Σκοπός αυτής της πτυχιακής είναι η δημιουργία μιας νέας εφαρμογής, για την πλατφόρμα Android. Αυτή η εφαρμογή προσφέρει τη δυνατότητα στους χρήστες της να μπορούν, με το πάτημα ενός κουμπιού, να στέλνουν σύντομες ειδοποιήσεις μεταξύ τους, μέσω του διαδικτύου (αν ο παραλήπτης είναι συνδεδεμένος) ή μέσω του δικτύου GSM, με χρήση SMS(στην περίπτωση που ο παραλήπτης δεν είναι συνδεδεμένος. Η όλη διαχείριση δεδομένων γίνεται κεντρικά από έναν Java server, ο οποίος είναι υπεύθυνος για μεταφορά των ζητούμενων δεδομένων στην κατάλληλη συσκευή. Τέλος, ο χρήστης έχει και την επιλογή να πραγματοποιήσει και τηλεφωνική κλήση προς την επαφή της επιλογής του, αν το θελήσει. Η προσέγγιση που ακολουθείται σε αυτή την εργασία, περιλαμβάνει αρχικά μια εκτενή επισκόπηση των σχετικών τεχνολογιών. Στη συνέχεια, καθορίζονται και αναλύονται οι προτεινόμενες απαιτήσεις του συστήματος.

Περιεχόμενα

Πίνακας Εικόνων

Λίστα Πινάκων

Εισαγωγή

Περίληψη

Στην πτυχιακή αυτή γίνεται ανασκόπηση της Android πλατφόρμας, με σκοπό την καλύτερη κατανόηση του «Τι είναι το Android». Επίσης, γίνονται ανάλυση απαιτήσεων, μελέτη καθώς και η υλοποίηση για μια εφαρμογή ανταλλαγής μηνυμάτων, η οποία έχει ως κύριο στόχο την γρήγορη και άμεση επικοινωνία, για να επιτευχθεί η μέγιστη ικανοποίηση των χρηστών, καθώς αυτοί είναι που θα διατηρήσουν μια εφαρμογή ζωντανή ή θα την απαξιώσουν περνώντας τη στην αχρηστία.

Σκοπός και Στόχοι Εργασίας

Σκοπός της εργασίας αυτής, είναι η τριβή και η ενασχόληση με την γλώσσα προγραμματισμού Java σαν εργαλείο υλοποίησης εφαρμογών γενικότερα αλλά κυρίως για την πλατφόρμα Android και με αρχιτεκτονική client-server.

Στόχος της εκπόνησης αυτής της εργασίας είναι η απόκτηση εμπειρίας σε θέματα υλοποίησης εφαρμογών. Πιο συγκεκριμένα ο στόχος που επιδιώκεται είναι μετά το πέρας της εργασίας αυτής, να μπορούν να δημιουργηθούν εφαρμογές client-server από την αρχή (τη σύλληψη της ιδέας) μέχρι και την υλοποίησή τους.

Ανασκόπηση Android Πλατφόρμας

Εισαγωγή

Στο κεφάλαιο αυτό, θα γίνει μια γνωριμία με την Android πλατφόρμα και θα παρουσιαστούν τα βασικά χαρακτηριστικά αυτής.

Ιστορικά στοιχεία

Android Inc.

Η Εταιρία Android A.E. ιδρύθηκε στην Καλιφόρνια των Ηνωμένων Πολιτειών τον Οκτώβριο του 2003 από τον Andy Rubin (συν-ιδρυτής της εταιρίας Danger), τον Rich Miner (συν-ιδρυτής της εταιρίας Wildfire Communications), τον Nick Sears (πρώην αντιπρόεδρος της T-Mobile), και τον Chris White (επικεφαλής σχεδιασμού και ανάπτυξης διεπαφής στην WebTV) για την ανάπτυξη, σύμφωνα με τα λόγια του Rubin «εξυπνότερων κινητών συσκευών που είναι περισσότερο ενήμερες για την τοποθεσία και προτιμήσεις του ιδιοκτήτη τους». Οι αρχικές προθέσεις της εταιρίας ήταν να αναπτύξουν ένα προηγμένο λειτουργικό σύστημα για ψηφιακές κάμερες. Όταν συνειδητοποίησαν ότι αυτή η αγορά δεν ήταν αρκετά μεγάλη, άλλαξαν σχέδιο δράσης και αποφάσισαν να δημιουργήσουν ένα λειτουργικό για smartphone, το οποίο θα αντιτίθετο στα Symbian και Windows Mobile. (το iPhone της Apple δεν είχε δημοσιευτεί εκείνη την περίοδο) Παρά τα προφανή επιτεύγματα των ιδρυτών και των πρώτων υπαλλήλων της, η Android A.E. λειτουργούσε κρυφά, αποκαλύπτοντας μόνο ότι δούλευε σε λογισμικό για κινητά τηλέφωνα. Την ίδια χρονιά, ο Rubin χρεοκόπησε. Ο Steve Perlman , στενός φίλος του Rubin, του έδωσε $10.000 σε μετρητά σε ένα φάκελο και αρνήθηκε να συμμετέχει στη χρηματοδότηση της εταιρείας.

Η Google αγόρασε την Android A.E. στις 17 Αυγούστου 2005 μετατρέποντας την σε μια εξολοκλήρου θυγατρική της Google. Βασικοί εργαζόμενοι της Android A.E., συμπεριλαμβανομένων των Rubin, Miner και White, παρέμειναν στην εταιρία και μετά την εξαγορά. Εκείνη την περίοδο δεν ήταν πολλά πράγματα γνωστά για την Android A.E., αλλά πολλοί υπέθεσαν ότι η Google σχεδίαζε να μπει στην αγορά των κινητών τηλεφώνων με αυτή την κίνηση. Στην Google, η ομάδα με αρχηγό τον Rubin, ανέπτυξε μια πλατφόρμα για κινητά τηλέφωνα βασισμένο στο πυρήνα του Linux. Η Google προώθησε την πλατφόρμα σε κατασκευαστές συσκευών και πάροχους τηλεπικοινωνιών με την υπόσχεση να παρέχει ένα ευέλικτο και αναβαθμίσιμο σύστημα. Η Google παρέθεσε μια σειρά από hardware και software κατασκευαστές ειδοποιώντας τους πάροχους τηλεπικοινωνιών ότι ήταν ανοιχτή για διάφορα επίπεδα συνεργασίας μαζί τους.

Open Handset Alliance

Στις 6 Νοεμβρίου 2007, η Open Handset Alliance (OHA), μια ομάδα από εταιρίες τεχνολογίας συμπεριλαμβανομένου της Google, κατασκευαστές συσκευών όπως η HTC, η Sony και η Samsung, πάροχοι τηλεπικοινωνιών όπως η Sprint και η T-Mobile, και κατασκευαστές chipset όπως η Qualcomm και η Texas Instruments, ιδρύθηκε με σκοπό την ανάπτυξη ανοιχτών προτύπων για φορητές συσκευές. Εκείνη την μέρα, η το Android παρουσιάστηκε σαν το πρώτο προϊόν της. Μια πλατφόρμα για φορητές συσκευές βασισμένη στον Linux πυρήνα 2.6.

Το πρώτο εμπορικά διαθέσιμο κινητό τηλέφωνο με Android ήταν το HTC Dream (γνωστό και ως T-MOBILE G1) το οποίο κυκλοφόρησε στις 22 Οκτωβρίου 2008.

Εικόνα 1 HTC Dream

Nexus project

Το 2010, η Google λάνσαρε τις δικές της συσκευές, με κωδική ονομασία Nexus. Μια σειρά από smartphones και tablets που λειτουργούν με λογισμικό Android, και έχουν κατασκευαστεί από κάποιον από τους συνεργαζόμενους κατασκευαστές συσκευών. Η HTC συνεργάστηκε με την Google για την κυκλοφορία του πρώτου Nexus smartphone, το Nexus One. Η σειρά Nexus, από τότε, έχει αναβαθμιστεί με νεότερες συσκευές όπως το Nexus 4 κινητό και το Nexus 10 tablet, κατασκευασμένα από την LG και την Samsung αντίστοιχα. Η Google κυκλοφορεί τα Nexus κινητά και tablets, σαν τις ναυαρχίδες Android συσκευές της, παρουσιάζοντας τα νεώτερα hardware και software χαρακτηρίστηκα του Android.

Εικόνα 2 Nexus One

Ιστορικό εκδόσεων

Το Android έχει υποστεί μια σειρά ενημερώσεων μετά από την αρχική κυκλοφορία του. Η καθεμία από αυτές διορθώνει σφάλματα και προσθέτει νέα χαρακτηριστικά. Σε κάθε έκδοση δίνεται ονομασία με αλφαβητική σειρά.

- **Android 1.0 (API level 1)**

Version	Release date	Features
1.0	**23/9/2008**	Android Market application download and updates through the Market app
		Web browser to show, zoom and pan full HTML and XHTML web pages – multiple pages show as windows ("cards")
		Camera support – however, this version lacked the option to change the camera's resolution, white balance, quality, etc.
		Folders allowing the grouping of a number of app icons into a single folder icon on the Home screen
		Access to web email servers, supporting POP3, IMAP4, and SMTP
		Gmail synchronization with the Gmail app
		Google Contacts synchronization with the People app
		Google Calendar synchronization with the Calendar app
		Google Maps with Latitude and Street View to view maps and satellite imagery, as well as find local business and obtain driving directions using GPS
		Google Sync, allowing management of over-the-air synchronization of Gmail, People, and Calendar
		Google Search, allowing users to search the Internet and phone apps, contacts, calendar, etc.
		Google Talk instant messaging
		Instant messaging, text messaging, and MMS
		Media Player, enabling management, importing, and playback of media files – however, this version lacked video and stereo Bluetooth support
		Notifications appear in the Status bar, with options to set ringtone, LED or vibration alerts
		Voice Dialer allows dialing and placing of phone calls without typing a name or number
		Wallpaper allows the user to set the background image or photo behind the Home screen icons and widgets
		YouTube video player
		Other apps include: Alarm Clock, Calculator, Dialer (Phone), Home screen (Launcher), Pictures (Gallery), and Settings
		Wi-Fi and Bluetooth support

Πίνακας 1 Android 1.0

- **Android 1.1 (API level 2)**

Version	Release date	Features
1.0	**9/2/2009**	Details and reviews available when a user searches for businesses on Maps
		Longer in-call screen timeout default when using the speakerphone, plus ability to show/hide dialpad
		Ability to save attachments in messages
		Support added for marquee in system layouts

Πίνακας 2 Android 1.1

- **Android 1.5 Cupcake (API level 3)**

Version	Release date	Features
1.5	30/3/2009	Support for third-party virtual keyboards with text prediction and user dictionary for custom words
		Support for Widgets – miniature application views that can be embedded in other applications (such as the Home screen) and receive periodic updates
		Video recording and playback in MPEG-4 and 3GP formats
		Auto-pairing and stereo support for Bluetooth (A2DP and AVRCP profiles)
		Copy and paste features in web browser
		User pictures shown for Favorites in Contacts
		Specific date/time stamp shown for events in call log, and one-touch access to a contact card from call log event
		Animated screen transitions
		Auto-rotation option
		New stock boot animation
		Ability to upload videos to YouTube
		Ability to upload photos to Picasa

Πίνακας 3 Android 1.5

- **Android 1.6 Donut (API level 4)**

Version	Release date	Features
1.6	15/9/2009	Voice and text entry search enhanced to include bookmark history, contacts, and the web
		Ability for developers to include their content in search results
		Multi-lingual speech synthesis engine to allow any Android application to "speak" a string of text
		Easier searching and ability to view app screenshots in Android Market
		Gallery, camera and camcorder more fully integrated, with faster camera access
		Ability for users to select multiple photos for deletion
		Updated technology support for CDMA/EVDO, 802.1x, VPNs, and a text-to-speech engine
		Support for WVGA screen resolutions
		Speed improvements in searching and camera applications
		Expanded Gesture framework and new GestureBuilder development tool

Πίνακας 4 Android 1.6

- **Android 2.0 Eclair (API level 5)**

Version	Release date	Features
2.0	26/10/2009	Expanded Account sync, allowing users to add multiple accounts to a device for synchronization of email and contacts
		Microsoft Exchange email support, with combined inbox to browse email from multiple accounts in one page
		Bluetooth 2.1 support
		Ability to tap a Contacts photo and select to call, SMS, or email the person
		Ability to search all saved SMS and MMS messages, with delete oldest messages in a conversation automatically deleted when a defined limit is reached
		Numerous new camera features, including flash support, digital zoom, scene mode, white balance, color effect and macro focus
		Improved typing speed on virtual keyboard, with smarter dictionary that learns from word usage and includes contact names as suggestions
		Refreshed browser UI with bookmark thumbnails, double-tap zoom and support for HTML5
		Calendar agenda view enhanced, showing attending status for each invitee, and ability to invite new guests to events
		Optimized hardware speed and revamped UI
		Support for more screen sizes and resolutions, with better contrast ratio
		Improved Google Maps 3.1.2
		MotionEvent class enhanced to track multi-touch events
		Addition of live wallpapers, allowing the animation of home-screen background images to show movement

Πίνακας 5 Android 2.0

- **Android 2.0.1 Eclair (API level 6)**

Version	Release date	Features
2.0.1	3/12/2009	Minor API changes, bug fixes and framework behavioral changes

Πίνακας 6 Android 2.0.1

- **Android 2.1 Eclair (API level 7)**

Version	Release date	Features
2.0.2	12/1/2010	Minor amendments to the API and bug fixes

Πίνακας 7 Android 2.1

- **Android 2.2–2.2.3 Froyo (API level 8)**

Version	Release date	Features
2.2	20/5/2010	Speed, memory, and performance optimizations
		Additional application speed improvements, implemented through JIT compilation
		Integration of Chrome's V8 JavaScript engine into the Browser application
		Support for the Android Cloud to Device Messaging (C2DM) service, enabling push notifications
		Improved Microsoft Exchange support, including security policies, auto-discovery, GAL look-up, calendar synchronization and remote wipe
		Improved application launcher with shortcuts to Phone and Browser applications
		USB tethering and Wi-Fi hotspot functionality
		Option to disable data access over mobile network
		Updated Market application with batch and automatic update features
		Quick switching between multiple keyboard languages and their dictionaries
		Support for Bluetooth-enabled car and desk docks
		Support for numeric and alphanumeric passwords
		Support for file upload fields in the Browser application
		The browser now shows all frames of animated GIFs instead of just the first frame only
		Support for installing applications to the expandable memory
		Adobe Flash support
		Support for high-PPI displays (up to 320 ppi), such as 4" 720p screens
		Gallery allows users to view picture stacks using a zoom gesture
2.2.1	18/1/2011	Bug fixes, security updates and performance improvements
2.2.1	22/1/2011	Minor bug fixes, including SMS routing issues that affected the Nexus One
2.2.3	21/11/2011	Two security patches

Πίνακας 8 Android 2.2-2.2.3

- **Android 2.3–2.3.2 Gingerbread (API level 9)**

Version	Release date	Features
2.3	6/10/2010	Updated user interface design with increased simplicity and speed
		Support for extra-large screen sizes and resolutions (WXGA and higher)[46]
		Native support for SIP VoIP internet telephony
		Faster, more intuitive text input in virtual keyboard, with improved accuracy, better suggested text and voice input mode
		Enhanced copy/paste functionality, allowing users to select a word by press-hold, copy, and paste
		Support for Near Field Communication (NFC), allowing the user to read an NFC tag embedded in a poster, sticker, or advertisement
		New audio effects such as reverb, equalization, headphone virtualization, and bass boost
		New Download Manager, giving users easy access to any file downloaded from the browser, email, or another application
		Support for multiple cameras on the device, including a front-facing camera, if available
		Support for WebM/VP8 video playback, and AAC audio encoding
		Improved power management with a more active role in managing apps that are keeping the device awake for too long
		Enhanced support for native code development
		Switched from YAFFS to ext4 on newer devices[51][52]
		Audio, graphical, and input enhancements for game developers
		Concurrent garbage collection for increased performance
		Native support for more sensors (such as gyroscopes and barometers)
2.3.1	12/2010	Improvements and bug fixes for the Google Nexus S
2.3.2	1/2011	

Πίνακας 9 Android 2.3-2.3.2

- **Android 2.3.3–2.3.7 Gingerbread (API level 10)**

Version	Release date	Features
2.3.3	9/2/2011	Several improvements and API fixes
2.3.4	28/3/2011	Support for voice or video chat using Google Talk
		Open Accessory Library support. Open Accessory was introduced in 3.1 (Honeycomb) but the Open Accessory Library grants 2.3.4 added support when connecting to a USB peripheral with compatible software and a compatible application on the device
		Switched the default encryption for SSL from AES256-SHA to RC4-MD5.
2.3.5	25/7/2011	Improved network performance for the Nexus S 4G, among other fixes and improvements
		Fixed Bluetooth bug on Samsung Galaxy S
		Improved Gmail application
		Shadow animations for list scrolling
		Camera software enhancements
		Improved battery efficiency
2.3.6	2/9/2011	Fixed a voice search bug
		(The 2.3.6 update had the side-effect of impairing the Wi-Fi hotspot functionality of many Canadian Nexus S phones. Google acknowledged this problem and fixed it in late September.)
2.3.7	21/9/2011	Google Wallet support for the Nexus S 4G

Πίνακας 10 Android 2.3.3-2.3.7

- **Android 3.0 Honeycomb (API level 11)**

Version	Release date	Features
3.0	22/2/2011	Optimized tablet support with a new virtual and “holographic” user interface
		Added System Bar, featuring quick access to notifications, status, and soft navigation buttons, available at the bottom of the screen
		Added Action Bar, giving access to contextual options, navigation, widgets, or other types of content at the top of the screen
		Simplified multitasking – tapping Recent Apps in the System Bar allows users to see snapshots of the tasks underway and quickly jump from one app to another
		Redesigned keyboard, making typing fast, efficient and accurate on larger screen sizes
		Simplified, more intuitive copy/paste interface
		Multiple browser tabs replacing browser windows, plus form auto-fill and a new “incognito” mode allowing anonymous browsing
		Quick access to camera exposure, focus, flash, zoom, front-facing camera, time-lapse, and other camera features
		Ability to view albums and other collections in full-screen mode in Gallery, with easy access to thumbnails for other photos
		New two-pane Contacts UI and Fast Scroll to let users easily organize and locate contacts
		New two-pane Email UI to make viewing and organizing messages more efficient, allowing users to select one or more messages
		Support for video chat using Google Talk
		Hardware acceleration
		Support for multi-core processors
		Ability to encrypt all user data
		HTTPS stack improved with Server Name Indication (SNI)
		Filesystem in Userspace (FUSE; kernel module)

Πίνακας 11 Android 3.0

- **Android 3.1 Honeycomb (API level 12)**

Version	Release date	Features
3.1	10/5/2010	
		UI refinements
		Connectivity for USB accessories
		Expanded Recent Apps list
		Resizable Home screen widgets
		Support for external keyboards and pointing devices
		Support for joysticks and gamepads
		Support for FLAC audio playback[68][69]
		High-performance Wi-Fi lock, maintaining high-performance Wi-Fi connections when device screen is off
		Support for HTTP proxy for each connected Wi-Fi access point

Πίνακας 12 Android 3.1

- **Android 3.2 Honeycomb (API level 13)**

Version	Release date	Features
3.2	**15/7/2011**	Improved hardware support, including optimizations for a wider range of tablets
		Increased ability of apps to access files on the SD card, e.g. for synchronization
		Compatibility display mode for apps that have not been optimized for tablet screen resolutions
		New display support functions, giving developers more control over display appearance on different Android devices
3.2.1	**30/8/2011**	Bug fixes and minor security, stability and Wi-Fi improvements
		Update to Android Market with automatic updates and easier-to-read Terms and Conditions text
		Update to Google Books
		Improved Adobe Flash support in browser
		Improved Chinese handwriting prediction
3.2.2	**20/11/2011**	Bug fixes and other minor improvements for the Motorola Xoom 4G
3.2.3		Bug fixes and other minor improvements for the Motorola Xoom and Motorola Xoom 4G
3.2.4	**12/2011**	"Pay as You Go" support for 3G and 4G tablets
3.2.5	**1/2012**	Bug fixes and other minor improvements for the Motorola Xoom and Motorola Xoom 4G
3.2.6	**2/2012**	Fixed data connectivity issues when coming out of airplane mode on the US 4G Motorola Xoom

Πίνακας 13 Android 3.2

- **Android 4.0–4.0.2 Ice Cream Sandwich (API level 14)**

Version	Release date	Features
4.0	19/10/2011	Soft buttons from Android 3.x are now available for use on phones
		Separation of widgets in a new tab, listed in a similar manner to apps
		Easier-to-create folders, with a drag-and-drop style
		A customizable launcher
		Improved visual voicemail with the ability to speed up or slow down voicemail messages
		Pinch-to-zoom functionality in the calendar
		Integrated screenshot capture (accomplished by holding down the Power and Volume-Down buttons)
		Improved error correction on the keyboard
		Ability to access apps directly from lock screen
		Improved copy and paste functionality
		Better voice integration and continuous, real-time speech to text dictation
		Face Unlock, a feature that allows users to unlock handsets using facial recognition software
		New tabbed web browser under Google's Chrome brand, allowing up to 16 tabs
		Automatic syncing of browser with users' Chrome bookmarks
		A new typeface family for the UI, Roboto
		Data Usage section in settings that lets users set warnings when they approach a certain usage limit, and disable data use when the limit is exceeded
		Ability to shut down apps that are using data in the background
		Improved camera app with zero shutter lag, time lapse settings, panorama mode, and the ability to zoom while recording
		Built-in photo editor
		New gallery layout, organized by location and person
		Refreshed "People" app with social network integration, status updates and hi-res images
		Android Beam, a near-field communication feature allowing the rapid short-range exchange of web bookmarks, contact info, directions, YouTube videos and other data
		Support for the WebP image format
		Hardware acceleration of the UI
		Wi-Fi Direct
		1080p video recording for stock Android devices
		Android VPN Framework (AVF), and TUN (but not TAP) kernel module. Prior to 4.0, VPN software required rooted Android.
4.0.1	21/10/2011	Fixed minor bugs for the Samsung Galaxy Nexus.
4.0.2	28/11/2011	Fixed minor bugs on the Verizon Galaxy Nexus, the US launch of which was later delayed until December 2011
		(For Canadian consumers, 4.0.2 reportedly created a bug on the Galaxy Nexus that crashed the application market when users attempted to view details of any Android application. It also inadvertently reduced theNFC capabilities of the Nexus phone).

Πίνακας 14 Android 4.0-4.0.2

- **Android 4.0.3–4.0.4 Ice Cream Sandwich (API level 15)**

Version	Release date	Features
4.0.3	16/12/2011	Numerous bug fixes and optimizations
		Improvements to graphics, databases, spell-checking and Bluetooth functionality
		New APIs for developers, including a social stream API in the Contacts provider
		Calendar provider enhancements
		New camera apps enhancing video stabilization and QVGA resolution
		Accessibility refinements such as improved content access for screen readers
4.0.4	29/3/2012	Stability improvements
		Better camera performance
		Smoother screen rotation
		Improved phone number recognition

Πίνακας 15 Android 4.0.3-4.0.4

- **Android 4.1 Jelly Bean (API level 16)**

Version	Release date	Features
4.1	9/7/2012	Smoother user interface:
		Vsync timing across all drawing and animation done by the Android framework, including application rendering, touch events, screen composition and display refresh
		Triple buffering in the graphics pipeline
		Enhanced accessibility
		Bi-directional text and other language support
		User-installable keyboard maps
		Expandable notifications
		Ability to turn off notifications on an app specific basis
		Shortcuts and widgets can automatically be re-arranged or re-sized to allow new items to fit on home screens
		Bluetooth data transfer for Android Beam
		Offline voice dictation
		Tablets with smaller screens now use an expanded version of the interface layout and home screen used by phones.
		Improved voice search
		Improved camera app
		Google Wallet (for the Nexus 7)
		High-resolution Google+ contact photos
		Google Now voice assistant and search application
		Multichannel audio[89]
		USB audio (for external sound DACs)
		Audio chaining (also known as gapless playback)
		Stock Android browser is replaced with the Android mobile version of Google Chrome in devices with Android 4.1 preinstalled
		Ability for other launchers to add widgets from the app drawer without requiring root access
4.1.1	23/7/2012	Fixed a bug on the Nexus 7 regarding the inability to change screen orientation in any application
4.1.2	9/10/2012	Lock/home screen rotation support for the Nexus 7
		One-finger gestures to expand/collapse notifications
		Bug fixes and performance enhancements

Πίνακας 16 Android 4.1

- **Android 4.2 Jelly Bean (API level 17)**

Version	Release date	Features
4.2	13/11/2012	"Photo Sphere" panorama photos
		Keyboard with gesture typing (this feature is also available for Android 4.0 and later via the Google Keyboard app)
		Lock screen improvements, including widget support and the ability to swipe directly to camera
		Notification power controls ("Quick Settings")
		"Daydream" screensavers, showing information when idle or docked
		Multiple user accounts (tablets only)
		Support for wireless display (Miracast)
		Accessibility improvements: triple-tap to magnify the entire screen, pan and zoom with two fingers. Speech output and Gesture Mode navigation for blind users
		New clock app with built-in world clock, stop watch and timer
		All devices now use the same interface layout, previously adapted from phones on 4.1 for smaller tablets (with centered software buttons, the system bar at the top of the screen, and a home screen with a dock and centered application menu), regardless of screen size
		Increased number of extended notifications and Actionable Notifications for more apps, allowing users to respond to certain notifications within the notification bar and without launching the app directly
		SELinux
		Always-on VPN
		Premium SMS confirmation
		Group Messaging
4.2.1	27/11/2012	Fixed a bug in the People app where December was not displayed on the date selector when adding an event to a contact
		Added Bluetooth gamepads and joysticks as supported HID (Human interface device)
4.2.2	11/2/2013	Fixed Bluetooth audio streaming bugs
		Long-pressing the Wi-Fi and Bluetooth icons in Quick Settings now toggles the on/off state
		New download notifications, which now shows the percentage and estimated time remaining for active app downloads
		New sounds for wireless charging and low battery
		New Gallery app animation allows faster loading
		USB debug whitelist
		Bug fixes and performance enhancements

Πίνακας 17 Android 4.2

- **Android 4.3 Jelly Bean (API level 18)**

Version	Release date	Features
4.3	24/7/2013	Bluetooth low energy support.
		Bluetooth Audio/Video Remote Control Profile (AVRCP) 1.3 support
		OpenGL ES 3.0 support, allowing for improved game graphics
		Restricted access mode for new user profiles
		Filesystem write performance improvement by running fstrim command while device is idle
		Dial pad auto-complete in the Phone app
		Improvements to Photo Sphere
		Reworked camera UI, previously introduced on Google Play edition phones
		Added fine-grained application permissions controls (hidden by default)
		4K resolution support
		Many security enhancements, performance enhancements, and bug fixes
		System-level support for geofencing and Wi-Fi scanning APIs
		Background Wi-Fi location still runs even when Wi-Fi is turned off
		Developer logging and analyzing enhancements
		Added support for five more languages
		Improved digital rights management (DRM) APIs
		Right-to-left (RTL) languages now supported
		Clock in the status bar disappears if clock is selected as lockscreen widget
4.3.1	3/10/2013	Bug fixes and small tweaks for the Nexus 7 LTE

Πίνακας 18 Android 4.3

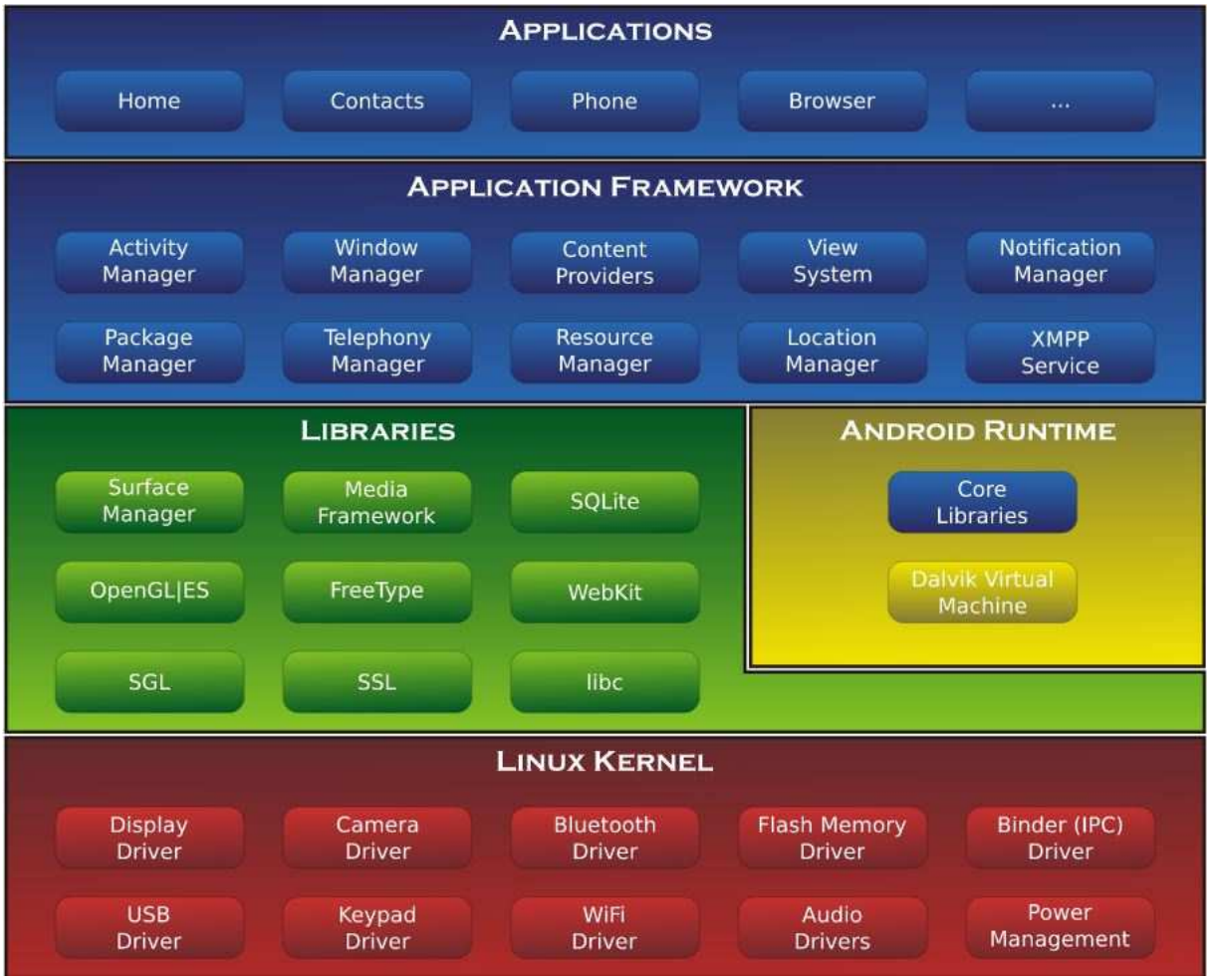
- **Android 4.4 KitKat (API level 19)**

Version	Release date	Features
4.4	31/10/2013	Refreshed interface with translucent status and navigation bars on home screen and white status bar icons/text, ability for apps to trigger the translucent appearance.
		Optimizations for performance on devices with lower specifications, Low RAM device API
		Printing framework
		NFC Host Card Emulation for emulating smart cards
		WebViews based on Chromium (feature parity with Chrome for Android 30)
		Expanded functionality for Notification listener services
		Public API for developing and managing text messaging clients, ability to specify a default SMS app
		New framework for UI transitions
		Storage access framework for retrieving content and documents from other sources
		Sensor batching, Step Detector and Counter APIs
		"Immersive" full screen mode, software buttons and status bar accessed with an edge swipe gesture
		Audio tunneling, audio monitoring, loudness enhancer
		Built-in screen recording
		Native infrared blaster API
		Expanded accessibility APIs, system-level closed captioning settings
		New experimental runtime virtual machine, ART
		Bluetooth Message Access Profile (MAP) support

Πίνακας 19 Android 4.4

Αρχιτεκτονική Android

Η πλατφόρμα Android, είναι μια στοίβα λογισμικού που αποτελείται από ένα πυρήνα που βασίζεται στον πυρήνα του Linux, με middleware, βιβλιοθήκες και APIs γραμμένα σε γλώσσα C και λογισμικό εφαρμογών που λειτουργούν σε ένα πλαίσιο εφαρμογών που περιλαμβάνει συμβατές βιβλιοθήκες Java, βασισμένες στο project Apache Harmony. Το Android χρησιμοποιεί την εικονική μηχανή Dalvik με δυνατότητα "just-in-time compilation" για να μπορεί να τρέχει Dalvik DEX-κώδικα (Εκτελέσιμο από Dalvik), που συνήθως μεταφράζεται από Java byte κώδικα. Η βασική πλατφόρμα hardware για το Android είναι η αρχιτεκτονική ARM. Υπάρχει υποστήριξη για x86 (32 bit) από το project Android x86, και η Google TV χρησιμοποιεί μια ειδική x86 έκδοση του Android.



Εικόνα 3 Αρχιτεκτονική του Android

Android Linux Kernel

Το Android είναι βασισμένο στα γερά θεμέλια του Linux. Ο πυρήνας Linux είναι δοκιμασμένος, σταθερός και πετυχημένος και μπορεί να βρεθεί παντού, από ρολόγια χειρός μέχρι υπερυπολογιστές. Το Linux παρέχει στο Android το αφαιρετικό επίπεδο υλικού, επιτρέποντάς του να μπορεί να χρησιμοποιηθεί σε μεγάλη ποικιλία πλατφόρμων στο μέλλον. Ειδικότερα, το Android χρησιμοποιεί τον πυρήνα Linux για την διαχείριση μνήμης, την διαχείριση διεργασιών, την δικτύωση και άλλες υπηρεσίες του λειτουργικού συστήματος.

Native Libraries

Στο αμέσως ψηλότερο επίπεδο υπάρχουν οι Native Libraries – Εγγενής Βιβλιοθήκες. Αυτές οι βιβλιοθήκες είναι γραμμένες στην γλώσσα προγραμματισμού C και C++ και μεταγλωττίστηκαν για την συγκεκριμένη αρχιτεκτονική υλικού που χρησιμοποιείται από το τηλέφωνο. Οι βιβλιοθήκες αυτές δεν είναι εφαρμογές που μπορούν να σταθούν από μόνες τους. Υπάρχουν για να μπορούν να κληθούν από προγράμματα υψηλότερου επιπέδου. Από την έκδοση Donut και μετά, οι κατασκευαστές μπορούν να γράφουν τις δικές τους τέτοιες βιβλιοθήκες με την χρήση της Εργαλειοθήκης NDK (Native Development Kit).

Android Runtime

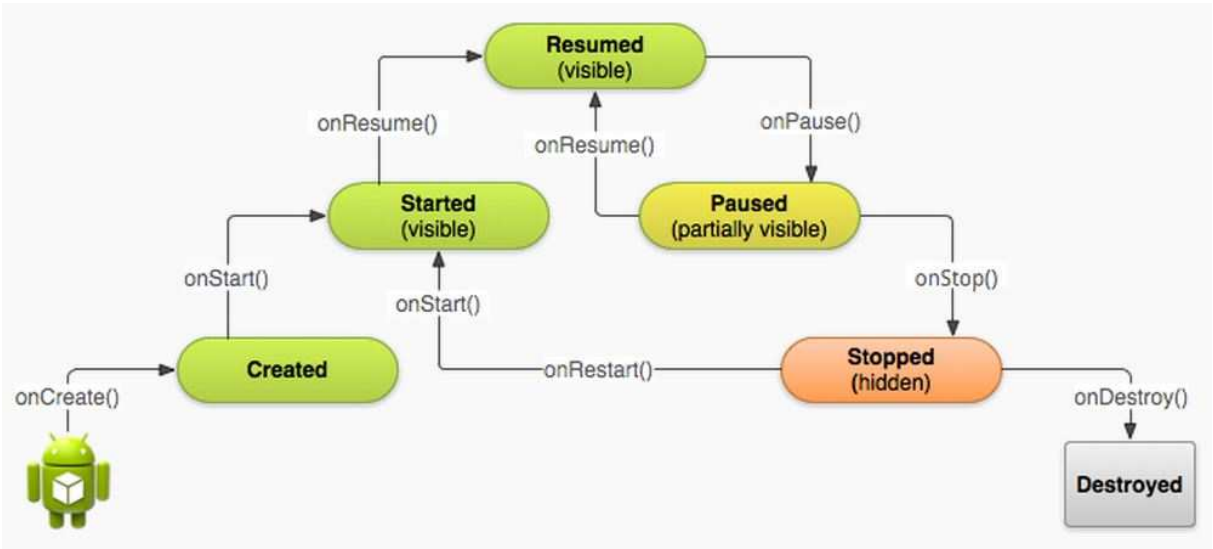
Στο ίδιο επίπεδο με τις εγγενής βιβλιοθήκες, βρίσκεται και το Android Runtime. Εδώ βρίσκονται βασικές βιβλιοθήκες της Java και η εικονική μηχανή Dalvik. Η Dalvik είναι μια βελτιστοποιημένη υλοποίηση μιας εικονικής μηχανής Java για φορητές συσκευές από την Google. Η Dalvik τρέχει .dex αρχεία, τα οποία είναι bytecodes που προέρχονται από αρχεία .class και .jar. Εν αντιθέσει όμως με τα .class αρχεία, τα .dex είναι πολύ πιο συμπαγή και αποδοτικά, γεγονός σημαντικό για συσκευές με περιορισμένη μνήμη και μπαταρία. Το Android περιλαμβάνει ένα σύνολο βασικών βιβλιοθηκών που παρέχουν τις περισσότερες από τις διαθέσιμες λειτουργίες των βασικών βιβλιοθηκών της Java. Κάποια πακέτα και κλάσεις υπάρχουν και στο Android κάποια άλλα δεν υποστηρίζονται καθόλου, ενώ ταυτόχρονα το Android παρέχει και επιπρόσθετα, προσαρμοσμένα στις δικές του ανάγκες.

Application Framework

Πάνω από τις εγγενής βιβλιοθήκες και το χρόνο εκτέλεσης Android, είναι το πλαίσιο εφαρμογής. Αυτό το επίπεδο μας παρέχει υψηλού επιπέδου δομικές μονάδες τις οποίες μπορούμε να χρησιμοποιούμε για την κατασκευή των εφαρμογών μας. Αυτό το πλαίσιο είναι προ-εγκατεστημένο στο Android, αλλά είναι επεκτάσιμο, αφού ο κάθε κατασκευαστής μπορεί να το συμπληρώσει με δικά του κομμάτια.

Τα σημαντικότερα δομικά στοιχεία του πλαισίου αυτού είναι:

- **Activity Manager:** Υπεύθυνο για τον έλεγχο του χρόνου ζωής (Εικόνα 4) των εφαρμογών και για την διατήρηση μιας στοίβας που επιτρέπει την πλοήγηση του χρήστη σε προηγούμενες οθόνες.
- **Content Providers:** Αυτά τα αντικείμενα περιέχουν δεδομένα που μπορούν να διαμοιραστούν μεταξύ εφαρμογών.
- **Resource Manager:** Οι πόροι, είναι οτιδήποτε υπάρχει σε ένα πρόγραμμα και δεν είναι κώδικας. Για παράδειγμα μπορεί να είναι κωδικοί χρωμάτων, αλφαριθμητικοί χαρακτήρες ή ακόμα και έτοιμα σχεδιαγράμματα οθονών φτιαγμένα σε XML, τα οποία μπορεί το πρόγραμμα να καλεί.
- **Location Manager:** Χρησιμοποιείται για να μπορεί να ξέρει το τηλέφωνο που βρίσκεται ανά πάσα στιγμή.
- **Notification Manager:** Ιδανικός τρόπος για να ενημερώνεις τον χρήστη για γεγονότα που συμβαίνουν, διακριτικά χωρίς να διακόπτεις την εργασία του.



Εικόνα 4 Activity Lifecycle

Applications and Widgets

Στο υψηλότερο επίπεδο της στοίβας Android, βρίσκονται οι εφαρμογές και τα widgets. Αυτό είναι που βλέπουν οι χρήστες χωρίς να γνωρίζουν την διαδικασία που κρύβεται από κάτω. Αυτές είναι εφαρμογές που γράφουν οι κατασκευαστές λογισμικού, εφαρμογές που ήδη είναι εγκατεστημένες στο τηλέφωνο ή που ο χρήστης παίρνει από το Google Play. Οι εφαρμογές είναι προγράμματα που καταλαμβάνουν ολόκληρη την οθόνη και αλληλεπιδρούν με το χρήστη. Από την άλλη τα widget λειτουργούν σε μικρά τετράγωνα μέσα στην αρχική οθόνη .

Κύρια χαρακτηριστικά του Android

Interface

Το Interface του Android είναι βασισμένο στον άμεσο χειρισμό, χρησιμοποιώντας σαν είσοδο ενέργειες που ελαφρώς αντιστοιχούν σε πραγματικές κινήσεις, όπως πέρασμα του χεριού κατά μήκος της οθόνης, άγγιγμα κ.α. για τον χειρισμό τον αντικειμένων πάνω στην οθόνη. Η απάντηση στην ενέργεια του χρήστη έχει σχεδιαστεί ώστε να είναι άμεση και να παρέχει μια συνεχή διεπαφή αφής, χρησιμοποιώντας αρκετά συχνά τις δυνατότητες δόνησης της συσκευής για να προσφέρει στον χρήστη απτική ανάδραση. Εσωτερικό hardware της συσκευής όπως επιταχυνσιόμετρα (accelerometers), γυροσκόπια (gyroscopes) και αισθητήρες εγγύτητας (proximity sensors) χρησιμοποιούνται από κάποιες εφαρμογές για να ανταποκριθούν σε επιπλέον ενέργειες του χρήστη, για παράδειγμα ρυθμίζοντας την οθόνη από κάθετη σε οριζόντια προβολή αναλόγως με το πώς είναι στραμμένη η συσκευή.

Οι συσκευές Android ανοίγουν στην αρχική οθόνη, το κεντρικό σημείο πλοήγησης και λήψης πληροφοριών στη συσκευή, που είναι παρόμοια με την επιφάνεια εργασία που βρίσκεται στα PCs. Οι αρχικές οθόνες των Android αποτελούνται, συνήθως, από εικονίδια εφαρμογών και widgets. Τα εικονίδια ξεκινούν την ανάλογη εφαρμογή, ενώ τα widgets δείχνουν live, ανανεώσιμο περιεχόμενο όπως για παράδειγμα την πρόβλεψη του καιρού, τα emails του χρήστη κ.α. Μια αρχική οθόνη μπορεί να αποτελείται από πολλές σελίδες στις οποίες ο χρήστης μπορεί να πλοηγηθεί. Το interface του Android είναι αρκετά παραμετροποιήσιμο, επιτρέποντας στον χρήστη να προσαρμόσει την εμφάνιση της συσκευής του στο δικό του γούστο.

Οι περισσότεροι κατασκευαστές, καθώς και κάποιοι πάροχοι τηλεπικοινωνιών, προσαρμόζουν την εμφάνιση των Android συσκευών τους ώστε να διαφοροποιήσουν τον εαυτό τους από τους ανταγωνιστές τους.

Στην κορυφή της οθόνης βρίσκεται η μπάρα κατάστασης (status bar), που δείχνει πληροφορίες σχετικά με τη συσκευή και τη συνδεσιμότητα της. Αυτή η μπάρα μπορεί να «τραβηχτεί» κάτω και να εμφανίσει μια οθόνη ειδοποιήσεων όπου οι εφαρμογές παρουσιάζουν σημαντικές πληροφορίες ή ενημερώσεις, όπως ένα νέο email ή SMS, με τέτοιο τρόπο που δεν διακόπτει ή ενοχλεί άμεσα τον χρήστη.

Εφαρμογές

Το Android έχει μια συνεχώς αναπτυσσόμενη συλλογή από εφαρμογές τρίτων, που ο χρήστη μπορεί να αποκτήσει είτε από ένα App Store όπως το Google Play, είτε κατεβάζοντας και εγκαθιστώντας το αρχείο APK της εφαρμογής από κάποιο site. Η Play Store εφαρμογή επιτρέπει στους χρήστες να αναζητήσουν, κατεβάσουν και να ενημερώσουν εφαρμογές που έχουν εκδοθεί από την Google και άλλους κατασκευαστές λογισμικού (developer), και είναι προεγκατεστημένη σε συσκευές που συμμορφώνονται με της απαιτήσεις συμβατότητας της Google. Η εφαρμογή φιλτράρει την λίστα των διαθέσιμων εφαρμογών σε αυτές που είναι διαθέσιμες για την συσκευή του χρήστη. Επίσης κάποιοι developers μπορεί να επιλέξουν να περιορίσουν την εφαρμογή τους μόνο σε κάποιες συγκεκριμένες χώρες ή πάροχους τηλεπικοινωνιών, για επιχειρησιακούς λόγους. Μέχρι τον Σεπτέμβριου του 2012, υπήρχαν πάνω από 675000 εφαρμογές διαθέσιμες για Android, και ο υπολογιζόμενος αριθμός των εφαρμογών που κατεβάστηκαν από το Play Store ήταν 25 δισεκατομμύρια.

Οι εφαρμογές αναπτύσσονται σε γλώσσα προγραμματισμού Java χρησιμοποιώντας το Android software development kit (SDK). Αυτό το SDK ένα σετ από εργαλεία ανάπτυξης, συμπεριλαμβανομένου ενός debugger, βιβλιοθήκες λογισμικού, έναν εξομοιωτή κινητού βασισμένο στο QEMU, documentation, παραδείγματα κώδικα και tutorials. Το επισήμως υποστηριζόμενο ενσωματωμένο περιβάλλον ανάπτυξης (IDE) είναι το Eclipse χρησιμοποιώντας το Andoid Development Tools (ADT) πρόσθετο.

Διαχείριση Μνήμης

Επειδή οι συσκευές Android τροφοδοτούνται συνήθως από μπαταρία, το Android έχει σχεδιαστεί να διαχειρίζεται την μνήμη της συσκευής (RAM) με κατάλληλο τρόπο ώστε να διατηρεί την κατανάλωση ενέργειας στο ελάχιστο δυνατό. Όταν μια εφαρμογή δεν χρησιμοποιείται πλέον, το σύστημα αυτόματα αναστέλλει αυτή την εφαρμογή στη μνήμη. Ενώ η εφαρμογή είναι τεχνικά «ανοιχτή», εφαρμογές που έχουν ανασταλεί δεν καταναλώνουν πόρους και μένουν άπραγες στο παρασκήνιο μέχρι να χρειαστούν ξανά.

Απαιτήσεις Hardware

Από τον Νοέμβριο του 2013, η τρέχουσα έκδοση του Android (4.4 KitKat) απαιτεί τουλάχιστον 512 MB RAM, και επεξεργαστή αρχιτεκτονικής 32-bit ARMv7, MIPS ή x86, μαζί με μονάδα επεξεργασίας γραφικών (GPU) συμβατή με OpenGL ES 2.0.

Μερίδιο Αγοράς

Η εταίρα ερευνών Canalys υπολόγισε ότι στο δεύτερο τρίμηνο του 2009, το Android κατείχε το 2,8% των αποστολών smartphones παγκοσμίως. Έως το τέταρτο τρίμηνο του 2010 αυτό το ποσοστό είχε ανέβει στο 33% της αγοράς, κάνοντας το Android κορυφαία σε πωλήσεις πλατφόρμα για smartphone. Μέχρι το τρίτο τρίμηνο του 2011, η Gartner υπολόγισε ότι πάνω από το μισό (52.5%) της αγοράς των smartphones ανήκε στο Android. Μέχρι το τρίτο τρίμηνο του 2012 το Android κατείχε το 75% της παγκόσμιας αγοράς smartphone, σύμφωνα με την έρευνα της IDC.

Τον Ιούλιο του 2011, η Google δήλωσε ότι 550.000 νέες συσκευές Android ενεργοποιούνται κάθε μέρα, αυξημένος αριθμός από τις 400.000 συσκευές ανά ημέρα το Μάιο, και πάνω από 100 εκατομμύρια συσκευές είχαν ενεργοποιηθεί με 4,4% αύξηση ανά εβδομάδα. Τον Σεπτέμβριο του 2012, 500 εκατομμύρια συσκευές είχαν ενεργοποιηθεί με 1,3 εκατομμύρια ενεργοποιήσεις ανά ημέρα. Τον Μάιο του 2013 στο Google I / O, ο Sundar Pichai ανακοίνωσε ότι 900 εκατομμύρια συσκευές Android είχαν ενεργοποιηθεί.

Το μερίδιο της αγοράς του Android ποικίλλει ανά περιοχή. Τον Ιούλιο του 2012, το μερίδιο αγοράς του Android στις Ηνωμένες Πολιτείες ήταν 52%, και ανήλθε σε 90% στην Κίνα. Κατά τη διάρκεια του τρίτου τριμήνου του 2012, το μερίδιο της παγκόσμιας αγοράς smartphone του Android ήταν 75%, με 750 εκατομμύρια ενεργοποιημένες συσκευές στο σύνολο και 1,5 εκατομμύρια ενεργοποιήσεις ανά ημέρα.

Από τον Μάρτιο του 2013 το μερίδιο του Android στην παγκόσμια αγορά smartphone, με επικεφαλής τα προϊόντα της Samsung, ήταν 64%. Η Kantar εταιρεία έρευνας αγοράς ανέφερε ότι η πλατφόρμα της Google αντιπροσώπευε πάνω από το 70% του συνόλου των πωλήσεων smartphone στην Κίνα κατά τη διάρκεια αυτής της περιόδου και ότι το ποσοστό αφοσίωσης της Samsung στη Βρετανία (59%) είναι η δεύτερο μετά από αυτό της Apple (79%).

Μερίδιο Χρήσης

Αυτός ο πίνακας παρέχει στοιχεία σχετικά με τον αριθμό των συσκευών είχαν πρόσβαση στο Play Store ,πρόσφατα και τρέχουν μια συγκεκριμένη έκδοση της πλατφόρμας Android, από της 2 Οκτωβρίου 2013.

Version	Code name	Release date	Distribution
4.4	KitKat	31/10/2013	0%
4.3.x	Jelly Bean	24/7/2013	1.5%
4.2.x	Jelly Bean	13/11/2012	10.6%
4.1.x	Jelly Bean	9/72012	36.5%
4.0.3–4.0.4	Ice Cream Sandwich	16/12/2011	20.6%
3.2	Honeycomb	15/7/2011	0.1%
3.1	Honeycomb	10/5/2011	0%
2.3.3–2.3.7	Gingerbread	9/2/2011	28.5%
2.3–2.3.2	Gingerbread	6/12/2010	0%
2.2	Froyo	20/5/2010	2.2%
2.0–2.1	Eclair	26/10/2009	0%
1.6	Donut	15/9/2009	0%

1.5	**Cupcake**	**30/3/2009**	0%

Πίνακας 20 Μερίδιο Χρήσης

Καταγραφή και Ανάλυση Απαιτήσεων

Εισαγωγή

Στο κεφάλαιο αυτό, θα καταγραφούν και θα αναλυθούν οι απαιτήσεις της εφαρμογής NotifyMe. Το όνομα αυτό δόθηκε στην εφαρμογή επειδή περιγράφει την λειτουργικότητα της εφαρμογή, μιας και τα μηνύματα που ανταλλάσσονται μεταξύ των χρηστών, λαμβάνονται από τον παραλήπτη σαν Notification.

Η διαδικασία της καταγραφής και της ανάλυσης των απαιτήσεων, είναι ιδιαιτέρως σημαντική, καθώς θα προσδιοριστούν ακριβώς οι ανάγκες που πρέπει να καλυφτούν από το NotifyMe.

Ορολογία

Πριν την καταγραφή και ανάλυση των απαιτήσεων για την υλοποίηση αυτής της εργασίας, πρέπει να οριστούν οι οντότητες που θα χρησιμοποιηθούν, για την αποφυγή παρεξηγήσεων και παρερμηνειών.

Δεδομένα (data): Στην επιστήμη της πληροφορικής, τα δεδομένα είναι οτιδήποτε σε μορφή κατάλληλη για χρήση από τον υπολογιστή, αλλά όχι κώδικας.

Βάση Δεδομένων (database): Με τον όρο βάσεις δεδομένων αναφερόμαστε σε οργανωμένες, διακριτές συλλογές σχετιζόμενων δεδομένων, ηλεκτρονικά και ψηφιακά αποθηκευμένων, στο λογισμικό που χειρίζεται τέτοιες συλλογές (Σύστημα Διαχείρισης Βάσεων Δεδομένων, ή DBMS) και στο γνωστικό πεδίο που το μελετά.

Πίνακας Δεδομένων (database table): Είναι ένα σύνολο δεδομένων τα οποία είναι οργανωμένα με ένα μοντέλο στηλών (που προσδιορίζονται από το όνομα τους) και γραμμές. Ένας πίνακας αποτελείται από ένα καθορισμένο αριθμό στηλών, αλλά μπορεί να έχει πολλές σειρές. Κάθε γραμμή αναγνωρίζεται από τις τιμές που υπάρχουν σε μία συγκεκριμένη στήλη ή σύνολο στηλών που έχει οριστεί σαν μοναδικό κλειδί / δείκτης.

Εξυπηρετητής (server): Είναι υλικό ή / και λογισμικό που αναλαμβάνει την παροχή διάφορων υπηρεσιών, «εξυπηρετώντας» αιτήσεις άλλων προγραμμάτων, γνωστούς ως πελάτες (clients) που μπορούν να τρέχουν στον ίδιο υπολογιστή ή σε σύνδεση μέσω δικτύου.

Χρήστης (user): Είναι το άτομο το οποίο πραγματοποιεί τη χρήση της εφαρμογής.

Προσδιορισμός Απαιτήσεων

Οι απαιτήσεις που πρέπει να υλοποιηθούν σε αυτή την εφαρμογή, σκοπό έχουν την μεγαλύτερη ικανοποίηση του χρήστη. Πιο συγκεκριμένα οι ανάγκες που πρέπει να ικανοποιηθούν στην υλοποίηση της εφαρμογής είναι οι εξής:

- Να μπορούν οι χρήστες να πραγματοποιούν εγγραφή στην εφαρμογή, με τον αριθμό τηλεφώνου τους
- Προβολή των επαφών του χρήστη και διαχωρισμός χρηστών της εφαρμογής καθώς και αυτών που είναι συνδεδεμένοι
- Δυνατότητα αποστολής σύντομου μηνύματος σε κάποιον άλλο χρήστη της εφαρμογής
- Δυνατότητα επιλογής αποστολής προεπιλεγμένου μηνύματος με την απλή επιλογή μιας επαφής
- Δυνατότητα λήψης σύντομων μηνυμάτων από άλλους χρήστες της εφαρμογής
- Δυνατότητα αποχώρησης του χρήστη από την εφαρμογή

Ανάλυση Απαιτήσεων

Σε αυτή την ενότητα, γίνεται η ανάλυση των απαιτήσεων που εντοπίστηκαν και καταγράφτηκαν παραπάνω.

Εγγραφή χρήστη: Ο χρήστης, την πρώτη φορά που θα χρησιμοποιήσει την εφαρμογή, θα πρέπει να μπορεί να δημιουργήσει λογαριασμό με την χρήση του αριθμού τηλεφώνου του.

Είσοδος στην εφαρμογή: Μετά την αρχική εγγραφή του, ο χρήστης θα πρέπει να μπορεί να χρησιμοποιεί τη εφαρμογή χωρίς να χρειάζεται να εισάγει κάθε φορά τα στοιχεία του κάθε φορά που θα την ανοίγει.

Προβολή άλλων χρηστών: Μετά την είσοδο στην εφαρμογή θα πρέπει να μπορεί να βλέπει μια λίστα με τις επαφές του, και με εύκολο τρόπο να καταλαβαίνει αν κάποια επαφή είναι χρήστης της εφαρμογής, χωρίς καμία ενέργεια από μέρος του.

Λήψη μηνυμάτων (μέσω διαδικτύου): Ο χρήστης θα μπορεί να δέχεται μηνύματα, από άλλους χρήστες, μέσω διαδικτύου.

Αποστολή μηνυμάτων (μέσω διαδικτύου): Ο χρήστης θα μπορεί να στέλνει μηνύματα, σε άλλους χρήστες, μέσω διαδικτύου.

Αποστολή μηνυμάτων (μέσω GSM): Ο χρήστης θα μπορεί να στέλνει μηνύματα, σε άλλους χρήστες και μη, μέσω του δικτύου GSM σε μορφή SMS.

Τηλεφωνική κλήση: Ο χρήστης θα μπορεί να πραγματοποιεί τηλεφωνικές κλήσεις προς οποιαδήποτε επαφή, είτε είναι χρήστης της εφαρμογής είτε όχι.

Αποστολή προεπιλεγμένου μηνύματος: Ο χρήστης θα μπορεί να ορίσει ένα προεπιλεγμένο μήνυμα το οποίο θα αποστέλλεται σαν το μήνυμα προς την επαφή που θα επιλέξει.

Αυτόματη αποστολή Notification/SMS: Ο χρήστης θα μπορεί να επιλέξει αν θα επιλέγει αυτός τον τρόπο με τον οποίο θα αποστέλλεται κάθε φορά το μήνυμα ή αν θα επιλέγεται αυτόματα από την εφαρμογή, βάσει της επιλεγμένης επαφής (αν είναι χρήστης της εφαρμογής ή όχι, αν είναι συνδεδεμένος εκείνη την στιγμή κλπ).

Αποχώρηση από την εφαρμογή: Ο χρήστης θα έχει την επιλογή να πραγματοποιήσει απεγγραφή από την εφαρμογή.

Υλοποίηση Συστήματος

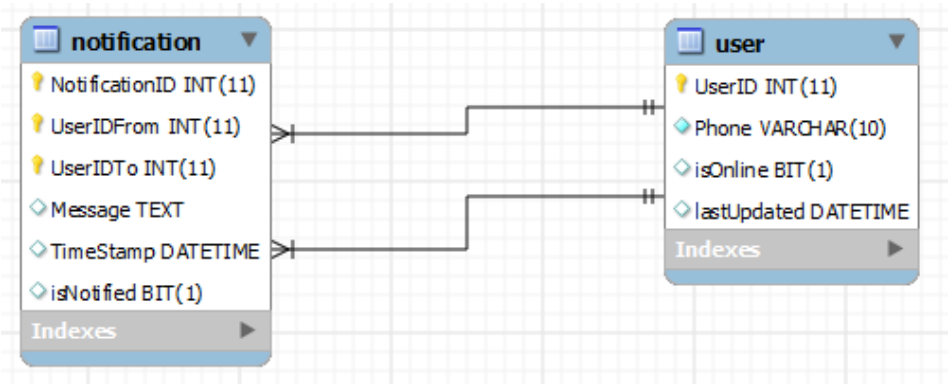
Εισαγωγή

Όπως έχει γίνει σαφές από τις απαιτήσεις για την υλοποίηση αυτής της εργασίας, η υλοποίηση αποτελείται από δυο σκέλη. Την δημιουργία του server, και την δημιουργία του client, που είναι ουσιαστικά και η Android εφαρμογή. Στο κεφάλαιο αυτό θα ασχοληθούμε με την υλοποίηση αυτών των δυο συστημάτων.

Σχήμα βάσης δεδομένων

Σε αυτή την ενότητα παρουσιάζεται το διάγραμμα της βάσης δεδομένων που θα χρησιμοποιηθεί για την εφαρμογή NotifyMe. Η αποσαφήνιση της βάσης των δεδομένων που θα περιέχει αυτή, είναι πολύ σημαντική, καθώς καθορίζεται η αποθήκευση των δεδομένων και ο τρόπος με τον οποίο, μέσω ερωτημάτων, μπορούν να ανακτηθούν δεδομένα από τη βάση αυτή.

Ποιο συγκεκριμένα στην Εικόνα 5, βλέπουμε ότι η εφαρμογή απαιτεί μια πολύ απλή βάση, η οποία αποτελείται από δύο πίνακες. Τον πίνακα «user» και τον πίνακα «notification». Στον πίνακα «user» αποθηκεύονται τα στοιχεία του χρήστη (αριθμός τηλεφώνου), όταν κάποιος κάνει εγγραφή στην εφαρμογή. Στον πίνακα «notification» αποθηκεύονται τα μηνύματα που ανταλλάσουν οι χρήστες μεταξύ τους.



Εικόνα 5 Σχήμα Βάσης Δεδομένων

Δημιουργία Server

Για το κομμάτι του server θα χρειαστούμε αρχικά να δημιουργήσουμε μια βάση δεδομένων για να αποθηκεύονται εκεί τα δεδομένα για την σωστή λειτουργία της εφαρμογής, και στην συνέχεια ένα περιβάλλον ανάπτυξης Java. Η βάση που θα χρησιμοποιηθεί είναι MySQL, και ο λόγος είναι η ευκολία με την οποία μπορεί να συνδεθεί σε αυτήν μια Java εφαρμογή, όπως ο server για την συγκεκριμένη άσκηση.

Εγκατάσταση απαραίτητων Εργαλείων

Εγκατάσταση MySQL

Για να γίνει η εγκατάσταση της MySQL πρέπει να ακολουθηθούν τα εξής βήματα:

1. Κατέβασμα του αρχείου εγκατάστασης από την σελίδα dev.mysql.com/downloads/.
2. Μετά το τέλος του κατεβάσματος του αρχείου, πρέπει να ανοιχτεί και να ξεκινήσει η εγκατάσταση.

Εικόνα 6 Οδηγός εγκατάστασης MySQL

3. Επιλέγοντας «Install MySQL Products» προχωράμε στην εγκατάσταση, και στο επόμενο παράθυρο που εμφανίζεται, αναγράφονται οι όροι της GNU General Public License, που επιτρέπει την χρήση του λογισμικού αυτού. Μετά την ανάγνωση της άδειας επιλέγεται το check box αποδοχής των όρων και στην συνέχεια το κουμπί next.
4. Στο επόμενο παράθυρο του οδηγού υπάρχουν επιλογές για την τοποθεσία που θα γίνει η εγκατάσταση, και για τον τρόπο της. Εδώ υπάρχουν οι εξής επιλογές:
    - Developer
    - Server Only

- Client Only
- Full
- Custom

Κάθε μία από αυτές τις επιλογές, επιλέγει ποια στοιχεία θα εγκατασταθούν. Κάνουμε τις επιλογές που θέλουμε και πατάμε next.

5. Ακολουθώντας τον οδηγό ολοκληρώνεται η εγκατάσταση των απαραίτητων στοιχείων για την λειτουργία της MySQL
6. Μετά την ολοκλήρωση της εγκατάστασης, έχουμε την αρχική παραμετροποίηση της MySQL, στο επόμενο παράθυρο του οδηγού. Επιλέγουμε τον τύπο του μηχανήματος που θα τρέχει η MySQL, την πόρτα στην οποία θα μιλάει / ακούει. Τέλος επιλέγουμε το check box: advanced options και πατάμε next.
7. Σε αυτό το παράθυρο επιλέγουμε τον κωδικό του root χρήστη, για σύνδεση στην MySQL.
8. Σε αυτό το παράθυρο έχουμε επιλογή να επιτρέπουμε στην MySQL να ξεκινάει μαζί με τα Windows.
9. Στο τελευταίο παράθυρο του οδηγού υπάρχουν επιλογές για την διατήρηση Log αρχείων.

Εγκατάσταση Java

Για να γίνει η εγκατάσταση της Java πρέπει να ακολουθηθούν τα εξής βήματα:

1. Κατέβασμα του αρχείου εγκατάστασης από την σελίδα java.com/en/download/.
2. Μετά το τέλος του κατεβάσματος του αρχείου, πρέπει να ανοιχτεί και να ξεκινήσει η εγκατάσταση.

Εικόνα 7 Οδηγός εγκατάστασης Java

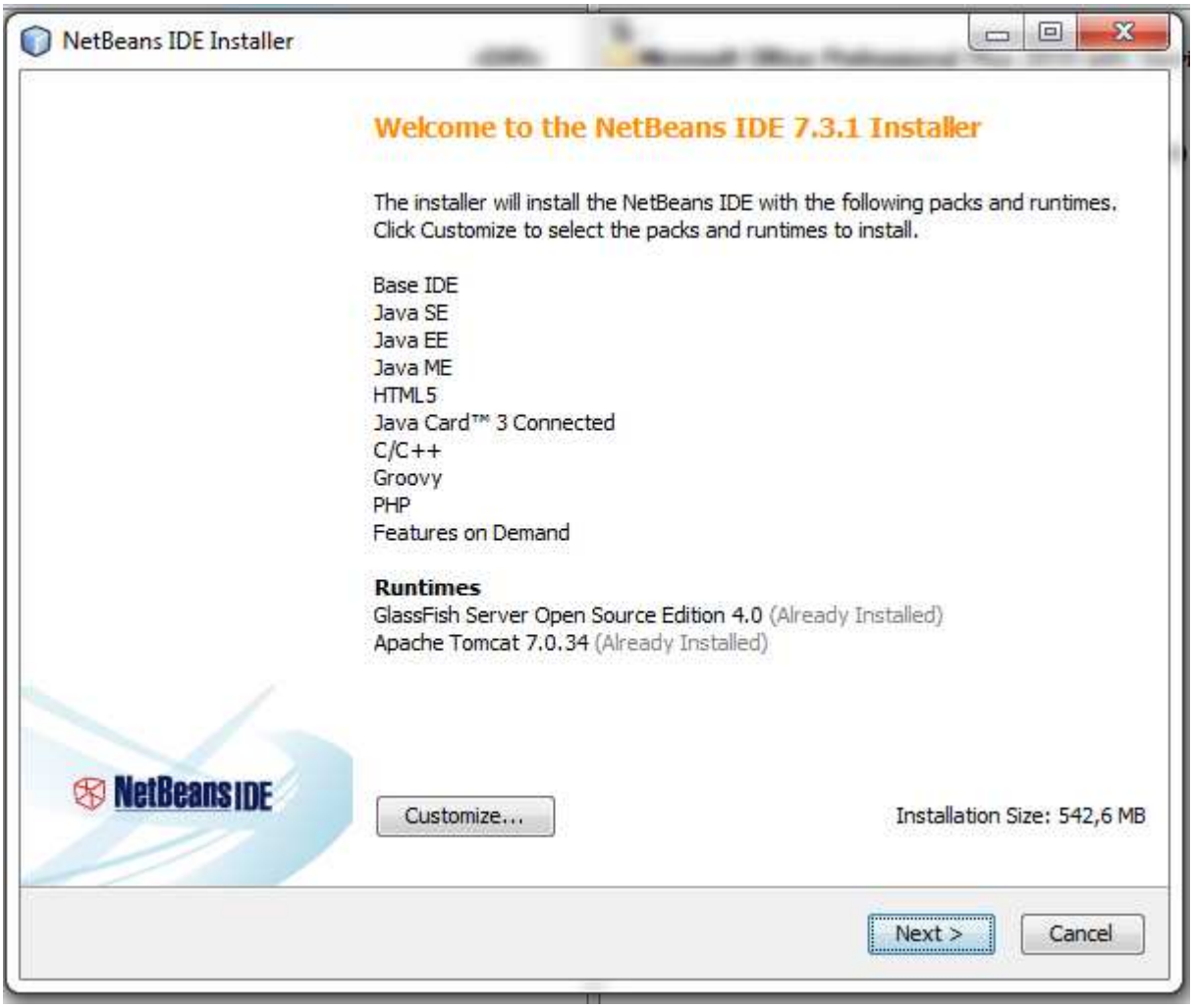
3. Επιλέγοντας το «Change destination folder» έχουμε επιλογή να αλλάξουμε την τοποθεσία εγκατάστασης της Java.

4. Πατώντας το κουμπί «Install» προχωράμε στην εγκατάσταση της Java.

Εγκατάσταση NetBeans

Για να γίνει η εγκατάσταση της Java πρέπει να ακολουθηθούν τα εξής βήματα:

1. Κατέβασμα του αρχείου εγκατάστασης από την σελίδα java.com/en/download/.
2. Μετά το τέλος του κατεβάσματος του αρχείου, πρέπει να ανοιχτεί και να ξεκινήσει η εγκατάσταση.

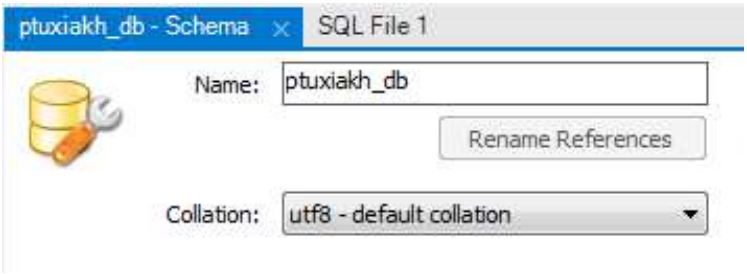


Εικόνα 8 Οδηγός εγκατάστασης NetBeans

3. Πατώντας το κουμπί «Customize» μπορούμε να επιλέξουμε ποια στοιχεία θα εγκατασταθούν.
4. Στο επόμενο παράθυρο αναγράφεται η άδεια χρήσης για το NetBeans.
5. Στο επόμενο παράθυρο του οδηγού αναγράφεται η άδεια χρήσης του JUnit.
6. Στο επόμενο παράθυρο επιλέγουμε την τοποθεσία που θα εγκατασταθεί το NetBeans, καθώς και σε ποια τοποθεσίας είναι εγκατεστημένη η Java που θα χρησιμοποιεί το πρόγραμμα.
7. Έπειτα προχωράμε με την εγκατάσταση του προγράμματος.

Δημιουργία Βάσης

Για την δημιουργία της βάσης δεδομένων, θα χρησιμοποιήσουμε το πρόγραμμα MySQL Workbench, που έχει εγκατασταθεί προηγουμένως κατά την εγκατάσταση της MySQL.



Εικόνα 9 Δημιουργία Βάσης Δεδομένων στην MySQL

1. Τρέξιμο του MySQL Workbench.
2. Σύνδεση στο instance της MySQL.
3. Από την γραμμή εργαλείων επιλέγουμε το κουμπί «Create a new schema in the current server».
4. Επιλέγουμε το όνομα της βάσης που θέλουμε να δημιουργήσουμε, και για Collation επιλέγουμε utf8-default collation.
5. Με την συμπλήρωση των παραπάνω στοιχείων, πατάμε το κουμπί Apply, και δημιουργείτε η βάση δεδομένων.

Δημιουργία Πινάκων

Για την δημιουργία των απαραίτητων πινάκων, για την λειτουργία της εφαρμογής, χρησιμοποιούμε τον παρακάτω MySQL κώδικα.

- Πίνακας user

```
CREATE TABLE `user` (
  `UserID` int(11) NOT NULL AUTO_INCREMENT,
  `Phone` varchar(10) NOT NULL,
  `isOnline` bit(1) DEFAULT NULL,
  `lastUpdated` datetime DEFAULT NULL,
  PRIMARY KEY (`UserID`),
  UNIQUE KEY `Phone_UNIQUE` (`Phone`)
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8;
```

Εικόνα 10 Πίνακας user

- Πίνακας notification

```
CREATE TABLE `notification` (
  `NotificationID` int(11) NOT NULL AUTO_INCREMENT,
  `UserIDFrom` int(11) NOT NULL,
  `UserIDTo` int(11) NOT NULL,
  `Message` text,
  `TimeStamp` datetime DEFAULT NULL,
  `isNotified` bit(1) DEFAULT NULL,
  PRIMARY KEY (`NotificationID`,`UserIDFrom`,`UserIDTo`),
  KEY `UserIDFrom_idx` (`UserIDFrom`),
  KEY `UserIDTo_idx` (`UserIDTo`),
  CONSTRAINT `UserIDFrom` FOREIGN KEY (`UserIDFrom`) REFERENCES `user` (`UserID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `UserIDTo` FOREIGN KEY (`UserIDTo`) REFERENCES `user` (`UserID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Εικόνα 11 Πίνακας notification

Δημιουργία server

Ο κώδικάς για την δημιουργία του server παρατίθεται στο Παράρτημα Α : Πηγαίος Κώδικας NotifyMe.

Δημιουργία Client

Για την δημιουργία του client (εφαρμογή Android) θα χρειαστούμε ένα περιβάλλον ανάπτυξης Java, με κατάλληλα εργαλεία για την δημιουργία Android εφαρμογών.

Εγκατάσταση απαραίτητων Εργαλείων

Εγκατάσταση Eclipse

Το Eclipse δεν χρησιμοποιεί κάποιο αρχείο εγκατάστασης. Το κατεβάζεις στον υπολογιστή και τρέχεις απευθείας το πρόγραμμα. Για την υλοποίηση της παρούσας εργασίας κατεβάζουμε το ADT (Android Developer Tools) Bundle for Windows το οποίο περιλαμβάνει:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator

Δημιουργία client (εφαρμογή)

Ο κώδικάς για την δημιουργία του client παρατίθεται στο Παράρτημα Α : Πηγαίος Κώδικας NotifyMe.

Αποτελέσματα

Συμπεράσματα

Στην εργασία αυτή μελετήθηκε η πλατφόρμα Android με σκοπό τη δημιουργία μίας νέας εφαρμογής, μέσω της οποίας οι χρήστες έχουν την δυνατότητα να στέλνουν σύντομα μηνύματα, μεταξύ τους, μέσω διαδικτύου ή του δικτύου GSM.

Αρχικά έγινε ανάλυση και καταγραφή απαιτήσεων, διαδικασία πάρα πολύ σημαντική για τις αποφάσεις που θα παρθούν κατά τη διάρκεια της υλοποίησης. Μέσα από το στάδιο αυτό, εντοπίζονται οι ανάγκες που πρέπει να ικανοποιούνται από την εφαρμογή. Η εξειδικευμένη αναζήτηση απαιτήσεων από τη σκοπιά του χρήστη για την ικανοποίηση τους, είναι ζωτικής σημασίας, καθώς στη περίπτωση που δεν ικανοποιηθούν αυτές, η εφαρμογή θα απαξιωθεί και δεν θα χρησιμοποιείται, άρα θα υπάρχει αποτυχία.

Στη συνέχεια, καταγράφηκαν τα σενάρια χρήσης, επίσης πολύ σημαντική διαδικασία, καθώς αποτυπώνεται η εφαρμογή σε ένα πρώιμο στάδιο, ικανό να παρουσιαστεί και σε εξωτερικούς χρήστες για την ανταλλαγή απόψεων με στόχο την βέλτιστη απόδοση της επιθυμητής λειτουργικότητας, κατά το στάδιο της υλοποίησης.

Αναλύθηκε και δημιουργήθηκε το πρότυπο της βάσης δεδομένων, για την αποθήκευση και διαχείριση των απαραίτητων δεδομένων. Ζωτικής σημασίας το στάδιο αυτό, καθώς με μία κακοσχεδιασμένη βάση δεδομένων, δημιουργούνται προβλήματα στην ανάκτηση των δεδομένων μέσω ερωτημάτων (SQL).

Συμπερασματικά, από την πτυχιακή αυτή, επεκτάθηκαν σε ικανοποιητικό βαθμό οι γνώσεις μου πάνω σε ένα πολύ ενδιαφέρον ,και αγαπημένο σε εμένα, θέμα. Την υλοποίηση εφαρμογών, μέσω της ανάπτυξης κώδικα.

Μελλοντική Εργασία και Επεκτάσεις

Πολύ σημαντικό μειονέκτημα εδώ, είναι η έλλειψη δυνατότητας αποστολής μηνύματος, μέσω διαδικτύου, σε κάποιος χρήστη που δεν είναι συνδεδεμένος τη συγκεκριμένη στιγμή, και να παραλάβει το μήνυμα όταν θα συνδεθεί. Λειτουργικότητα που υπολογίζεται να προστεθεί σε μελλοντικές αναβαθμίσεις την εφαρμογής.

Επίσης, μία άλλη λειτουργικότητα που μπορεί να προστεθεί σε κάποια μελλοντική αναβάθμιση, είναι η διατήρηση ιστορικού των μηνυμάτων που έχουν σταλεί μεταξύ δύο χρηστών.

Τελικός στόχος που τίθεται για αυτή την εφαρμογή, είναι η φιλοξενία του server σε έναν real server, ώστε να μπορούν οι χρήστες να έχουν πρόσβαση σε αυτόν από οποιοδήποτε σημείο έχουν πρόσβαση στο διαδίκτυο, και έπειτα την ανάρτηση της στο Google Play Store.

Βιβλιογραφία

- Wikipedia – Java, http://en.wikipedia.org/wiki/Java_(programming_language)
- Wikipedia - Java Development Kit, http://en.wikipedia.org/wiki/Java_Development_Kit
- Wikipedia - Android, http://en.wikipedia.org/wiki/Android_(operating_system)
- Wikipedia - Android version history, http://en.wikipedia.org/wiki/Android_version_history
- Wikipedia - Android software development, http://en.wikipedia.org/wiki/Android_software_development
- Eclipse IDE, http://en.wikipedia.org/wiki/Eclipse_(software)
- ADT Plugin for Eclipse, http://developer.android.com/sdk/eclipse-adt.html
- Android Tools, http://developer.android.com/tools/help/index.html
- Android Developing Introduction, http://developer.android.com/guide/developing/index.html

Παράρτημα Α : Πηγαίος Κώδικας NotifyMe

Server Side

Database

```
public class Database {

    ResultSet rs, rs2;
    PreparedStatement st, st2;
    private static Connection con;
    private static final String Driver =
"oracle.jdbc.driver.OracleDriver";
    private static final String ConnectionString =
"jdbc:mysql://localhost:3306/ptuxiakh_db";
    private static final String user = "root";
    private static final String pwd = "123456";
    private CurrentTime time = new CurrentTime();

    public Database() {
    }

    public static Connection loadDriver() throws SQLException {
        try {
            Class.forName(Driver).newInstance();
        } catch (Exception ex) {
            //ignore
        }
        return DriverManager.getConnection(ConnectionString, user,
pwd);
    }

    public boolean addUser(String _Phone) throws SQLException {
        boolean boolResult = false;
        try {
            con = loadDriver();
            boolean isUser = isUser(_Phone);
            if (!isUser) {
                st = con.prepareStatement("INSERT INTO
user(Phone,isOnline,lastUpdated) values(?,?,?)");
                st.setString(1, _Phone);
                st.setBoolean(2, true);
                st.setString(3, time.getTime());
                st.executeUpdate();
                boolResult = true;
```

```
                return boolResult;
            }
        } finally {
            if (rs != null) {
                rs.close();
            }
            if (st != null) {
                st.close();
            }
            if (con != null) {
                con.close();
            }
        }
        return boolResult;
    }

    public void deleteUser(String _Phone) throws SQLException {
        int _UserID;
        try {
            con = loadDriver();
            _UserID = findUserID(_Phone);
            st = con.prepareStatement("DELETE FROM user WHERE
UserID=?");
            st.setInt(1, _UserID);
            st.executeUpdate();
            System.out.println("'" + _Phone + "' left the
application");
        } finally {
            if (rs != null) {
                rs.close();
            }
            if (st != null) {
                st.close();
            }
            if (con != null) {
                con.close();
            }
        }
    }

    public void updUser(String _Phone, String _lastUpdated) throws
SQLException {
        int _UserID = findUserID(_Phone);
        try {
            con = loadDriver();
            st = con.prepareStatement("UPDATE user SET lastUpdated=?
WHERE UserID=?");
            st.setString(1, _lastUpdated);
            st.setInt(2, _UserID);
            st.executeUpdate();
        } finally {
            if (rs != null) {
                rs.close();
            }
            if (st != null) {
                st.close();
```

```
            }
            if (con != null) {
                con.close();
            }
        }
    }

    public void updUser(String _Phone, boolean logged) throws
SQLException {
        try {
            con = loadDriver();
            int _UserID = findUserID(_Phone);
            boolean isOnline = statusUser(_UserID);
            st = con.prepareStatement("UPDATE user SET isOnline=?
WHERE UserID=?");
            st.setBoolean(1, logged);
            st.setInt(2, _UserID);
            st.executeUpdate();
        } finally {
            if (rs != null) {
                rs.close();
            }
            if (st != null) {
                st.close();
            }
            if (con != null) {
                con.close();
            }
        }
    }

    public boolean statusUser(int _UserID) throws SQLException {
        try {
            con = loadDriver();
            st = con.prepareStatement("SELECT isOnline FROM user
WHERE UserID=?");
            st.setInt(1, _UserID);
            rs = st.executeQuery();
            if (rs.next()) {
                boolean isOnline = rs.getBoolean("isOnline");
                return isOnline;
            }
        } finally {
            if (rs != null) {
                rs.close();
            }
            if (st != null) {
                st.close();
            }
//            if (con != null) {con.close();}
        }
        return false;
    }

    public boolean checkLogin(String _Phone) throws SQLException {
        String dbPhone = "";
```

```
        try {
            con = loadDriver();
            int _UserID = findUserID(_Phone);
            if (_UserID != 0) {
                st = con.prepareStatement("SELECT Phone FROM user
WHERE UserID=?");
                st.setInt(1, _UserID);
                rs = st.executeQuery();
                if (rs.next()) {
                    dbPhone = rs.getString("Phone");
                    if (dbPhone.equals(_Phone)) {
                        return true;
                    } else {
                        return false;
                    }
                } else if (!rs.next()) {
                    return false;
                }
            }
        } finally {
            if (rs != null) {
                rs.close();
            }
            if (st != null) {
                st.close();
            }
        }
        return false;
    }

    public boolean isUser(String _Phone) throws SQLException {
        boolean isUser = false;
        try {
            con = loadDriver();
            st = con.prepareStatement("SELECT Phone FROM user WHERE
Phone=?");
            st.setString(1, _Phone);
            rs = st.executeQuery();
            if (rs.next()) {
                isUser = true;
            } else {
                isUser = false;
            }
        } finally {
        }
        return isUser;
    }

    public int findUserID(String _Phone) throws SQLException {
        int _UserID;
        try {
            boolean isUser = isUser(_Phone);
            if (isUser) {
                st = con.prepareStatement("SELECT UserID FROM user
WHERE Phone=?");
                st.setString(1, _Phone);
```

```
                rs = st.executeQuery();
                if (rs.next()) {
                    _UserID = rs.getInt("UserID");
                } else {
                    return 0;
                }
            } else {
                return 0;
            }
        } finally {
        }
        return _UserID;
    }

    public String getPhone(String _Phone) throws SQLException {
        int _UserID;
        try {
            con = loadDriver();

            _UserID = findUserID(_Phone);
            st = con.prepareStatement("SELECT Phone FROM user WHERE
UserID=?");
            st.setInt(1, _UserID);
            rs = st.executeQuery();

            while (rs.next()) {
                _Phone = rs.getString("Phone");
            }
        } finally {
            if (rs != null) {
                rs.close();
            }
            if (st != null) {
                st.close();
            }
            if (con != null) {
                con.close();
            }
        }
        return _Phone;
    }

    public String findUserPhone(int _UserID) throws SQLException {
        String _UserName = "";
        try {
            st = con.prepareStatement("SELECT UserName FROM user
WHERE UserID=?");
            st.setInt(1, _UserID);
            rs = st.executeQuery();
            if (rs.next()) {
                _UserName = rs.getString("UserName");
            }
        } finally {
        }
        return _UserName;
    }
```

```
    public ArrayList<User> getUsers() throws SQLException {
        ArrayList<User> listUsers = new ArrayList<>();
        String _Phone;
        boolean _isOnline;
        int online;
        try {
            con = loadDriver();
            st = con.prepareStatement("SELECT Phone, isOnline FROM
user");
            rs = st.executeQuery();

            while (rs.next()) {
                _Phone = rs.getString("Phone");
                    System.out.println("Phone: "+ _Phone);
                _isOnline = rs.getBoolean("isOnline");
                    System.out.println("onLine: "+_isOnline);
                if (_isOnline == true) {
                    online = 1;
                } else {
                    online = 0;
                }
                listUsers.add(new User(online, _Phone));
            }
        } finally {
            if (rs != null) {
                rs.close();
            }
            if (st != null) {
                st.close();
            }
            if (con != null) {
                con.close();
            }
        }
        return listUsers;
    }

    public int addNotification(String _UserFrom, String _UserTo,
String _Message, String _Time) throws SQLException {
        int From, To, NotifID = 0;
        try {
            con = loadDriver();
            From = findUserID(_UserFrom);
            To = findUserID(_UserTo);
            st = con.prepareStatement("INSERT INTO
Notification(UserIDFrom,UserIDTo,Message,Timestamp,isNotified)
values(?,?,?,?,?)");
            st.setInt(1, From);
            st.setInt(2, To);
            st.setString(3, _Message);
            st.setString(4, _Time);
            st.setBoolean(5, false);
            st.executeUpdate();
            st2 = con.prepareStatement("SELECT max(NotificationID)
AS NotifID FROM notification ");
```

```
            rs2 = st2.executeQuery();

            while (rs2.next()) {
                NotifID = rs2.getInt("NotifID");
            }
        } finally {
            if (rs != null) {
                rs.close();
            }
            if (st != null) {
                st.close();
            }
            if (rs2 != null) {
                rs2.close();
            }
            if (st2 != null) {
                st2.close();
            }
            if (con != null) {
                con.close();
            }
        }
        return NotifID;
    }
}
```

IncomingMsgHandler

```
public class IncomingMsgHandler {

    public int msgRecipient;
    private Database db = new Database();
    private NotificationCustom notification;
    private ArrayList<User> listUsers = new ArrayList<>();

    public Object messageReceived(MessageCustom message) throws
SQLException {
        MessageCustom _msg = message;
        switch (_msg.type) {
            case "getUsers":
                listUsers.clear();
                System.out.println("getUsers call now");
                listUsers = db.getUsers();
                for(int i=0;i<listUsers.size();i++){

if(message.sender.endsWith(listUsers.get(i).getPhone())){
                        listUsers.remove(i);
                        break;
                    }
                }
                db.updUser(_msg.sender, _msg.msgTime);
                return listUsers;
            case "deleteUser":
                db.deleteUser(_msg.sender);
                db.updUser(_msg.sender, _msg.msgTime);
                break;
```

```
            case "addUser":
                boolean addResult;
                addResult = db.addUser(_msg.sender);
                return addResult;
            case "getLogin":
                if (_msg.flag == 6) {
                    boolean checkLogin = db.checkLogin(_msg.sender);
                    if (checkLogin) {
                        db.updUser(_msg.sender, true);
                        db.updUser(_msg.sender, _msg.msgTime);
                    }
                    return checkLogin;
                } else if (_msg.flag == 7) {
                    db.updUser(_msg.sender, false);
                    db.updUser(_msg.sender, _msg.msgTime);
                }
                break;
            case "message":
                int NotifID = db.addNotification(_msg.sender,
_msg.recipient, _msg.content, _msg.msgTime);
                notification = new NotificationCustom(NotifID,
_msg.sender, _msg.recipient, _msg.content, _msg.msgTime);
                db.updUser(_msg.sender, _msg.msgTime);
                return notification;
        }

        return null;
    }
}
```

ServerThread

```
public class ServerThread extends Thread {

    public T_Server server = null;
    public Socket socket = null;
    public int ID = -1;
    public String username = "";
    public ObjectInputStream streamIn = null;
    public ObjectOutputStream streamOut = null;
//    public ServerFrame ui;

    public ServerThread(T_Server _server, Socket _socket) {
        super();
        server = _server;
        socket = _socket;
        ID = socket.getPort();
//        ui = _server.ui;
    }

    public void send(Object msg) {
        try {
            if (msg != null) {
                streamOut.writeObject(msg);
                streamOut.flush();
                System.out.println("Message has been sent: " + msg);
            } else {
```

```
                System.out.println("Message is null");
            }

//            streamOut.close();
        } catch (IOException ex) {
            System.out.println("Exception [SocketClient :
send(...)]" + ex);
            ex.printStackTrace();
        }
    }

    public int getID() {
        return ID;
    }

    @SuppressWarnings("deprecation")
    public void run() {
        System.out.println("\nServer Thread " + ID + " running.");
        while (true) {
            try {
                MessageCustom msg = (MessageCustom)
streamIn.readObject();
                System.out.println("Message has been read: " + msg);
                server.handle(ID, msg);
            } catch (Exception ioe) {
                System.out.println(ID + " ERROR reading: " + ioe);
                server.remove(ID);
                stop();
            }
        }
    }

    public void open() throws IOException {
        streamOut = new
ObjectOutputStream(socket.getOutputStream());
        streamOut.flush();
        streamIn = new ObjectInputStream(socket.getInputStream());

    }

    public void close() throws IOException {
        if (socket != null) {
            socket.close();
        }
        if (streamIn != null) {
            streamIn.close();
        }
        if (streamOut != null) {
            streamOut.close();
        }
    }
}
```

Server

```
public class Server implements Runnable {

    public ArrayList<ServerThread> clients = new ArrayList(1);
    public ServerSocket server = null;
    public Thread thread = null;
    public int port = 4444;
    public Database db;
    private ObjectOutputStream out;
    private IncomingMsgHandler msgListener;

    public Server(IncomingMsgHandler _msgListener) {
        this.msgListener = _msgListener;

        try {
            server = new ServerSocket(port);
            port = server.getLocalPort();
            System.out.println("Server startet. IP : " +
InetAddress.getLocalHost() + ", Port : " + server.getLocalPort());
            start();
        } catch (IOException ioe) {
            System.out.println("Can not bind to port : " + port +
"\nRetrying");
        }
    }

    public Server(IncomingMsgHandler _msgListener, int _port) {
        this.msgListener = _msgListener;
        port = _port;
        clients = new ArrayList<>();

        try {
            server = new ServerSocket(port);
            port = server.getLocalPort();
            System.out.println("Server startet. IP : " +
InetAddress.getLocalHost() + ", Port : " + server.getLocalPort());
            start();
        } catch (IOException ioe) {
            System.out.println("Can not bind to port : " + port +
"\nRetrying");
        }
    }

    public void run() {
        while (thread != null) {
            try {
                System.out.println("\nWaiting for a client ...");
                addThread(server.accept());
            } catch (Exception ioe) {
                System.out.println("\nServer accept error: \n" +
ioe);
            }
```

```
        }
    }

    public void start() {
        if (thread == null) {
            thread = new Thread(this);
            thread.start();
        }
    }

    @SuppressWarnings("deprecation")
    public void stop() {
        if (thread != null) {
            thread.stop();
            thread = null;
        }
    }

    private int findClient(int ID) {
        for (int i = 0; i < clients.size(); i++) {
            if (clients.get(i).getID() == ID) {
                return i;
            }
        }
        return -1;
    }

    public void handle(int ID, MessageCustom _msg) throws
SQLException, IOException {
        System.out.println("Message Type is: " + _msg.type);
        ArrayList<User> listUsers=new ArrayList<>();
        NotificationCustom notification;
        boolean loginStatus,addStatus;
        switch (_msg.type) {
            case "message":
                notification = (NotificationCustom)
msgListener.messageReceived(_msg);
                System.out.println("Message has been processed: " +
notification);
                findUserThread(_msg.recipient).send(notification);
                break;
            case "deleteUser":
                msgListener.messageReceived(_msg);
                break;
            case "getLogin":
                loginStatus = (boolean)
msgListener.messageReceived(_msg);
                if (_msg.flag == 6) {
                    if (loginStatus) {
                        System.out.println("username: " +
clients.get(findClient(ID)).username);
                        if
(clients.get(findClient(ID)).username.equals("")) {
                            clients.get(findClient(ID)).username =
_msg.sender;
```

```
clients.get(findClient(ID)).send(loginStatus);
                        } else {

clients.get(findClient(ID)).send(loginStatus);
                        }
                    } else if (!loginStatus) {

clients.get(findClient(ID)).send(loginStatus);
                    }
                } else if (_msg.flag == 7) {
                    System.out.println("Message has been processed:
getLogin and log out");
                    remove(findClient(ID));
                }
                break;
            case "getUsers":
                listUsers.clear();
                listUsers = (ArrayList<User>)
msgListener.messageReceived(_msg);
                System.out.println("Message has been processed: " +
listUsers);
                clients.get(findClient(ID)).send(listUsers);
                break;
            case "addUser":
                addStatus = (boolean)
msgListener.messageReceived(_msg);
                System.out.println("Message has been processed: " +
addStatus);
                clients.get(findClient(ID)).send(addStatus);
                break;
        }
    }

    public ServerThread findUserThread(String usr) {
        for (int i = 0; i < clients.size(); i++) {
            if (clients.get(i).username.equals(usr)) {
                return clients.get(i);
            }
        }
        return null;
    }

    @SuppressWarnings("deprecation")
    public synchronized void remove(int ID) {
        int pos = findClient(ID);
        if (pos >= 0) {
            ServerThread toTerminate = clients.get(pos);
            System.out.println("\nRemoving client thread " + ID + "
at " + pos);
            if (pos < clients.size() - 1) {
                for (int i = pos + 1; i < clients.size(); i++) {
                    clients.set(i - 1, clients.get(i));
                }
            }
            try {
```

```
                toTerminate.close();
            } catch (IOException ioe) {
                System.out.println("\nError closing thread: " +
ioe);
            }
            toTerminate.stop();
        }
    }

    private void addThread(Socket _socket) {
        System.out.println("\nClient accepted: " + _socket);
        int clientCount = clients.size();
        ServerThread newThread = new ServerThread(this, _socket);
        clients.add(newThread);
        try {
            clients.get(clientCount).open();
            clients.get(clientCount).start();

        } catch (IOException ioe) {
            System.out.println("\nError opening thread: " + ioe);
        }
    }

    public void send(Object _msg) {
        if (out != null) {
            try {
                out.writeObject(_msg);
                System.out.println("Data has been sent: " + _msg);
                out.flush();
            } catch (IOException ex) {
                System.out.println("Error when sending: " + ex);
            }
        }
    }

    public static void main(String[] args) {
        IncomingMsgHandler _msgListener = new IncomingMsgHandler();
        Server server = new Server(_msgListener);

    }
}
```

Client Side

SplashScreen

```
public class SplashScreen extends Activity {
	private static final String TAG = "SplashScreen";
	private static final String tagHandler = "SplashScreen msgHandle";
	public static Activity actSplash;

	private ContantsClass getConst = new ContantsClass();

	Messenger mService = null;
	MessageCustom msgSend, msgSendUserGet;
	private Button btnSignUp, btnLogin, btnTestNet;
	private boolean mIsBound, isRegistered, netAccess;
	private String userPhone, contactName;
	private ArrayList<User> users;
	private final ArrayList<ContactEntry> cont = new
ArrayList<ContactEntry>();
	private ArrayList<ContactEntry> contSorted = new
ArrayList<ContactEntry>();
	private final ListSort listToSort = new ListSort();
	private CurrentTime msgTime = new CurrentTime();
	private IncomingMsgHandler handle = new IncomingMsgHandler();
	public SharedPreferences prefs;
	public SharedPreferences.Editor editor;

	final Messenger mMessenger = new Messenger(new IncomingHandler());

	@SuppressLint("HandlerLeak")
	class IncomingHandler extends Handler {
		@Override
		public void handleMessage(Message msg) {
			msgHandle(msg.obj);
			super.handleMessage(msg);
		}
	}

	@SuppressWarnings({ "static-access", "unchecked" })
	public void msgHandle(Object msgIn) {
		if (msgIn instanceof Boolean) {
			boolean boolResult = (Boolean) handle.msgHandle(msgIn,
getConst.HANDLE_LOGIN);
			Log.i(tagHandler, "boolResult: " + boolResult);
			if (boolResult) {
				Log.i(tagHandler, "msgSendUserGet will send");
				msgSendUserGet = null;
				msgSendUserGet = new MessageCustom(userPhone,
"getUsers", msgTime.getTime());
				sendMessageToService(msgSendUserGet);
			}
		} else if (msgIn instanceof ArrayList<?>) {
			Log.i(tagHandler, "ArrayList recieved");
			users = (ArrayList<User>) handle.msgHandle(msgIn,
getConst.HANDLE_USERS);

			editor.putBoolean("launch", true);
```

```
                editor.commit();

                queryAllRawContacts();
                msgSendUserGet = null;
                Intent myIntent = new Intent(SplashScreen.this,
MainActivity.class);
                myIntent.putParcelableArrayListExtra("Contacts",
contSorted);
                myIntent.putParcelableArrayListExtra("Users", users);
                doUnbindService();
                SplashScreen.this.startActivity(myIntent);
                finish();
            }
        }

        private ServiceConnection mConnection = new ServiceConnection() {
            public void onServiceConnected(ComponentName className, IBinder
service) {
                mService = new Messenger(service);
                try {
                    Message msg = Message.obtain(null,
ConnectionService.MSG_REGISTER_CLIENT);
                    msg.replyTo = mMessenger;
                    mService.send(msg);
                } catch (RemoteException e) {
                    // In this case the service has crashed before we
could even do
                    // anything
                    // with it
                }
            }

            public void onServiceDisconnected(ComponentName className) {
                // This is called when the connection with the service
has been
                // unexpectedly disconnected - process crashed.
                mService = null;
            }
        };

        private boolean haveNetworkConnection() {
            boolean haveConnectedWifi = false;
            boolean haveConnectedMobile = false;

            ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo[] netInfo = cm.getAllNetworkInfo();
            for (NetworkInfo ni : netInfo) {
                if (ni.getTypeName().equalsIgnoreCase("WIFI"))
                    if (ni.isConnected())
                        haveConnectedWifi = true;
                if (ni.getTypeName().equalsIgnoreCase("MOBILE"))
                    if (ni.isConnected())
                        haveConnectedMobile = true;
            }
            return haveConnectedWifi || haveConnectedMobile;
        }

        @Override
        protected void onDestroy() {
            doUnbindService();
```

```
		super.onDestroy();
	}

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.splash);

		actSplash=this;

		btnLogin = (Button) findViewById(R.id.btnSpLogin);
		btnLogin.setBackgroundColor(Color.TRANSPARENT);
		btnSignUp = (Button) findViewById(R.id.btnSpSignUp);
		btnSignUp.setBackgroundColor(Color.TRANSPARENT);
		btnTestNet = (Button) findViewById(R.id.btnTestNet);
		btnTestNet.setBackgroundColor(Color.TRANSPARENT);

		btnLogin.setOnClickListener(btnLoginOnClickListener);
		btnSignUp.setOnClickListener(btnSignUpOnClickListener);
		btnTestNet.setOnClickListener(btnTestNetOnClickListener);

		prefs = getSharedPreferences("NotifyMe", MODE_PRIVATE);
		userPhone = prefs.getString("Phone", "");
		Log.e(TAG, "userPhone: " + userPhone);
		editor = prefs.edit();

		isRegistered = prefs.getBoolean("isRegistered", false);
		// Log.v(TAG, "Check if Registered: " + isRegistered);

		startService(new Intent(SplashScreen.this,
ConnectionService.class));

		// Log.v(TAG, "CheckIfServiceIsRunning");
		CheckIfServiceIsRunning();

		Handler handlerMain = new Handler();
		Runnable rMain = new Runnable() {
			public void run() {
				btnTestNet.setPressed(true);
				btnTestNet.invalidate();
				Handler handler1 = new Handler();
				Runnable r1 = new Runnable() {
					public void run() {
						btnTestNet.setPressed(false);
						btnTestNet.invalidate();
						btnTestNet.performClick();
					}
				};
				handler1.post(r1);
			}
		};
		Log.v(TAG, "net test button");
		handlerMain.post(rMain);
	}

	private void CheckIfServiceIsRunning() {
		// If the service is running when the activity starts, we want
to
		// automatically bind to it.
		Log.i(TAG, "CheckIfServiceIsRunning here: " +
ConnectionService.isRunning());
```

```
            if (ConnectionService.isRunning()) {
                  doBindService();
            } else if (!ConnectionService.isRunning()) {
                  doBindService();
            }
      }

      public void loging(View v) {
            v.performClick();
      }

      private void sendMessageToService(MessageCustom _msg) {
            Log.v(TAG, "mIsBound: " + mIsBound);
            if (mIsBound) {
                  if (mService != null) {
                        try {
                              Message msg = new Message();
                              Log.v(TAG, "sendMessageToService Message" +
_msg.toString());
                              msg.obj = _msg;
                              msg.what =
ConnectionService.MSG_SET_INT_VALUE;
                              msg.replyTo = mMessenger;
                              mService.send(msg);
                        } catch (RemoteException e) {
                        }
                  }
            }
      }

      void doBindService() {
            Log.i(TAG, "doBindService now");
            bindService(new Intent(this, ConnectionService.class),
mConnection, Context.BIND_AUTO_CREATE);
            mIsBound = true;
      }

      void doUnbindService() {
            if (mIsBound) {
                  // If we have received the service, and hence registered
with it,
                  // then now is the time to unregister.
                  if (mService != null) {
                        try {
                              Message msg = Message.obtain(null,
ConnectionService.MSG_UNREGISTER_CLIENT);
                              msg.replyTo = mMessenger;
                              mService.send(msg);
                        } catch (RemoteException e) {
                              // There is nothing special we need to do if
the service has
                              // crashed.
                        }
                  }
                  // Detach our existing connection.
                  unbindService(mConnection);
                  mIsBound = false;
            }
      }
```

```
	private OnClickListener btnSignUpOnClickListener = new
OnClickListener() {
		public void onClick(View v) {
			Intent myIntent = new Intent(SplashScreen.this,
RegisterActivity.class);
			doUnbindService();
			SplashScreen.this.startActivity(myIntent);
			finish();
		}
	};
	private OnClickListener btnLoginOnClickListener = new
OnClickListener() {
		public void onClick(View v) {
			Log.i(TAG, "Login button pressed");
			msgSend = new MessageCustom(userPhone, "getLogin",
msgTime.getTime(), 6);
			sendMessageToService(msgSend);
		}
	};
	private OnClickListener btnTestNetOnClickListener = new
OnClickListener() {
		public void onClick(View v) {
			netAccess = haveNetworkConnection();
			Log.i(TAG, "netAccess: " + netAccess);
			if (!netAccess) {
				Toast.makeText(SplashScreen.this, "No connection",
Toast.LENGTH_LONG).show();
				stopService(new Intent(SplashScreen.this,
ConnectionService.class));
				doUnbindService();
				SplashScreen.this.finish();
			} else if (netAccess) {
					Log.e(TAG, "netAccess is true");
					if (isRegistered) {
						Log.e(TAG, "Login button will press");

						Handler handler = new Handler();
						Runnable r = new Runnable() {
							public void run() {
								btnLogin.setPressed(true);
								btnLogin.invalidate();
								Handler handler1 = new
Handler();

								Runnable r1 = new
Runnable() {

	btnLogin.setPressed(false);

	btnLogin.invalidate();

	btnLogin.performClick();

			public void run() {

			}
		};
		handler1.post(r1);

	}
};
handler.post(r);
```

```
                        } else if (!isRegistered) {
                              Log.e(TAG, "Register button will
press");
                              Handler handler = new Handler();
                              Runnable r = new Runnable() {
                                    public void run() {
                                          btnSignUp.setPressed(true);
                                          btnSignUp.invalidate();
                                          Handler handler1 = new
Handler();
                                          Runnable r1 = new
Runnable() {
                                                public void run() {
      btnSignUp.setPressed(false);

      btnSignUp.invalidate();

      btnSignUp.performClick();

                                                }
                                          };
                                          handler1.post(r1);

                                    }
                              };
                              handler.post(r);

                        }
                  }
            }
      };

      private void queryAllRawContacts() {

            final String[] projection = new String[] {
RawContacts.CONTACT_ID, RawContacts.DELETED };

            final Cursor rawContacts =
getContentResolver().query(RawContacts.CONTENT_URI, projection, null, //
selection,


                  // retrieve all


                  // entries
                        null, // not required because selection does not
contain
                                    // parameters
                        null); // do not order

            final int contactIdColumnIndex =
rawContacts.getColumnIndex(RawContacts.CONTACT_ID);
            final int deletedColumnIndex =
rawContacts.getColumnIndex(RawContacts.DELETED);

            if (rawContacts.moveToFirst()) {
                  while (!rawContacts.isAfterLast()) {
                        final int contactId =
rawContacts.getInt(contactIdColumnIndex);
```

```
                        final boolean deleted =
(rawContacts.getInt(deletedColumnIndex) == 1);
                        if (!deleted) {
                              contactName = getContactName(contactId);
                              createContactEntry(contactId, cont,
contactName);
                        }
                        rawContacts.moveToNext(); // move to the next entry
                  }
            }

            rawContacts.close();

            contSorted = listToSort.getSortedList(cont);
      }

      public String getContactName(int contactId) {
            final String[] projection = new String[] {
Contacts.DISPLAY_NAME };

            final Cursor contact =
getContentResolver().query(Contacts.CONTENT_URI, projection, Contacts._ID +
"=?", // filter on contact id
                        new String[] { String.valueOf(contactId) }, // the
parameter to

                  // which the contact

                  // id column is

                  // compared to
                        null);

            if (contact.moveToFirst()) {
                  final String name =
contact.getString(contact.getColumnIndex(Contacts.DISPLAY_NAME));
                  contact.close();
                  return name;
            }
            contact.close();
            return null;
      }

      public void createContactEntry(int contactId, List<ContactEntry>
cont, String name) {
            final String[] projection = new String[] { Phone.NUMBER, };
            final Cursor phone =
getContentResolver().query(Phone.CONTENT_URI, projection, Data.CONTACT_ID +
"=?", new String[] { String.valueOf(contactId) }, null);

            if (phone.moveToFirst()) {
                  final int contactNumberColumnIndex =
phone.getColumnIndex(Phone.NUMBER);

                  String number =
phone.getString(contactNumberColumnIndex);
                  number = number.replace(" ", "");
                  number = number.replace("-", "");
                  number = number.replace("(", "");
                  number = number.replace(")", "");
                  number = number.trim();
```

```
                  cont.add(new ContactEntry(contactId, name, number, 0,
0));

            }
            phone.close();
      }
}
```

RegisterActivity

```
public class RegisterActivity extends Activity {
      private static final String TAG = "RegisterActivity";
      private static final String tagHandler = "RegisterActivity
msgHandle";

      private ContantsClass getConst=new ContantsClass();
      private ArrayList<User> users;
      private final ArrayList<ContactEntry> cont = new
ArrayList<ContactEntry>();
      private ArrayList<ContactEntry> contSorted = new
ArrayList<ContactEntry>();
      private final ListSort listToSort = new ListSort();

      private EditText txtPhone;
      private Button btnSignUp, btnCancel;
      Messenger mService = null;
      MessageCustom msgSend,msgSendUserGet;
      private boolean mIsBound;
      private String contactName;
      private CurrentTime msgTime = new CurrentTime();
      private IncomingMsgHandler handle = new IncomingMsgHandler();
      public SharedPreferences prefs;
      public SharedPreferences.Editor editor;

      final Messenger mMessenger = new Messenger(new IncomingHandler());

      @SuppressLint("HandlerLeak")
      class IncomingHandler extends Handler {
            @Override
            public void handleMessage(Message msg) {
                  msgHandle(msg.obj);
                  super.handleMessage(msg);
            }
      }

      @SuppressWarnings({ "static-access", "unchecked" })
      public void msgHandle(Object msgIn) {
            boolean boolResult;
            if (msgIn instanceof Boolean) {
            if (msgSend.type.equals("addUser")) {
                  boolResult = (Boolean)
handle.msgHandle(msgIn,getConst.ADD_USER);
                  if (boolResult) {
                        Log.w(tagHandler, "write in SharedPreferences");
                        editor.putString("Phone",
txtPhone.getText().toString());
                        editor.putBoolean("isRegistered", true);
```

```
                        editor.commit();

                        msgSend = null;
                        msgSend = new
MessageCustom(prefs.getString("Phone", ""), "getLogin", msgTime.getTime(),
6);
                        sendMessageToService(msgSend);
                  }else if (!boolResult) {
                        Toast.makeText(this, "Sign up failed",
Toast.LENGTH_LONG).show();
                        stopService(new Intent(RegisterActivity.this,
ConnectionService.class));
                        doUnbindService();
                        finish();
                  }
            } else if (msgSend.type.equals("getLogin")) {
                  boolResult = (Boolean) handle.msgHandle(msgIn,
getConst.HANDLE_LOGIN);
                  if (boolResult) {
                        editor.putString("Phone",
txtPhone.getText().toString());
                        editor.commit();


                        msgSendUserGet = null;
                        msgSendUserGet = new
MessageCustom(prefs.getString("Phone", ""), "getUsers", msgTime.getTime());
                        sendMessageToService(msgSendUserGet);

                  }else if (!boolResult) {
                        Toast.makeText(this, "Login failed",
Toast.LENGTH_LONG).show();
                        stopService(new Intent(RegisterActivity.this,
ConnectionService.class));
                        doUnbindService();
                        finish();
                  }
            }
            } else if (msgIn instanceof ArrayList<?>) {
                  Log.i(tagHandler, "ArrayList recieved");
                  users = (ArrayList<User>) handle.msgHandle(msgIn,
getConst.HANDLE_USERS);

                  editor.putBoolean("launch", true);
                  editor.commit();

                  queryAllRawContacts();
                  msgSendUserGet = null;
                  Intent myIntent = new Intent(RegisterActivity.this,
MainActivity.class);
                  myIntent.putParcelableArrayListExtra("Contacts",
contSorted);
                  myIntent.putParcelableArrayListExtra("Users", users);
                  doUnbindService();
                  RegisterActivity.this.startActivity(myIntent);
                  finish();
            }
      }

      private ServiceConnection mConnection = new ServiceConnection() {
```

```
		public void onServiceConnected(ComponentName className, IBinder
service) {
			mService = new Messenger(service);
			try {
				Message msg = Message.obtain(null,
ConnectionService.MSG_REGISTER_CLIENT);
				msg.replyTo = mMessenger;
				mService.send(msg);
			} catch (RemoteException e) {
				// In this case the service has crashed before we
could even do
				// anything
				// with it
			}
		}

		public void onServiceDisconnected(ComponentName className) {
			// This is called when the connection with the service
has been
			// unexpectedly disconnected - process crashed.
			mService = null;
		}
	};

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.register);

		txtPhone = (EditText) findViewById(R.id.txtRegPhone);
		btnSignUp = (Button) findViewById(R.id.btnSignUp);
		btnCancel = (Button) findViewById(R.id.btnCancel);

		btnSignUp.setOnClickListener(btnSignUpOnClickListener);
		btnCancel.setOnClickListener(btnCancelOnClickListener);

		txtPhone.requestFocus();
		startService(new Intent(RegisterActivity.this,
ConnectionService.class));

		CheckIfServiceIsRunning();

		prefs = getSharedPreferences("NotifyMe", MODE_PRIVATE);
		editor = prefs.edit();

	}


	private void CheckIfServiceIsRunning() {
		// If the service is running when the activity starts, we want
to
		// automatically bind to it.
		Log.i(TAG, "CheckIfServiceIsRunning here: " +
ConnectionService.isRunning());
		if (ConnectionService.isRunning()) {
			doBindService();
		} else if (!ConnectionService.isRunning()) {
			doBindService();
		}
	}
```

```
	private void sendMessageToService(MessageCustom _msg) {
		Log.v(TAG, "mIsBound: " + mIsBound);
		if (mIsBound) {
			if (mService != null) {
				try {
					Message msg = new Message();
					Log.v(TAG, "sendMessageToService Message" +
_msg.toString());
					msg.obj = _msg;
					msg.what =
ConnectionService.MSG_SET_INT_VALUE;
					msg.replyTo = mMessenger;
					mService.send(msg);
				} catch (RemoteException e) {
				}
			}
		}
	}

	void doBindService() {
		Log.i(TAG, "doBindService now");
		bindService(new Intent(this, ConnectionService.class),
mConnection, Context.BIND_AUTO_CREATE);
		mIsBound = true;
	}

	void doUnbindService() {
		if (mIsBound) {
			// If we have received the service, and hence registered
with it,
			// then now is the time to unregister.
			if (mService != null) {
				try {
					Message msg = Message.obtain(null,
ConnectionService.MSG_UNREGISTER_CLIENT);
					msg.replyTo = mMessenger;
					mService.send(msg);
				} catch (RemoteException e) {
					// There is nothing special we need to do if
the service has
					// crashed.
				}
			}
			// Detach our existing connection.
			unbindService(mConnection);
			mIsBound = false;
		}
	}

	private OnClickListener btnSignUpOnClickListener = new
OnClickListener() {
		public void onClick(View v) {
			Log.i(TAG, "SignUp button pressed");
			msgSend = new
MessageCustom(txtPhone.getText().toString(), "addUser", msgTime.getTime(),
5);
			sendMessageToService(msgSend);
		}
	};
	private OnClickListener btnCancelOnClickListener = new
OnClickListener() {
```

```
            public void onClick(View v) {
                  Log.i(TAG, "Cancel button pressed");
                  doUnbindService();
                  stopService(new Intent(RegisterActivity.this,
ConnectionService.class));
                  finish();
            }
      };
      @Override
      protected void onDestroy() {
            super.onDestroy();
            try {
                  doUnbindService();
            } catch (Throwable t) {
                  Log.e("RegisterActivity", "Failed to unbind from the
service", t);
            }
      }

      private void queryAllRawContacts() {

            final String[] projection = new String[] {
RawContacts.CONTACT_ID, RawContacts.DELETED };

            final Cursor rawContacts =
getContentResolver().query(RawContacts.CONTENT_URI, projection, null, //
selection,


                  // retrieve all


                  // entries
                        null, // not required because selection does not
contain
                                    // parameters
                        null); // do not order

            final int contactIdColumnIndex =
rawContacts.getColumnIndex(RawContacts.CONTACT_ID);
            final int deletedColumnIndex =
rawContacts.getColumnIndex(RawContacts.DELETED);

            if (rawContacts.moveToFirst()) {
                  while (!rawContacts.isAfterLast()) {
                        final int contactId =
rawContacts.getInt(contactIdColumnIndex);
                        final boolean deleted =
(rawContacts.getInt(deletedColumnIndex) == 1);
                        if (!deleted) {
                              contactName = getContactName(contactId);
                              createContactEntry(contactId, cont,
contactName);
                        }
                        rawContacts.moveToNext(); // move to the next entry
                  }
            }

            rawContacts.close();

            contSorted = listToSort.getSortedList(cont);
```

```
	}


	public String getContactName(int contactId) {
		final String[] projection = new String[] {
Contacts.DISPLAY_NAME };

		final Cursor contact =
getContentResolver().query(Contacts.CONTENT_URI, projection, Contacts._ID +
"=?", // filter on contact id
				new String[] { String.valueOf(contactId) }, // the
parameter to

				// which the contact

				// id column is

				// compared to
				null);

		if (contact.moveToFirst()) {
			final String name =
contact.getString(contact.getColumnIndex(Contacts.DISPLAY_NAME));
			contact.close();
			return name;
		}
		contact.close();
		return null;
	}

	public void createContactEntry(int contactId, List<ContactEntry>
cont, String name) {
		final String[] projection = new String[] { Phone.NUMBER, };
		final Cursor phone =
getContentResolver().query(Phone.CONTENT_URI, projection, Data.CONTACT_ID +
"=?", new String[] { String.valueOf(contactId) }, null);

		if (phone.moveToFirst()) {
			final int contactNumberColumnIndex =
phone.getColumnIndex(Phone.NUMBER);

			String number =
phone.getString(contactNumberColumnIndex);
			number = number.replace(" ", "");
			number = number.replace("-", "");
			number = number.replace("(", "");
			number = number.replace(")", "");
			number = number.trim();
			cont.add(new ContactEntry(contactId, name, number, 0,
0));

		}
		phone.close();
	}

}
```

MainActivity

```
public class MainActivity extends Activity {
	private static final String TAG = "MainActivity";
	private static final String tagHandler = "MainActivity msgHandle";
	private static final String tagMenu = "MainActivity
onContextItemSelected";

	public static Activity main_new;
	private ContantsClass getConst = new ContantsClass();

	PowerManager.WakeLock wakeLock;
	WifiManager.WifiLock wifiLock;

	private GridView gv;
	Messenger mService = null;
	boolean mIsBound;
	private CurrentTime msgTime = new CurrentTime();
	private ListSort listToSort = new ListSort();
	private IncomingMsgHandler handle = new IncomingMsgHandler();
	private MessageCustom msgSendNotification, msgSendUserGet,
msgSendLogin;
	final Messenger mMessenger = new Messenger(new IncomingHandler());
	private final FindUser findUser = new FindUser();
	public SharedPreferences prefs;
	public SharedPreferences.Editor editor;
	private ContactAdapter mAdapter;
	private NotificationManager mNotificationManager;
	private Intent servIntent;
	private boolean launch, autoSMS, contRefresh=false;

	private String contactName, userPhone, phone, autoMsg;
	private NotificationCustom not;
	private String notTime, notSender, notBody, recipient, msgNotify;
	private int notID;

	private ArrayList<User> listUsers;
	private ArrayList<ContactEntry> listContacts;
	private ArrayList<ContactEntry> contSortedisUser = new
ArrayList<ContactEntry>();
	private ArrayList<ContactEntry> listFinal = new
ArrayList<ContactEntry>();

	private final ArrayList<ContactEntry> cont = new
ArrayList<ContactEntry>();

	@SuppressLint("HandlerLeak")
	class IncomingHandler extends Handler {
		@Override
		public void handleMessage(Message msg) {
			msgHandle(msg.obj);
			super.handleMessage(msg);
		}
	}

	@SuppressWarnings({ "unchecked", "static-access" })
	public void msgHandle(Object msgIn) {
		if (msgIn instanceof ArrayList<?>) {
			listUsers.clear();
			listUsers = (ArrayList<User>) handle.msgHandle(msgIn,
getConst.HANDLE_USERS);
```

```
                  listFinal = getFinalList(listContacts, listUsers);
                  mAdapter = null;
                  mAdapter = new ContactAdapter(listFinal,
MainActivity.this);
                  gv.setAdapter(null);
                  gv.setAdapter(mAdapter);

                  msgSendUserGet = null;
            } else if (msgIn instanceof NotificationCustom) {
                  not = (NotificationCustom) handle.msgHandle(msgIn,
getConst.HANDLE_MESSAGE);
                  notTime = not.getTime();
                  notSender = not.getSender();
                  notBody = not.getBody();
                  notID = not.getID();
                  editor.putString("recipient", notSender);
                  editor.putString("notifMessage", notBody);
                  editor.commit();
                  msgNotify = "[" + notTime + " " + notSender + "]: " +
notBody;
                  CreateNotificationBig(null, notID);
            }
      }

      private ArrayList<ContactEntry> getFinalList(ArrayList<ContactEntry>
listContacts, ArrayList<User> users) {

            for (int i = 0; i < users.size(); i++) {
                  String searchForPhone = users.get(i).getPhone();
                  boolean isUser = findUser.listFindItem(listContacts,
searchForPhone);
                  int isOnline = users.get(i).getOnLine();
                  if (isUser) {
                        listContacts.get(findUser.getIndex()).setIsUser(1);

      listContacts.get(findUser.getIndex()).setIsOnline(isOnline);
                  }
            }
            contSortedisUser = listToSort.getSortedListUsers(listContacts);

            return listToSort.getSortedListOnline(contSortedisUser);
      }

      public void CreateNotificationBig(View v, int mId) {
            Intent notificationIntent = new Intent(this,
NotificationReply.class);
            notificationIntent.putExtra("ricipient", notSender);
            PendingIntent intent = PendingIntent.getActivity(this, 0,
notificationIntent, 0);

            NotificationCompat.Builder builder = new
NotificationCompat.Builder(this)//
                        .setSmallIcon(R.drawable.ic_notification)//
                        .setContentTitle(getString(R.string.app_name))//
                        .setContentText(msgNotify)//
                        .setAutoCancel(true)//
                        .setContentIntent(intent)//
                        .setStyle(new
NotificationCompat.BigTextStyle().bigText(msgNotify));//
```

```
            Uri alarmSound =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
            builder.setSound(alarmSound);

            mNotificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
            // mId allows you to update the notification later on.
            mNotificationManager.notify(mId, builder.build());
      }

      public void menuPopup(final String recPhone, String recName, int
online) {
            String[] optionsOnline = { "Push Notification", "Send SMS",
"Make a Call" };
            String[] optionsOffline = { "Send SMS", "Make a Call" };
            AlertDialog.Builder alert = new AlertDialog.Builder(this);

            alert.setTitle(recName);
            if (online == 1) {
                  alert.setItems(optionsOnline, new
DialogInterface.OnClickListener() {

                        @Override
                        public void onClick(DialogInterface dialog, int
which) {
                              switch (which) {
                              case 0:// Push Notification
                                    msgSendNotification = new
MessageCustom(userPhone, recPhone, "message", autoMsg, msgTime.getTime());

      sendMessageToService(msgSendNotification);
                                    msgSendNotification = null;
                                    break;
                              case 1:// Send SMS

      SmsManager.getDefault().sendTextMessage(recPhone, null, autoMsg,
null, null);
                                    break;
                              case 2:// Make a Call
                                    Intent call = new
Intent(Intent.ACTION_CALL);
                                    call.setData(Uri.parse("tel:" +
recPhone));
                                    startActivity(call);
                                    break;
                              }

                        }
                  });
            } else if (online == 0) {
                  alert.setItems(optionsOffline, new
DialogInterface.OnClickListener() {

                        @Override
                        public void onClick(DialogInterface dialog, int
which) {
                              switch (which) {
                              case 0:// Send SMS

      SmsManager.getDefault().sendTextMessage(recPhone, null, autoMsg,
null, null);
```

```
                                    break;
                              case 1:// Make a Call
                                    Intent call = new
Intent(Intent.ACTION_CALL);
                                    call.setData(Uri.parse("tel:" +
recPhone));
                                    startActivity(call);
                                    break;
                              }

                        }
                  });
            }

            alert.show();
      }

      public void createPopup(final String recipient, final String
contactName, final int isOnline) {
            AlertDialog.Builder alert = new AlertDialog.Builder(this);
            // Set an EditText view to get user input
            final EditText input = new EditText(this);
            alert.setView(input);

            if (isOnline == 0) {
                  alert.setTitle("Message");
                  alert.setMessage("SMS to " + contactName);

                  alert.setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int
whichButton) {
                              Editable value = input.getText();

      SmsManager.getDefault().sendTextMessage(recipient, null,
value.toString(), null, null);
                        }
                  });
            } else if (isOnline == 1) {
                  alert.setTitle("Message");
                  alert.setMessage("Message to " + contactName);

                  alert.setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int
whichButton) {
                              Editable value = input.getText();
                              msgSendNotification = new
MessageCustom(userPhone, recipient, "message", value.toString(),
msgTime.getTime());
                              sendMessageToService(msgSendNotification);
                              msgSendNotification = null;
                        }
                  });
            }

            alert.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
                  public void onClick(DialogInterface dialog, int
whichButton) {
                        // Canceled.
```

```
                  }
            });

            alert.show();
      }

      @Override
      public boolean onContextItemSelected(MenuItem item) {
            AdapterView.AdapterContextMenuInfo menuInfo =
(AdapterView.AdapterContextMenuInfo) item.getMenuInfo();
            int pos = menuInfo.position;
            int isOnline = mAdapter.getItem(pos).getIsOnline();
            switch (item.getItemId()) {
            case R.id.msg:
                  recipient = mAdapter.getItem(pos).getContactPhone();
                  editor.putString("recipient", recipient);
                  editor.commit();
                  createPopup(recipient,
mAdapter.getItem(pos).getContactName(), isOnline);
                  Log.v(tagMenu, "msg to: " + recipient);
                  return true;
            case R.id.sms:
                  // startActivity(new Intent(Intent.ACTION_VIEW,
Uri.parse("sms:" +
                  // phone)));
                  recipient = mAdapter.getItem(pos).getContactPhone();
                  Log.v(tagMenu, "sms to: " + recipient);
                  return true;
            case R.id.call:
                  phone = mAdapter.getItem(pos).getContactPhone();
                  Intent call = new Intent(Intent.ACTION_CALL);
                  call.setData(Uri.parse("tel:" + phone));
                  startActivity(call);
                  Log.v(tagMenu, "call to: " + phone);
                  return true;

            default:
                  return super.onContextItemSelected(item);
            }
      }

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            Log.v(TAG, "onCreate");

            main_new = this;
            servIntent = new Intent(MainActivity.this,
ConnectionService.class);

            CheckIfServiceIsRunning();

            gv = (GridView) findViewById(R.id.gvMain);
            prefs = getSharedPreferences("NotifyMe", MODE_PRIVATE);
            editor = prefs.edit();
            userPhone = prefs.getString("Phone", "");

            launch = prefs.getBoolean("launch", false);

            setTitle("Contacts");
```

```
            try {
                  ViewConfiguration config = ViewConfiguration.get(this);
                  Field menuKeyField =
ViewConfiguration.class.getDeclaredField("sHasPermanentMenuKey");
                  if (menuKeyField != null) {
                        menuKeyField.setAccessible(true);
                        menuKeyField.setBoolean(config, false);
                  }
            } catch (Exception ex) {
                  // Ignore
            }
            listUsers = new ArrayList<User>();

            if (launch) {
                  listContacts =
getIntent().getExtras().getParcelableArrayList("Contacts");
                  listUsers =
getIntent().getExtras().getParcelableArrayList("Users");
                  listFinal = getFinalList(listContacts, listUsers);

                  mAdapter = null;
                  mAdapter = new ContactAdapter(listFinal,
MainActivity.this);
                  gv.setAdapter(null);
                  gv.setAdapter(mAdapter);

                  launch = false;
                  editor.putBoolean("launch", launch);
                  editor.commit();

            }
            gv.setLongClickable(true);
            gv.setOnItemClickListener(new OnItemClickListener() {

                  @Override
                  public void onItemClick(AdapterView<?> parent, View view,
int pos, long id) {
                        int isUser = mAdapter.getItem(pos).getIsUser();
                        int isOnline = mAdapter.getItem(pos).getIsOnline();
                        String recipient =
mAdapter.getItem(pos).getContactPhone();
                        String contact =
mAdapter.getItem(pos).getContactName();

                        if (!autoMsg.equals("")) {
                              if (autoSMS) {
                                    if (isUser == 1 && isOnline == 1) {
                                          msgSendNotification = new
MessageCustom(userPhone, recipient, "message", autoMsg, msgTime.getTime());

      sendMessageToService(msgSendNotification);
                                          msgSendNotification = null;
                                    } else {

      SmsManager.getDefault().sendTextMessage(recipient, null, autoMsg,
null, null);
                                    }
                              } else if (!autoSMS) {
                                    menuPopup(recipient, contact,
isOnline);
```

```
                        }
                    }
                }

            });
            gv.setOnCreateContextMenuListener(new
OnCreateContextMenuListener() {
                @Override
                public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenuInfo menuInfo) {
                    Log.v(TAG, "onCreateContextMenu here");
                    MenuInflater inflater = getMenuInflater();
                    AdapterView.AdapterContextMenuInfo info =
(AdapterView.AdapterContextMenuInfo) menuInfo;

                    int pos = info.position;

      menu.setHeaderTitle(mAdapter.getItem(pos).getContactName());

                    if (!autoSMS) {
                        if (mAdapter.getItem(pos).getIsUser() == 1 &&
mAdapter.getItem(pos).getIsOnline() == 1) {
                            inflater.inflate(R.menu.main_list,
menu);
                        } else {

      inflater.inflate(R.menu.main_list_offline, menu);
                        }
                    } else if (autoSMS) {
                        inflater.inflate(R.menu.main_list_autosms,
menu);
                    }
                }
            });
            keepSocketAlive();
      }

      @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 boolean onOptionsItemSelected(MenuItem item) {
            // AdapterContextMenuInfo menuInfo = (AdapterContextMenuInfo)
item.getMenuInfo();
            switch (item.getItemId()) {
            case R.id.update:
                  Log.v(TAG, "Update");
                  contRefresh = false;
                  Update();
                  return true;
            case R.id.refresh:
                  Log.v(TAG, "Refresh");
                  contRefresh = true;
                  Update();
                  return true;
            case R.id.settings:
```

```
                  Log.v(TAG, "SettingsActivity");
                  Intent myIntent = new Intent(MainActivity.this,
SettingsActivity.class);
                  this.startActivity(myIntent);
                  return true;
            case R.id.exit:
                  Log.v(TAG, "Exit");
                  msgSendLogin = new MessageCustom(userPhone, "getLogin",
msgTime.getTime(), 7);
                  sendMessageToService(msgSendLogin);
                  stopService(servIntent);
                  doUnbindService();
                  finish();
                  return true;
            default:
                  return super.onOptionsItemSelected(item);
            }
      }

      public void Update() {
            Log.e(TAG, "Update method");
            if (!contRefresh) {
                  msgSendUserGet = new MessageCustom(userPhone, "getUsers",
msgTime.getTime());
                  sendMessageToService(msgSendUserGet);
            } else if (contRefresh) {
                  queryAllRawContacts();
                  listFinal = getFinalList(listContacts, listUsers);

                  msgSendUserGet = new MessageCustom(userPhone, "getUsers",
msgTime.getTime());
                  sendMessageToService(msgSendUserGet);
            }
      }

      private void CheckIfServiceIsRunning() {
            // If the service is running when the activity starts, we want
to
            // automatically bind to it.
            Log.i(TAG, "CheckIfServiceIsRunning here: " +
ConnectionService.isRunning());
            if (ConnectionService.isRunning()) {
                  doBindService();
            } else if (!ConnectionService.isRunning()) {
                  startService(servIntent);
                  doBindService();
            }
      }

      void doBindService() {
            Log.i(TAG, "doBindService now");
            bindService(new Intent(this, ConnectionService.class),
mConnection, Context.BIND_AUTO_CREATE);
            mIsBound = true;
      }

      void doUnbindService() {
            if (mIsBound) {
                  // If we have received the service, and hence registered
with it,
                  // then now is the time to unregister.
```

```
                if (mService != null) {
                        try {
                                Message msg = Message.obtain(null,
ConnectionService.MSG_UNREGISTER_CLIENT);
                                msg.replyTo = mMessenger;
                                mService.send(msg);
                        } catch (RemoteException e) {
                                // There is nothing special we need to do if
the service has
                                // crashed.
                        }
                }
                // Detach our existing connection.
                unbindService(mConnection);
                mIsBound = false;
        }
    }

    private ServiceConnection mConnection = new ServiceConnection() {
        public void onServiceConnected(ComponentName className, IBinder
service) {
                mService = new Messenger(service);
                try {
                        Message msg = Message.obtain(null,
ConnectionService.MSG_REGISTER_CLIENT);
                        msg.replyTo = mMessenger;
                        mService.send(msg);
                } catch (RemoteException e) {
                        // In this case the service has crashed before we
could even do
                        // anything
                        // with it
                }
        }

        public void onServiceDisconnected(ComponentName className) {
                // This is called when the connection with the service
has been
                // unexpectedly disconnected - process crashed.
                mService = null;
        }
    };

    private void sendMessageToService(MessageCustom _msg) {
        Log.i(TAG, "sendMessageToService here");
        if (mIsBound) {
                if (mService != null) {
                        try {
                                Message msg = new Message();
                                msg.obj = _msg;
                                msg.what =
ConnectionService.MSG_SET_INT_VALUE;
                                msg.replyTo = mMessenger;
                                mService.send(msg);
                        } catch (RemoteException e) {
                        }
                }
        }
    }

    @Override
```

```
	public void onBackPressed() {
		moveTaskToBack(true);
		// super.onBackPressed();
		Log.v(TAG, "onBackPressed");
	}

	@Override
	protected void onPause() {
		super.onPause();
		Log.v(TAG, "onPause");
	}

	@Override
	protected void onStop() {
		super.onStop();
		Log.v(TAG, "onStop");
	}

	@Override
	public void onResume() {
		super.onResume();
		autoSMS = prefs.getBoolean("autoSMS", false);
		autoMsg = prefs.getString("autoMsg", "");
		Update();
		Log.v(TAG, "onResume");
	}

	@SuppressLint("Wakelock")
	@Override
	protected void onDestroy() {
		super.onDestroy();
		try {
			doUnbindService();
			finish();
		} catch (Throwable t) {
			Log.e(TAG, "Failed to unbind from the service", t);
		}
		Log.v(TAG, "onDestroy");
		wifiLock.release();
		wakeLock.release();
	}

	public void keepSocketAlive() {
		WifiManager wMgr = (WifiManager)
getSystemService(Context.WIFI_SERVICE);
		wifiLock = wMgr.createWifiLock(WifiManager.WIFI_MODE_FULL,
"MyWifiLock");
		wifiLock.acquire();

		PowerManager pMgr = (PowerManager)
getSystemService(Context.POWER_SERVICE);
		wakeLock = pMgr.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,
"MyWakeLock");
		wakeLock.acquire();
	}

	public Uri getPhotoUri(Integer contactid) {
		Uri photo = null;
		Cursor photoCur =
getContentResolver().query(ContactsContract.Contacts.CONTENT_URI, null,
ContactsContract.Contacts.IN_VISIBLE_GROUP + " = '1'", null,
```

```
                        ContactsContract.Contacts.DISPLAY_NAME + " COLLATE
LOCALIZED ASC");
            if (photoCur.moveToFirst()) {
                  photoCur.moveToPosition(contactid);
                  Uri person =
ContentUris.withAppendedId(ContactsContract.Contacts.CONTENT_URI,
photoCur.getLong(photoCur.getColumnIndex(ContactsContract.Contacts._ID)));
                  photo = Uri.withAppendedPath(person,
ContactsContract.Contacts.Photo.CONTENT_DIRECTORY);
            }
            photoCur.close();
            return photo;
      }

      private void queryAllRawContacts() {

            final String[] projection = new String[] {
RawContacts.CONTACT_ID, RawContacts.DELETED };

            final Cursor rawContacts =
getContentResolver().query(RawContacts.CONTENT_URI, projection, null, //
selection,


                  // retrieve all


                  // entries
                        null, // not required because selection does not
contain
                                    // parameters
                        null); // do not order

            final int contactIdColumnIndex =
rawContacts.getColumnIndex(RawContacts.CONTACT_ID);
            final int deletedColumnIndex =
rawContacts.getColumnIndex(RawContacts.DELETED);

            if (rawContacts.moveToFirst()) {
                  while (!rawContacts.isAfterLast()) {
                        final int contactId =
rawContacts.getInt(contactIdColumnIndex);
                        final boolean deleted =
(rawContacts.getInt(deletedColumnIndex) == 1);
                        if (!deleted) {
                              contactName = getContactName(contactId);
                              createContactEntry(contactId, cont,
contactName);
                        }
                        rawContacts.moveToNext(); // move to the next entry
                  }
            }

            rawContacts.close();

            listContacts = listToSort.getSortedList(cont);
      }

      public String getContactName(int contactId) {
            final String[] projection = new String[] {
Contacts.DISPLAY_NAME };
```

```
            final Cursor contact =
getContentResolver().query(Contacts.CONTENT_URI, projection, Contacts._ID +
"=?", // filter on contact id
                        new String[] { String.valueOf(contactId) }, // the
parameter to

                  // which the contact

                  // id column is

                  // compared to
                        null);

            if (contact.moveToFirst()) {
                  final String name =
contact.getString(contact.getColumnIndex(Contacts.DISPLAY_NAME));
                  contact.close();
                  return name;
            }
            contact.close();
            return null;
      }

      public void createContactEntry(int contactId, List<ContactEntry>
cont, String name) {
            final String[] projection = new String[] { Phone.NUMBER, };
            final Cursor phone =
getContentResolver().query(Phone.CONTENT_URI, projection, Data.CONTACT_ID +
"=?", new String[] { String.valueOf(contactId) }, null);

            if (phone.moveToFirst()) {
                  final int contactNumberColumnIndex =
phone.getColumnIndex(Phone.NUMBER);

                  String number =
phone.getString(contactNumberColumnIndex);
                  number = number.replace(" ", "");
                  number = number.replace("-", "");
                  number = number.replace("(", "");
                  number = number.replace(")", "");
                  number = number.trim();
                  cont.add(new ContactEntry(contactId, name, number, 0,
0));

            }
            phone.close();
      }
}
```

SettingsActivity

```
public class SettingsActivity extends Activity {
	private static final String TAG = "SettingsActivity";

	private Button btnCancel, btnSave;
	private TextView tvLeave;
	private CheckBox cbAutoSMS;
	private EditText etAutoMsg;

	private boolean autoSMS, mIsBound;
	private String userPhone, autoMsg;
	private Intent servIntent;
	public SharedPreferences prefs;
	public SharedPreferences.Editor editor;
	private MessageCustom msgSendUseDel;


	Messenger mService = null;
	private CurrentTime msgTime = new CurrentTime();
	final Messenger mMessenger = new Messenger(new IncomingHandler());

	@SuppressLint("HandlerLeak")
	class IncomingHandler extends Handler {
		@Override
		public void handleMessage(Message msg) {
			super.handleMessage(msg);
		}
	}

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.settings);

		prefs = getSharedPreferences("NotifyMe", MODE_PRIVATE);
		editor = prefs.edit();

		cbAutoSMS = (CheckBox) findViewById(R.id.cbAutoSMS);
		etAutoMsg = (EditText) findViewById(R.id.etAutoMsg);
		tvLeave = (TextView) findViewById(R.id.tvLeave);
		btnSave = (Button) findViewById(R.id.btnSetSave);
		btnCancel = (Button) findViewById(R.id.btnSetCancel);

		userPhone = prefs.getString("Phone", "");
		autoSMS = prefs.getBoolean("autoSMS", false);
		autoMsg = prefs.getString("autoMsg", "");

		cbAutoSMS.setChecked(autoSMS);
		etAutoMsg.setText(autoMsg);

		btnCancel.setOnClickListener(btnCancelOnClickListener);
		btnSave.setOnClickListener(btnSaveOnClickListener);
		tvLeave.setOnClickListener(tvLeaveOnClickListener);

		servIntent = new Intent(SettingsActivity.this,
ConnectionService.class);
		CheckIfServiceIsRunning();
	}
```

```
	private OnClickListener tvLeaveOnClickListener = new
OnClickListener() {
		public void onClick(View v) {
			msgSendUseDel = new MessageCustom(userPhone,
"deleteUser", msgTime.getTime());
			sendMessageToService(msgSendUseDel);
			stopService(servIntent);
			doUnbindService();
			editor.putBoolean("isRegistered", false);
			editor.commit();
			finish();
			MainActivity.main_new.finish();
		}
	};
	private OnClickListener btnCancelOnClickListener = new
OnClickListener() {
		public void onClick(View v) {
			doUnbindService();
			finish();
		}
	};
	private OnClickListener btnSaveOnClickListener = new
OnClickListener() {
		public void onClick(View v) {
			if (cbAutoSMS.isChecked()) {
				editor.putBoolean("autoSMS", true);
			} else if (!cbAutoSMS.isChecked()) {
				editor.putBoolean("autoSMS", false);
			}
			editor.putString("autoMsg",
etAutoMsg.getText().toString());
			editor.commit();
			doUnbindService();
			finish();
		}
	};

	@Override
	protected void onDestroy() {
		super.onDestroy();
	}

	private void CheckIfServiceIsRunning() {
		// If the service is running when the activity starts, we want
to
		// automatically bind to it.
		Log.i(TAG, "CheckIfServiceIsRunning here: " +
ConnectionService.isRunning());
		if (ConnectionService.isRunning()) {
			doBindService();
		} else if (!ConnectionService.isRunning()) {
			startService(servIntent);
			doBindService();
		}
	}

	void doBindService() {
		Log.i(TAG, "doBindService now");
		bindService(new Intent(this, ConnectionService.class),
mConnection, Context.BIND_AUTO_CREATE);
		mIsBound = true;
```

```
	}

	void doUnbindService() {
		if (mIsBound) {
			// If we have received the service, and hence registered
with it,
			// then now is the time to unregister.
			if (mService != null) {
				try {
					Message msg = Message.obtain(null,
ConnectionService.MSG_UNREGISTER_CLIENT);
					msg.replyTo = mMessenger;
					mService.send(msg);
				} catch (RemoteException e) {
					// There is nothing special we need to do if
the service has
					// crashed.
				}
			}
			// Detach our existing connection.
			unbindService(mConnection);
			mIsBound = false;
		}
	}

	private ServiceConnection mConnection = new ServiceConnection() {
		public void onServiceConnected(ComponentName className, IBinder
service) {
			mService = new Messenger(service);
			try {
				Message msg = Message.obtain(null,
ConnectionService.MSG_REGISTER_CLIENT);
				msg.replyTo = mMessenger;
				mService.send(msg);
			} catch (RemoteException e) {
				// In this case the service has crashed before we
could even do
				// anything
				// with it
			}
		}

		public void onServiceDisconnected(ComponentName className) {
			// This is called when the connection with the service
has been
			// unexpectedly disconnected - process crashed.
			mService = null;
		}
	};

	private void sendMessageToService(MessageCustom _msg) {
		// Log.i(TAG, "sendMessageToService here");
		// Log.v(TAG, "mIsBound: " + mIsBound);
		if (mIsBound) {
			if (mService != null) {
				try {
					Message msg = new Message();
					// Log.e(TAG, "sendMessageToService msg: " +
_msg);
					msg.obj = _msg;
```

```
                                msg.what =
ConnectionService.MSG_SET_INT_VALUE;
                                msg.replyTo = mMessenger;
                                mService.send(msg);
                          } catch (RemoteException e) {
                          }
                    }
              }
        }

}
```

ConnectionService

```
public class ConnectionService extends Service {
      public static final String TAG = "ConnectionService";
      static final int MSG_REGISTER_CLIENT = 1;
      static final int MSG_UNREGISTER_CLIENT = 2;
      static final int MSG_SET_INT_VALUE = 3;
      static final int MSG_SET_STRING_VALUE = 4;
      public static final String SERVERIP = "192.168.2.7";
      public static final int SERVERPORT = 4444;

      private NotificationManager nm;
      private static boolean isRunning = false;
      // Keeps track of all current registered clients.
      ArrayList<Messenger> mClients = new ArrayList<Messenger>();
      int mValue = 0; // Holds last value set by a client.
      // Target we publish for clients to send messages to IncomingHandler.
      final Messenger mMessenger = new Messenger(new IncomingHandler());
      public boolean reading;
      public InetAddress serverAddr;
      public Socket socket;
      public ObjectOutputStream out;
      public ObjectInputStream in;

      @Override
      public IBinder onBind(Intent intent) {
            return mMessenger.getBinder();
      }

      // Handler of incoming messages from clients.
      @SuppressLint("HandlerLeak")
      class IncomingHandler extends Handler {
            @Override
            public void handleMessage(Message msg) {
                  switch (msg.what) {
                  case MSG_REGISTER_CLIENT:
                        mClients.add(msg.replyTo);
                        break;
                  case MSG_UNREGISTER_CLIENT:
                        mClients.remove(msg.replyTo);
                        break;
                  case MSG_SET_INT_VALUE:
                        try {
                              send((MessageCustom) msg.obj);
                        } catch (Exception e) {
                              Log.e(TAG, "IncomingHandler send error: " +
e);
                        }
                        break;
```

```
                default:
                        super.handleMessage(msg);
                }
        }
    }

    private void sendMessageToUI(Object objIn) {
        for (int i = mClients.size() - 1; i >= 0; i--) {
                try {
                        Message msg = new Message();
                        msg.obj = objIn;
                        mClients.get(i).send(msg);

                } catch (RemoteException e) {
                        // The client is dead. Remove it from the list; we
are going
                        // through the list from back to front so this is
safe to do
                        // inside the loop.
                        mClients.remove(i);
                }
        }
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.i("ConnectionService", "Service Started.");

        Runnable connect = new connectSocket();
        new Thread(connect).start();
        showNotification();
        isRunning = true;
    }

    private void showNotification() {

        PendingIntent contentIntent = PendingIntent.getActivity(this,
0, new Intent(this, MainActivity.class), 0);
        NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(this)//
                        .setSmallIcon(R.drawable.ic_launcher)//

    .setContentTitle(getString(R.string.service_label))//
                        .setContentIntent(contentIntent)//
                        .setOngoing(true)//
                        .setContentText(getText(R.string.service_running));
        nm = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
        nm.notify(R.string.service_running, mBuilder.build());
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.i("ConnectionService", "Received start id " + startId + ":
" + intent);
        return START_STICKY; // run until explicitly stopped.
    }

    public static boolean isRunning() {
        return isRunning;
```

```
	}

	@Override
	public void onDestroy() {
		super.onDestroy();
		nm.cancel(R.string.service_running); // Cancel the persistent
                                                                        //
notification.
		Log.i("ConnectionService", "Service Stopped.");
		isRunning = false;
	}

	public void send(MessageCustom msg) {
		try {
			// Log.v(TAG, "send method msg:" + msg.toString());
			// Log.e(TAG,"socket status: "+ socket.isClosed());
			if (out != null) {
				out.writeObject(msg);
				out.flush();
			}
		} catch (Exception e) {
			Log.e(TAG, "Send method error: " +
e.getLocalizedMessage());
			e.printStackTrace();
		}
	}

	class connectSocket implements Runnable {
		@Override
		public void run() {
			reading = true;

				try {
					serverAddr = InetAddress.getByName(SERVERIP);
					Log.e("TCP Client", "C: Connecting...");

					socket = new Socket(serverAddr, SERVERPORT);
					try {
						out = new
ObjectOutputStream(socket.getOutputStream());
						in = new
ObjectInputStream(socket.getInputStream());
						Log.i(TAG, "C: Connected.");
						while (reading) {
							Object objIn = in.readObject();
							// Log.i(TAG, "Object read: " +
objIn);

							sendMessageToUI(objIn);
						}
					} catch (Exception e) {
						Log.e("TCP", "S: Error", e);
					}
				} catch (Exception e) {
					Log.e("TCP", "C: Error", e);
				}
		}

	}
}
```

IncomingMsgHandler

```
public class IncomingMsgHandler {

	private ContantsClass getConst = new ContantsClass();

	private ArrayList<User> listUsers, listMain;
	private NotificationCustom notification;
	private ListSort sortedList = new ListSort();

	@SuppressWarnings({ "static-access", "unchecked" })
	public Object msgHandle(Object objIn, int handleConst) {
		if (objIn instanceof Boolean) {
			boolean boolResult = (Boolean) objIn;
			return boolResult;
		} else if (objIn instanceof ArrayList<?>) {
			if (handleConst == getConst.HANDLE_USERS) {
				listMain = (ArrayList<User>) objIn;
				listUsers = sortedList.getSortedListDB(listMain);
				return listUsers;
			}
		} else if (objIn instanceof NotificationCustom) {
			if (handleConst == getConst.HANDLE_MESSAGE) {
				notification = (NotificationCustom) objIn;
				return notification;
			}
		}
		return null;

	}

}
```

ContactAdapter

```
public class ContactAdapter extends BaseAdapter {
	public static final String TAG = "ContactAdapter";

	private final ArrayList<ContactEntry> listContacts;
	private LayoutInflater mLayoutInflater;
	private Context context;
	private ImageView iv;
    private static LruCache<Integer, Bitmap> cache;

	public ContactAdapter(ArrayList<ContactEntry> listContacts, Context
context) {
		this.listContacts = listContacts;
		this.context = context;
		mLayoutInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);


		final int maxMem = (int) Runtime.getRuntime().maxMemory() /
1024;
		final int cacheSize = maxMem / 8;

		cache = new LruCache<Integer, Bitmap>(cacheSize);
	}

	@Override
	public int getCount() {
```

```
            return listContacts.size();
      }

      @Override
      public ContactEntry getItem(int position) {
            return listContacts.get(position);
      }

      @Override
      public long getItemId(int position) {
            return position;
      }

      @TargetApi(Build.VERSION_CODES.ICE_CREAM_SANDWICH)
      @SuppressLint({ "InlinedApi", "NewApi" })
      @Override
      public View getView(int position, View view, ViewGroup parent) {
            if (view == null) {
                  view = mLayoutInflater.inflate(R.layout.grid_item_icon,
null);
            }
            final TextView name = (TextView)
view.findViewById(R.id.tvUser);
            final ContactEntry current = listContacts.get(position);
            int isUser = current.getIsUser();
            int online = current.getIsOnline();
            String ContactName = current.getContactName();
            CheckBox cb = (CheckBox) view.findViewById(R.id.cbIsUser);
            iv = (ImageView) view.findViewById(R.id.ivContact);

            Uri contactUri =
ContentUris.withAppendedId(Contacts.CONTENT_URI,
Long.valueOf(current.getContactId()));
            iv.setImageResource(R.drawable.afro);

            if (getBitmapFromCache(current.getContactId()) != null) {

      iv.setImageBitmap(getBitmapFromCache(current.getContactId()));
            } else {
                  ImageLoader loader = new ImageLoader(iv, contactUri,
current.getContactId());
                  loader.execute();
            }
            if (ContactName != null) {
                  name.setText(ContactName);
            }

            if (online == 1) {
                  if (isUser == 1) {
                        name.setBackground(null);

      name.setBackground(view.getResources().getDrawable(R.drawable.back_on
));
                  } else {
                        name.setBackground(null);
                  }
            } else if (online == 0) {
                  if (isUser == 1) {
                        name.setBackground(null);
```

```
	name.setBackground(view.getResources().getDrawable(R.drawable.back_of
f));
				} else {
					name.setBackground(null);
				}
			}

			if (isUser == 1) {
				cb.setVisibility(0);// visible
				cb.setChecked(true);
			} else if (isUser == 0) {
				cb.setVisibility(8);// invisible
				cb.setChecked(false);
			}

			return view;
		}



		class ImageLoader extends AsyncTask<Void, Void, Bitmap> {
			private ImageView mView;
			private Uri mUri;
			private int position;

			public ImageLoader(ImageView view, Uri uri, int position) {

				if (view == null) {
					throw new IllegalArgumentException("View Cannot be
null");
				}
				if (uri == null) {
					throw new IllegalArgumentException("Uri cant be
null");
				}
				mView = view;
				this.position = position;
				mUri = uri;
			}

			protected Bitmap doInBackground(Void... args) {
				Bitmap bitmap;

				// Load image from the Content Provider
				InputStream in =
ContactsContract.Contacts.openContactPhotoInputStream(context.getContentRes
olver(), mUri);
				bitmap = BitmapFactory.decodeStream(in);

				return bitmap;
			}

			protected void onPostExecute(Bitmap bitmap) {
				// If is in somewhere else, do not temper

				// If no image was there and do not put it to cache
				if (bitmap != null) {

					mView.setImageBitmap(bitmap);
					addBitmapToCache(position, bitmap);
```

```
                        return;
                  }
                  // Otherwise, welcome to cache
                  return;
            }
      }

      /** Add image to cache */
      private void addBitmapToCache(int key, Bitmap bitmap) {
            if (getBitmapFromCache(key) == null) {
                  cache.put(key, bitmap);
            }
      }

      /** Retrive image from cache */
      private Bitmap getBitmapFromCache(int i) {
            return cache.get(i);
      }

}
```

NotificationReply

```
public class NotificationReply extends Activity {
      private static final String TAG = "NotificationReply";

      private MessageCustom msgSendNotification;
      String UserName, recipient, notifMessage, userPhone;
      Messenger mService = null;
      private boolean mIsBound;
      private Intent servIntent;
      private CurrentTime msgTime = new CurrentTime();
      public SharedPreferences prefs;
      public SharedPreferences.Editor editor;

      final Messenger mMessenger = new Messenger(new IncomingHandler());

      @SuppressLint("HandlerLeak")
      class IncomingHandler extends Handler {
            @Override
            public void handleMessage(Message msg) {
                  super.handleMessage(msg);
            }
      }

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            requestWindowFeature(Window.FEATURE_NO_TITLE);
            super.onCreate(savedInstanceState);
            setContentView(R.layout.notification_reply);

            prefs = getSharedPreferences("NotifyMe", MODE_PRIVATE);
            editor = prefs.edit();
            userPhone = prefs.getString("Phone", "");
            Log.w(TAG, "Phone is: " + userPhone);
            recipient = prefs.getString("recipient", "");
            Log.e(TAG,"recipient phone: "+recipient);
            notifMessage = prefs.getString("notifMessage", "");
            editor.putString("recipient", recipient);
            editor.commit();

            Log.w(TAG, "start service now");
```

```
            servIntent = new Intent(NotificationReply.this,
ConnectionService.class);
            startService(servIntent);

            Log.w(TAG, "CheckIfServiceIsRunning now");
            CheckIfServiceIsRunning();

            createPopup(recipient);
      }

      public void createPopup(final String recipient) {
            AlertDialog.Builder alert = new AlertDialog.Builder(this);

            alert.setTitle(notifMessage);
            alert.setMessage("Reply ");

            // Set an EditText view to get user input
            final EditText input = new EditText(this);
            alert.setView(input);

            alert.setPositiveButton("Ok", new
DialogInterface.OnClickListener() {

                  public void onClick(DialogInterface dialog, int
whichButton) {
                        Editable value = input.getText();
                        msgSendNotification = new MessageCustom(userPhone,
recipient, "message", value.toString(), msgTime.getTime());
                        Log.e(TAG,"recipient phone: "+recipient);
                        Log.e(TAG,"msg to send:
"+msgSendNotification.toString());
                        sendMessageToService(msgSendNotification);
                  }
            });

            alert.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
                  public void onClick(DialogInterface dialog, int
whichButton) {
                        finish();
                  }
            });

            alert.show();
      }

      private void CheckIfServiceIsRunning() {
            // If the service is running when the activity starts, we want
to
            // automatically bind to it.
            Log.i(TAG, "CheckIfServiceIsRunning here: " +
ConnectionService.isRunning());
            if (ConnectionService.isRunning()) {
                  doBindService();
            } else if (!ConnectionService.isRunning()) {
                  doBindService();
            }
      }

      void doBindService() {
            Log.i(TAG, "doBindService now");
```

```
            bindService(new Intent(this, ConnectionService.class),
mConnection, Context.BIND_AUTO_CREATE);
            mIsBound = true;
      }

      void doUnbindService() {
            if (mIsBound) {
                  // If we have received the service, and hence registered
with it,
                  // then now is the time to unregister.
                  if (mService != null) {
                        try {
                              Message msg = Message.obtain(null,
ConnectionService.MSG_UNREGISTER_CLIENT);
                              msg.replyTo = mMessenger;
                              mService.send(msg);
                        } catch (RemoteException e) {
                              // There is nothing special we need to do if
the service has
                              // crashed.
                        }
                  }
                  // Detach our existing connection.
                  unbindService(mConnection);
                  mIsBound = false;
            }
      }

      private ServiceConnection mConnection = new ServiceConnection() {
            public void onServiceConnected(ComponentName className, IBinder
service) {
                  mService = new Messenger(service);
                  try {
                        Message msg = Message.obtain(null,
ConnectionService.MSG_REGISTER_CLIENT);
                        msg.replyTo = mMessenger;
                        mService.send(msg);
                  } catch (RemoteException e) {
                        // In this case the service has crashed before we
could even do
                        // anything
                        // with it
                  }
            }

            public void onServiceDisconnected(ComponentName className) {
                  // This is called when the connection with the service
has been
                  // unexpectedly disconnected - process crashed.
                  mService = null;
            }
      };

      private void sendMessageToService(MessageCustom _msg) {
            // Log.i(TAG, "sendMessageToService here");
            // Log.v(TAG, "mIsBound: " + mIsBound);
            if (mIsBound) {
                  if (mService != null) {
                        try {
                              Message msg = new Message();
```

```
                                // Log.e(TAG, "sendMessageToService msg: " +
_msg);
                                msg.obj = _msg;
                                msg.what =
ConnectionService.MSG_SET_INT_VALUE;
                                msg.replyTo = mMessenger;
                                mService.send(msg);
                          } catch (RemoteException e) {
                          }
                          finish();
                    }
              }
        }

        @Override
        protected void onDestroy() {
              super.onDestroy();
              try {
                    stopService(servIntent);
                    doUnbindService();
                    finish();
              } catch (Throwable t) {
                    Log.e(TAG, "Failed to unbind from the service", t);
              }
        }

}
```

Utility Classes

ContactEmtry

```
public class ContactEntry implements Parcelable {

    private int contactId;
    private String contactName;
      private String contactPhone;
      private int isUser=0, isOnline=0;

    public ContactEntry(int contactId, String contactName,String
contactPhone,int isUser,int isOnline) {
            this.contactId = contactId;
            this.contactName = contactName;
            this.contactPhone = contactPhone;
            this.isUser=isUser;
            this.isOnline=isOnline;
      }

    public void setIsUser(int isUser){
            this.isUser=isUser;
      }

    public void setIsOnline(int isOnline){
            this.isOnline=isOnline;
      }

      public int getIsOnline() {
            return isOnline;
      }


      public int getIsUser() {
            return isUser;
      }

      public int getContactId() {
            return contactId;
      }

      public String getContactName() {
            return contactName;
      }

      public String getContactPhone() {
            return contactPhone;
      }

      // The following methods that are required for using Parcelable
    private ContactEntry(Parcel in) {
        // This order must match the order in writeToParcel()
      contactId= in.readInt();
      contactName = in.readString();
      contactPhone= in.readString();
        isUser = in.readInt();
        isOnline = in.readInt();
```

```
        // Continue doing this for the rest of your member data
    }

    public void writeToParcel(Parcel out, int flags) {
        // Again this order must match the User(Parcel) constructor
        out.writeInt(contactId);
        out.writeString(contactName);
        out.writeString(contactPhone);
        out.writeInt(isUser);
        out.writeInt(isOnline);
        // Again continue doing this for the rest of your member data
    }

    // Just cut and paste this for now
    public int describeContents() {
        return 0;
    }

    // Just cut and paste this for now
    public static final Parcelable.Creator<ContactEntry> CREATOR = new
Parcelable.Creator<ContactEntry>() {
        public ContactEntry createFromParcel(Parcel in) {
            return new ContactEntry(in);
        }

        public ContactEntry[] newArray(int size) {
            return new ContactEntry[size];
        }
    };
}
```

ConstantsClass

```
public final class ConstantsClass {
      public static final int HANDLE_USERS = 1;
      public static final int HANDLE_NOTIFICATIONS = 2;
      public static final int HANDLE_MESSAGE = 3;
      public static final int HANDLE_LOGIN = 4;
      public static final int ADD_USER = 5;
      public static final int DEL_USER = 6;

      public ConstantsClass(){

      }
}
```

CurrentTime

```
public class CurrentTime {

    Calendar cal=Calendar.getInstance();
    SimpleDateFormat dt = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    String timeStamp = dt.format(cal.getTime());

    public String getTime(){
        return timeStamp;
    }
}
```

FindInfo

```
public class FindInfo {
	private boolean isUser=false;
	private int searchINIndex;

	public FindInfo(){
	}

	public boolean listFindIfUser(ArrayList<ContactEntry> searchIN,String
searchFOR){
			for(int j=0;j<searchIN.size();j++){

	if(searchIN.get(j).getContactPhone().equals(searchFOR)){
					isUser=true;
					searchINIndex=j;
					return isUser;
				}
			}
		return isUser;
	}

	public int getIndex(){
		return searchINIndex;
	}

	public String findContactName(ArrayList<ContactEntry> searchIN,String
phone){
		for(int j=0;j<searchIN.size();j++){
			if(searchIN.get(j).getContactPhone().equals(phone)){
				return searchIN.get(j).getContactName();

			}
		}
		return "";
	}
}
```

ListSort

```
public class ListSort {

	private ArrayList<ContactEntry> listSort;
	private ArrayList<User> listSortDB;

	public ListSort(){
	}

//	sort by db user
	public ArrayList<User> getSortedListDB(ArrayList<User> _listSort){
		this.listSortDB=_listSort;

		Collections.sort(listSortDB, new Comparator<User>() {
			@Override
			public int compare(User p1, User p2) {
				return p2.getOnLine() - p1.getOnLine();
			}

		});

		return listSortDB;
```

```
	}


//	sort by ContactName
	public ArrayList<ContactEntry> getSortedList(ArrayList<ContactEntry>
cont){
		this.listSort=cont;

		Collections.sort(listSort, new Comparator<ContactEntry>() {
			@Override
			public int compare(ContactEntry p1, ContactEntry p2) {
				return
p1.getContactName().compareTo(p2.getContactName());
			}

		});

		return listSort;
	}
//	sort by isUser
	public ArrayList<ContactEntry>
getSortedListUsers(ArrayList<ContactEntry> cont){
		this.listSort=cont;

		Collections.sort(listSort, new Comparator<ContactEntry>() {
			@Override
			public int compare(ContactEntry p1, ContactEntry p2) {
				return p2.getIsUser()-p1.getIsUser();
			}

		});

		return listSort;
	}
//	sort by isOnline
	public ArrayList<ContactEntry>
getSortedListOnline(ArrayList<ContactEntry> cont){
		this.listSort=cont;

		Collections.sort(listSort, new Comparator<ContactEntry>() {
			@Override
			public int compare(ContactEntry p1, ContactEntry p2) {
				return p2.getIsOnline()-p1.getIsOnline();
			}

		});

		return listSort;
	}
}
```

MessageCustom

```
public class MessageCustom implements Serializable {

	private static final long serialVersionUID = 1L;
    public String sender, content, type, recipient, msgTime;
    public int flag;

    //message
    public MessageCustom(String sender, String recipient, String type,
String content, String msgTime) {
```

```
        this.sender = sender;
        this.recipient = recipient;
        this.type = type;
        this.content = content;
        this.msgTime = msgTime;
    }

    // getUsers/deleteUser
    public MessageCustom(String sender, String type, String msgTime) {
        this.sender = sender;
        this.type = type;
        this.msgTime = msgTime;
    }

    // update/login/addUser
    // 5-addUser
    // 6-login
    // 7-log out
    public MessageCustom(String sender, String type, String msgTime, int
flag) {
        this.sender = sender;
        this.type = type;
        this.msgTime = msgTime;
        this.flag = flag;
    }



    @Override
    public String toString() {
        return "{type='" + type + "', sender='" + sender + "', recipient='"
+ recipient + "', content='" + content + "', msgTime='" + msgTime + "',
flag='" + flag + "'}";
    }
}
```

NotificationCustom

```
public class NotificationCustom implements Serializable {

      private static final long serialVersionUID = 1L;
      private String sender, recipient, body, time;
      private int id;

      // db add
      public NotificationCustom(int id, String sender, String recipient,
String body, String time) {
            this.id = id;
            this.sender = sender;
            this.recipient = recipient;
            this.body = body;
            this.time = time;
      }

      // sending
      public NotificationCustom(String sender, String recipient, String
body, String time) {
            this.sender = sender;
            this.recipient = recipient;
            this.body = body;
            this.time = time;
      }
```

```
	// receiving
	public NotificationCustom(int id, String body, String time) {
		this.id = id;
		this.body = body;
		this.time = time;
	}

	public String getSender() {
		return sender;
	}

	public String geRecipient() {
		return recipient;
	}

	public String getBody() {
		return body;
	}

	public String getTime() {
		return time;
	}

	public int getID() {
		return id;
	}



	@Override
	public String toString() {
		return "{id='" + id + "', sender='" + sender + "', recipient='"
+ recipient + "', body='" + body + "', time='" + time + "'}";
	}
}
```

User

```
public class User implements Serializable,Parcelable{
	private static final long serialVersionUID = 1L;
    private String Phone;
    private int onLine;

    public User(int _onLine, String _phone) {
        this.onLine = _onLine;
        this.Phone = _phone;
    }


    public int getOnLine() {
        return onLine;
    }

    public String getPhone() {
        return Phone;
    }

    @Override
    public String toString() {
        return "{Phone='" + Phone + "', onLine='" + onLine + "'}";
    }

	// The following methods that are required for using Parcelable
```

```
    private User(Parcel in) {
        // This order must match the order in writeToParcel()
        onLine = in.readInt();
        Phone = in.readString();
        // Continue doing this for the rest of your member data
    }

    public void writeToParcel(Parcel out, int flags) {
        // Again this order must match the User(Parcel) constructor
        out.writeInt(onLine);
        out.writeString(Phone);
        // Again continue doing this for the rest of your member data
    }

    // Just cut and paste this for now
    public int describeContents() {
        return 0;
    }

    // Just cut and paste this for now
    public static final Parcelable.Creator<User> CREATOR = new
Parcelable.Creator<User>() {
        public User createFromParcel(Parcel in) {
            return new User(in);
        }

        public User[] newArray(int size) {
            return new User[size];
        }
    };
}
```

Client Layout Files

main

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <GridView
        android:id="@+id/gvMain"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:numColumns="auto_fit">"
    </GridView>

</LinearLayout>
```

grid_item_icon

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:weightSum="2" >

    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="130dp"
        android:layout_weight="1" >

        <ImageView
            android:id="@+id/ivContact"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_alignParentBottom="true"
            android:layout_alignParentLeft="true"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:contentDescription="@string/ivDescreption"
            android:scaleType="fitXY"
            android:src="@drawable/afro" />

        <CheckBox
            android:id="@+id/cbIsUser"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignBaseline="@+id/ivContact"
            android:layout_alignBottom="@+id/ivContact"
            android:layout_alignRight="@+id/ivContact"
            android:clickable="false"
            android:focusable="false" />
    </RelativeLayout>

    <LinearLayout
        android:layout_width="match_parent"
```

```
        android:layout_height="0dp"
        android:layout_weight="1" >

        <TextView
            android:id="@+id/tvUser"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:singleLine="true"
            android:text="@string/tvContact"
            android:textAppearance="?android:attr/textAppearanceMedium" />

    </LinearLayout>

</LinearLayout>
```

settings

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".SettingsActivity" >

    <CheckBox
        android:id="@+id/cbAutoSMS"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="@string/autoSMS"
        android:textSize="20sp" />

    <EditText
        android:id="@+id/etAutoMsg"
        android:layout_width="600dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:hint="@string/autoMsgHINT"
        android:ems="10" >

        <requestFocus />
    </EditText>

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/cbAutoSMS"
        android:layout_alignRight="@+id/etAutoMsg"
        android:layout_below="@+id/cbAutoSMS"
        android:layout_marginTop="37dp"
        android:text="@string/autoMsg"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textSize="20sp" />

    <Button
        android:id="@+id/btnSetCancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

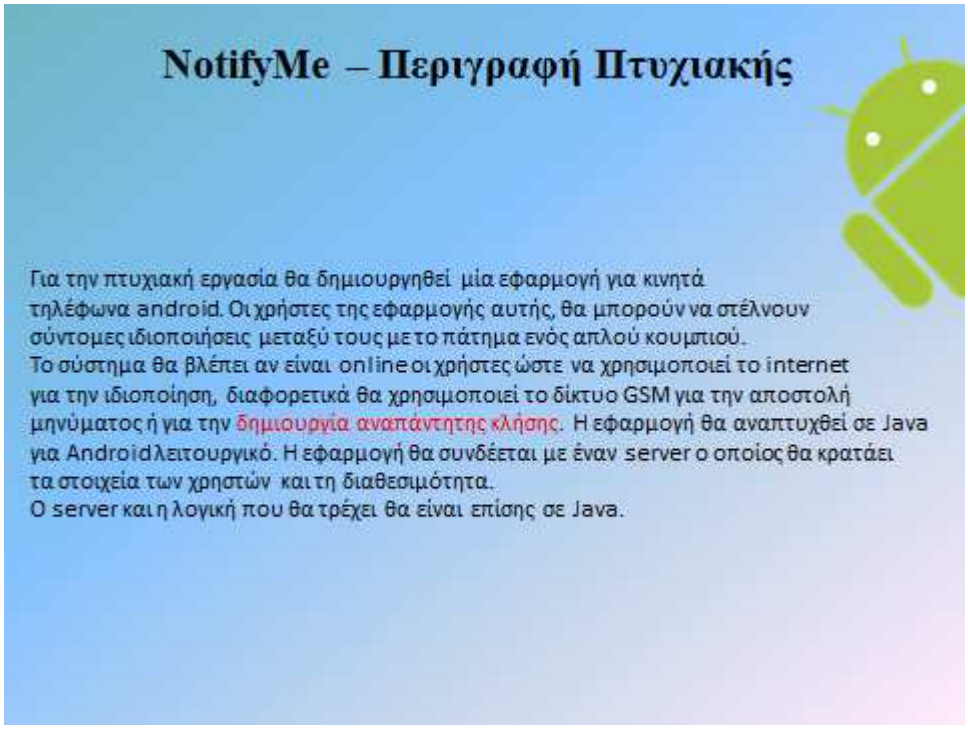
```
        android:layout_alignBaseline="@+id/btnSetSave"
        android:layout_alignBottom="@+id/btnSetSave"
        android:layout_alignRight="@+id/etAutoMsg"
        android:text="@string/btnCancel" />

    <Button
        android:id="@+id/btnSetSave"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/etAutoMsg"
        android:layout_below="@+id/tvLeave"
        android:layout_marginTop="38dp"
        android:text="@string/btnSave" />

    <TextView
        android:id="@+id/tvLeave"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/etAutoMsg"
        android:layout_marginTop="36dp"
        android:layout_toLeftOf="@+id/btnSetCancel"
        android:text="@string/leave"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

register

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".RegisterActivity" >

    <EditText
        android:id="@+id/txtRegPhone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="22dp"
        android:ems="10"
        android:hint="@string/phone"
        android:inputType="phone" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/btnSignUp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/txtRegPhone"
        android:layout_below="@+id/txtRegPhone"
        android:layout_marginTop="39dp"
        android:text="@string/signup" />

    <Button
        android:id="@+id/btnCancel"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/btnSignUp"
        android:layout_alignBottom="@+id/btnSignUp"
        android:layout_alignRight="@+id/txtRegPhone"
        android:text="@string/cancel" />

</RelativeLayout>
```

Παράρτημα Β :Παρουσίαση Πτυχιακής (Διαφάνειες)



Android

- Τι είναι το Android

Android Inc.

- Ιδρύθηκε στην Καλιφόρνια τον Οκτώβριο του 2003 από τους: Andy Rubin, Rich Miner, Nick Sears και Chris White
- Αρχικός στόχος: λειτουργικό σύστημα για ψηφιακές κάμερες

- 17 Αυγούστου 2005 εξαγορά από την Google
- Ανάπτυξη μίας πλατφόρμας βασισμένη στον Linux kernel

Open Handset Alliance

- 6 Νοεμβρίου 2007, ίδρυση
- Μέλη: κατασκευαστές συσκευών,
  πάροχοι τηλεπικοινωνιών,
  κατασκευαστές chipset
- Σκοπός: ανάπτυξη ανοιχτών προτύπων για φορητές συσκευές

- Μέλη της OHA δεν επιτρέπεται να κατασκευάσουν τηλέφωνα που να τρέχουν «μη συμβατές εκδόσεις» Android

Ιστορικό εκδόσεων Android

- Android 1.0 (API level 1)
- Android 1.1 (API level 2)
- Android 1.5 Cupcake (API level 3)
- Android 1.6 Donut (API level 4)
- Android 2.0 Eclair (API level 5)
- Android 2.0.1 Eclair (API level 6)
- Android 2.1 Eclair (API level 7)
- Android 2.2–2.2.3 Froyo (API level 8)
- Android 2.3–2.3.2 Gingerbread (API level 9)
- Android 2.3.3–2.3.7 Gingerbread (API level 10)
- Android 3.0 Honeycomb (API level 11)
- Android 3.1 Honeycomb (API level 12)
- Android 3.2 Honeycomb (API level 13)
- Android 4.0–4.0.2 Ice Cream Sandwich (API level 14)
- Android 4.0.3–4.0.4 Ice Cream Sandwich (API level 15)
- Android 4.1 Jelly Bean (API level 16)
- Android 4.2 Jelly Bean (API level 17)
- Android 4.3 Jelly Bean (API level 18)
- Android 4.4 KitKat (API level 19)

NotifyMe

- Τι είναι η εφαρμογή NotifyMe

- Τρόπος υλοποίησης
  - Server
  - Client (Android εφαρμογή)

NotifyMe – Απαιτήσεις Server

- Επικοινωνία με την βάση, ανάκτηση / καταγραφή δεδομένων
- Αποδοχή αιτημάτων από την εφαρμογή και απάντηση με κατάλληλα δεδομένα
- Μεταφορά μηνυμάτων από έναν χρήστη σε άλλο
- Διαχείριση πολλαπλών συνδέσεων

NotifyMe – Απαιτήσεις Χρήστη

- Εγγραφή στην εφαρμογή με τον αριθμό τηλεφώνου του
- Προβολή των επαφών του, διαχωρισμός χρηστών/μη χρηστών και συνδεδεμένων και μη
- Αποστολή/λήψη σύντομων μηνύματος σε/από άλλο χρήστη της εφαρμογής
- Επιλογή αποστολής προεπιλεγμένου μηνύματος με την απλή επιλογή επαφής
- Δυνατότητα αποχώρησης από την εφαρμογή

NotifyMe Παρουσίαση Εφαρμογής

NotifyMe


ΗΡΘΕ ΤΟ
ΤΕΛΟΣ