

פרויקט גמר בסייבר

Computer Resources Monitoring

**שם התלמיד:**

**ת"ז:**

**מנחים: מיכאל צ'רנובילסקי וגד רוזנטל**

**בית הספר: אורט גוטמן נתניה**

**כיתה: י"ב 7**

**מאי 2015**

**תוכן עניינים**

[תקציר כולל ורציונל הפרויקט 3](#_Toc451518445)

[מבוא ורקע כללי 4](#_Toc451518446)

[תיאור המוצר המוגמר 5](#_Toc451518447)

[שפת התכנות וסביבת העבודה 6](#_Toc451518448)

[ניסוח, ניתוח ופתרון הבעיה האלגוריתמית 8](#_Toc451518450)

[תיאור מבנה המוצר 17](#_Toc451518451)

[מדריך התקנה למשתמש 26](#_Toc451518452)

[תיאור הממשק למשתמש 27](#_Toc451518453)

[מבט אישי על העבודה ותהליך הפיתוח 29](#_Toc451518454)

[ביבליוגרפיה 31](#_Toc451518455)

[קוד התוכנה 32](#_Toc451518456)

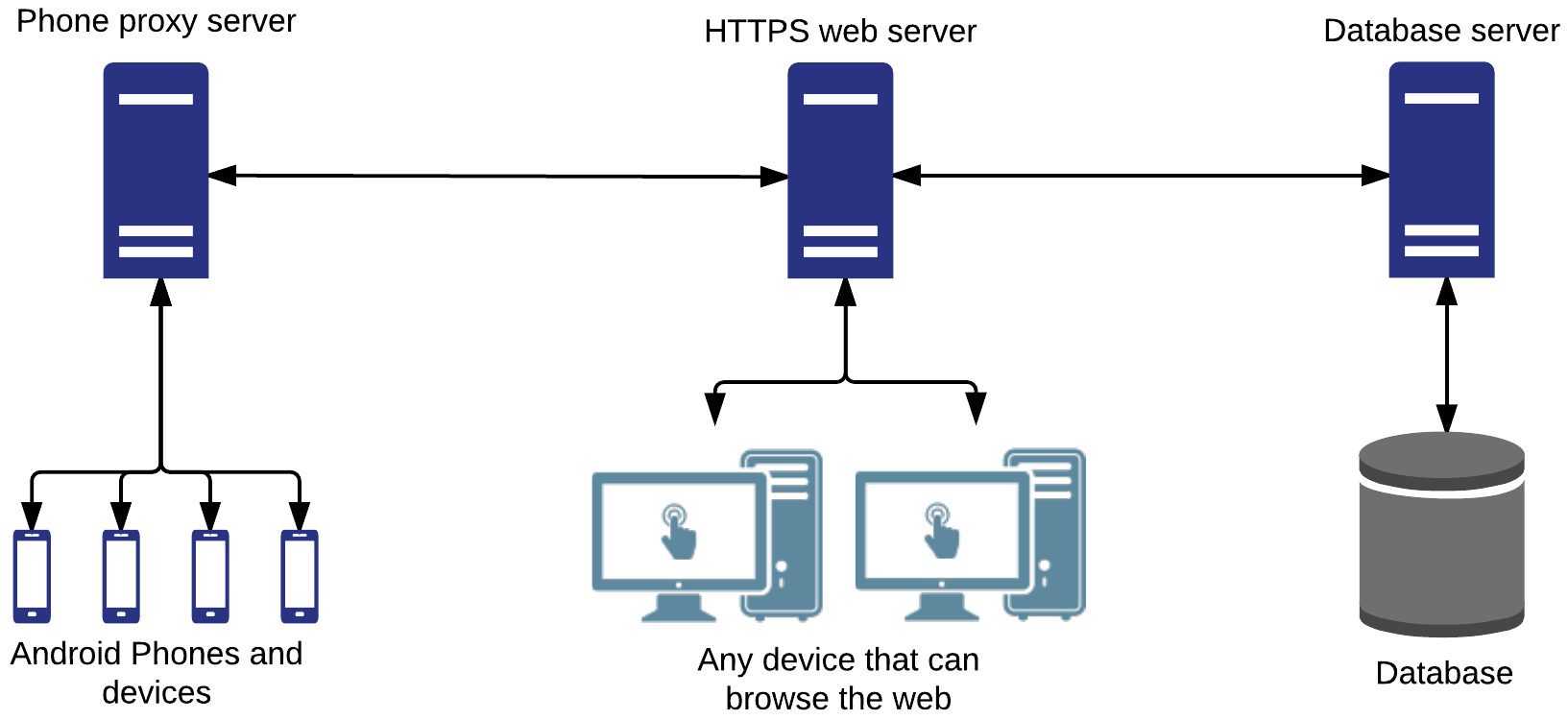
# תקציר כולל ורציונל הפרויקט

בימינו אלה להאנשים יש שתי עמדות עבודה; העמדה הניידת שהיא הטלפון והעמדה הנייחת שהיא המחשב הנייח. האנשים משתמשים במחשב לצורך כתיבה וניהול מסמכים ומעוניינים לנהל את אותו מידע גם מהטלפון. ניהול המידע גם מהטלפון מצריך יכולת ניהול קבצים וסינכרון בין המחשב לבין הטלפון.

הפתרון שלנו לבעיה זו הוא ה"RDroid"; שירות שנותן ללקוח להעביר מידע מההמחשב לטלפון וגם מהטלפון למחשב.

השירות יינתן בעזרת אפליקציית אנדריוד בטלפון אשר תאפשר העברת מידע בין המשתמשים לבין הטלפון הפרטי שלהם. המידע יועבר על ידי כך שהמשתמשים יתממשקו לאתר אינטרנט אשר הוא יתקשר לאפליקציה המותקנת בטלפון שלהם.

הסיבה שבחרתי לעבוד בפרויקט על ניהול טלפונים מרחוק היא שלעיתים קרובות מצאתי את עצמי במצבים בהם לא הייתה לי דרך לגשת למידע חשוב שהיה זמין בטלפון בשעה שעבדתי על המחשב, ולפעמים לא היה לי המידע בטלפון כאשר המידע היה על המחשב.

מבנה התקשורת של השירות בנוי על העברת קבצים בין המחשב לטלפון באמצעות תקשורת שנעשית דרך שרת Proxy ושרת המשמש כאתר אינטרנט.

# מבוא ורקע כללי

הפרויקט שלי מבוסס על תקשורת בין שרתים ומכשירים. השרת הראשי מנהל את התקשורת עם הלקוחות שרוצים לגשת לטלפון שלהם ואת העברתה לשרת הפרוקסי שמנהל את התקשורת עם הטלפונים.

הנושא המרכזי של הפרויקט הוא ניהול תקשורת עם לקוחות ועבודה עם המערכת ההפעלה אנדרואיד. פיתוח אפליקציותל לאנדרואיד שונה מאוד מפיתוח תוכנות בווינדוס ודורש פיתוח בצורה שונה. לדוגמא: באנדרואיד App אינו מסוגל לרוץ אם הוא לא מופיע על המסך. לכן, צריך ליצור Service שיפעל ברקע ושה-App יתקשר איתו.

הפרויקט שלי עוסק בהעברת פקודות ומידע בין מכשירים שונים בשביל ניהול הטלפון מרחוק דרך האינטרנט.

# תיאור המוצר המוגמר

שם המוצר: Rdroid.

המוצר אמור לבצע העברת מידע מרחוק בין המחשב לטלפון דרך האינטרנט לפי המשתמש.

המוצר עושה זאת בעזרת שרתים המספקים ממשק למשתמש לניהול ומעבירים את המידע ממכשיר למכשיר. השרתים מזהים את הלקוחות בעזרת מייל וסיסמא וכך מוודאים שהטלפון והלקוח שמתחבר לאתר הם הזוג הנכון (שאנשים לא יוכלו לגשת לטלפונים שלא שלהם).

השאלות שהמערכת עונה עליהן:

1. **כיצד להגן על המידע הרגיש מפני גורמים לא מורשים?** התקשורת בין הטלפונים לשרת מוצפנת, כך שגורם לא מורשה לא יוכל להפיק שום תועלת מהאזנה למידע הנשלח, הוא לא יוכל לגלות שום מידע הנמצא על הטלפון ומה הפקודות שהלקוח מבקש ממנו לעשות.
2. **כיצד לעבוד בו-זמנית עם כל הטלפונים השונים?** כאשר טלפונים מתחברים לשרת, הוא פותח Thread נפרד לכל אחד ומסוגל לתקשר במקביל עם כולם, לקבל מהם מידע ולשלוח להם פקודות. כל טלפון ולקוח מתחבר עם שם משתמש וסיסמא כך שלא ניתן לגשת לטלפונים של אחרים.

# שפת התכנות וסביבת העבודה

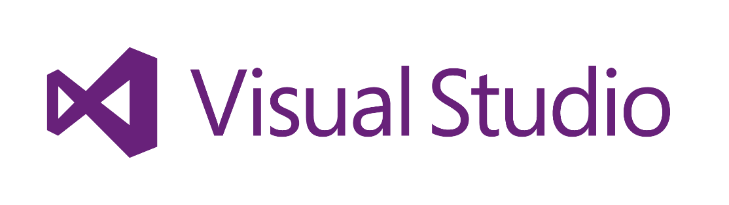
הפרויקט נכתב ב- Python, C# ו-Java.

* שרת הפרוקסי נכתב ב- Python בשל הפשטות והנוחות של השפה.
* השרת הראשי נכתב ב- C# כי התלמיד שעבד איתי למד את סביבת .Net לעבודה בשירותי רשת.
* האפליקציה נכתבה ב-Java מכיוון שגוגל תומכים בשפה זו לסביבת אנדריוד.

סביבת העבודה עם Python הייתה PyCharm שהיא תוכנה מקצועית ונוחה לעבודה על פרויקטים מרובי מודולים וקבצים. היא שימשה גם לפיתוח הקוד ועבודה במקביל עם כל המודולים השונים וגם להרצתו בשביל בדיקות.

סביבת העבודה עם C# הייתה Microsoft Visual Studio 2015 ששימשה לכתיבת השרת באמצעות סביבת .Net, וה- debugger המובנה שימש לבדיקות. אני כתבתי מספר מחלקות שנועדו לפשט את התקשורת אל שרת הפרוקסי אך רוב העבודה בשפה זאת נעשתה ע"י התלמיד שעבד איתי, יובל ינושפולסקי.

סביבת העבודה על Java היא Android Studio שהיא סביבת העבודה של גוגל לפיתוח אפליקציות לאנדרואיד. הסביבה ממש נוחה ותומכת במגוון פעולות העוזרות לפיתוח האפליקציה. בסביבת העבודה יש גם Debugger ו-Emulator למכשירי אנדרואיד.



# http://famouslogos.net/images/android-logo.jpghttp://www.eliteisinc.com/wp-content/uploads/2015/11/android-studio.pnghttp://www.logospike.com/wp-content/uploads/2014/11/Java_logo-2.jpg

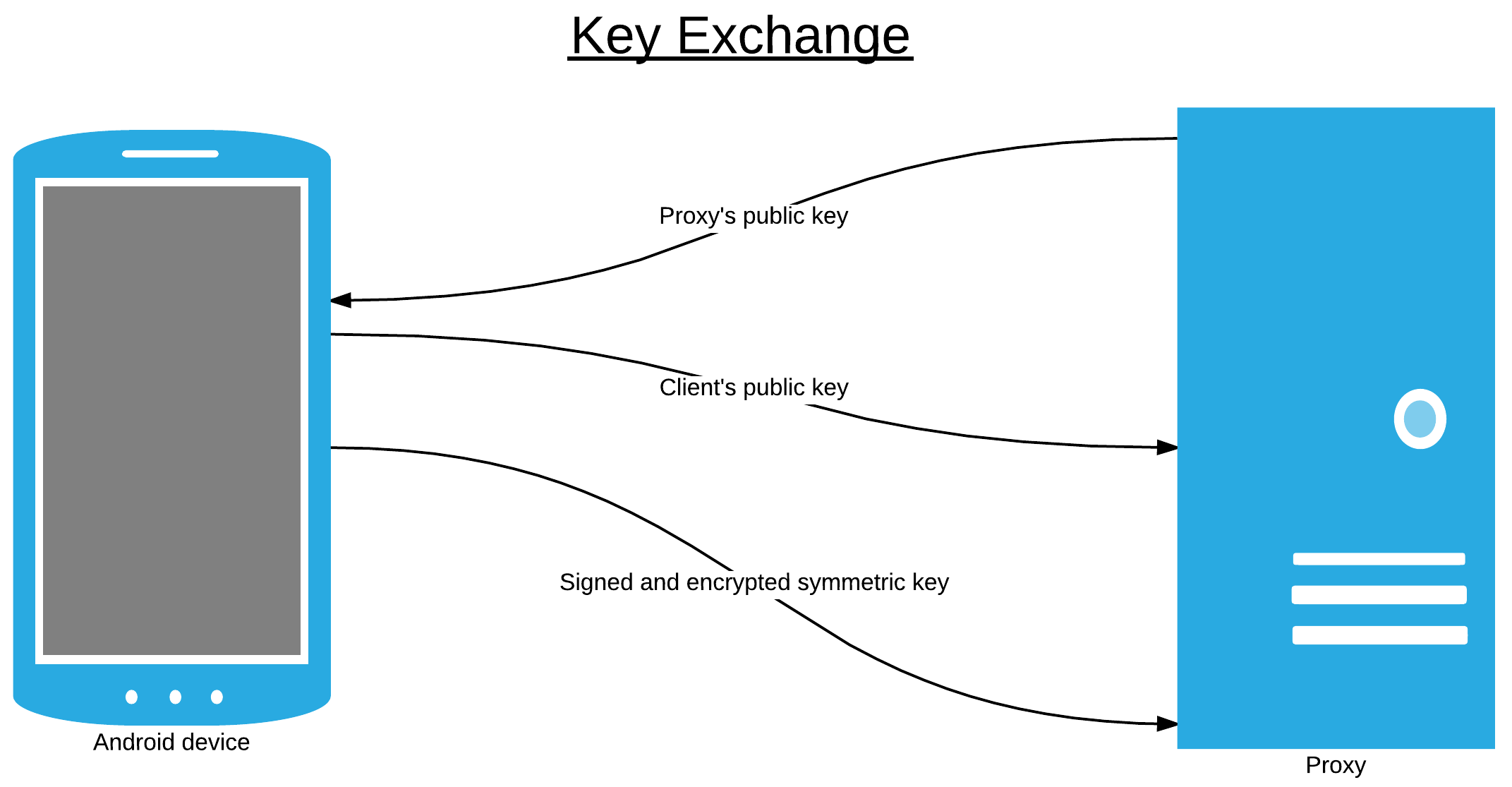
# ניסוח, ניתוח ופתרון הבעיה האלגוריתמית

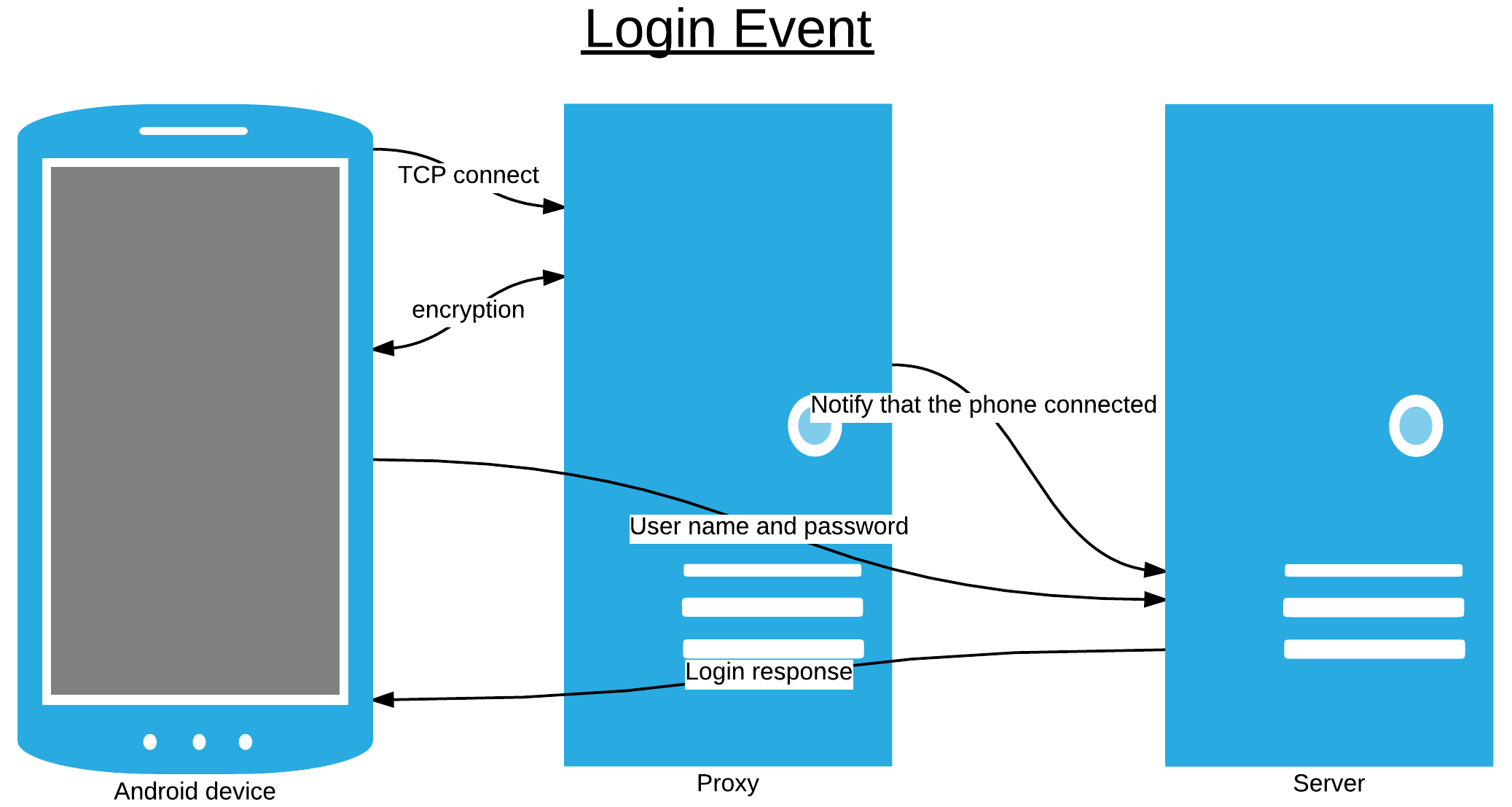
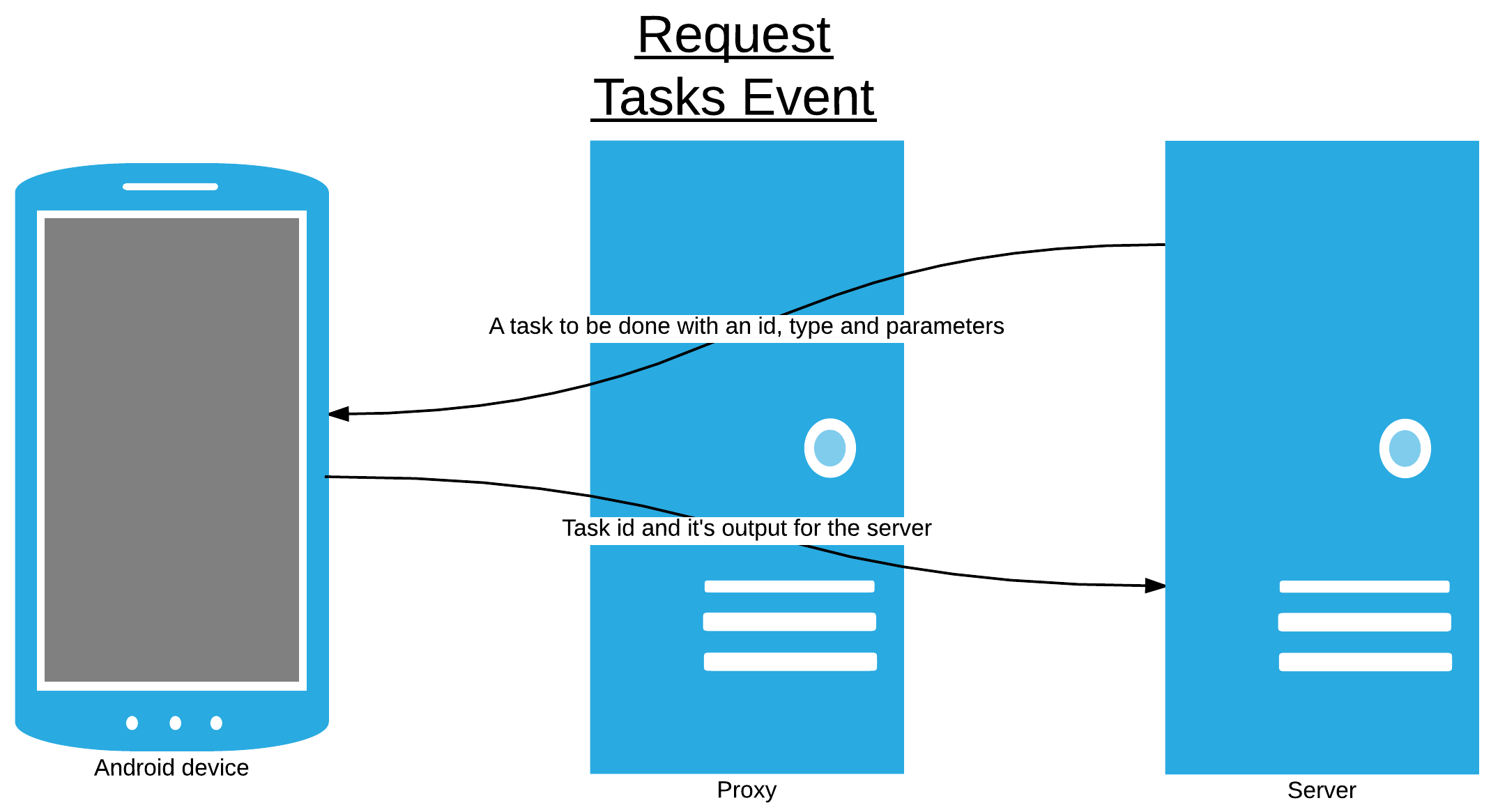
במהלך פיתוח המוצר נתקלתי בבעיות אלגוריתמיות רבות:

* **מימוש שרת מולטי-קליינט:**

הבעיה הראשונה שעמדה בפניי בפיתוח מוצר זה היא יצירת שרת מולטי-קליינט. כלומר, הייתי צריך לממש שרת שמתחברים אליו מספר לקוחות בו-זמנית והוא מסוגל לדבר עם כולם. בתוך בעיה אלגוריתמית זו ישנן מספר תתי-בעיות:

* **פיתוח פרוטוקול משלי:** הפרוטוקול שפיתחתי צריך לנהל תקשורת בין הטלפונים לבין הפרוקסי ובה בעת לנהל את התקשורת בין הפרוקסי לבין שרת האינטרנט. במקביל הפרוטוקול צריך לתמוך בהצפנה ובתמיכה בכמה לקוחות במקביל.
* **מימוש אבטחת מידע:** כדי למנוע מהמידע לעבור בין השרת ללקוחות באופן חשוף הייתי צריך לממש אבטחת מידע, כלומר להוסיף לפרוטוקול שלי להצפנה. הבעיה האלגוריתמית היא שליחה של מידע מוצפן בפרוטוקול ופענוח שלו בצד השני של הקו.
* **העברת מידע בין השרתים:** מכיוון שהטלפונים לא מתחברים ישירות לשרת, היה לי צורך להוסיף לפרוטוקול חלק נוסף שנועד לניווט ההודעות בין השרת לטלפונים דרך הפרוקסי והתראות על מצבי החיבור איתם.

הנה פירוט הפרוטוקול:



המבנה של הפקטות:

כל הפקטות של הלקוח מתחילות ב-

Rdroid CLIENT

כל הפקטות של שרת הפרוקסי מתחילות ב-

Rdroid PROXY

כל הפקטות של השרת הראשי מתחילות ב-

Rdroid SERVER

בסוף כל שורה בפקטה כתוב-

/r/n

בשורה השנייה של הפקטה כתובה מטרת הפקטה-

רשימת מטרות:

DOWNLOAD

LOGIN

TASK

TASK\_RESULTS

NOTIFY\_SESSION\_ID

NOTIFY\_SESSION\_DISCONNECT

בשורה השלישית והלאה של הפקטה מצויינים הפרמטרים של הפעולה-

דוגמא של שורה שלישית ורביעית של פקטת LOGIN של הלקוח:

email:mighty

password:python

פרמטרים של פעולות:

DOWNLOAD (CLIENT):

requests a file to download from the server.

location - the location of the requested file.

LOGIN (CLIENT):

logins into the server.

username - the username of the client

password - the password of the client

LOGIN (SERVER):

informs the client whether the login was successful or not.

result - answers whether the login was successful or not.

TASK (SERVER):

sends a list of task that the client needs to do

task block - because this packet can send more than one task, each task will be defined in a block where all of its parameters are present.

<Task> - indicates the start of a task block.

</Task> - indicates the end of a task block.

id - the id of the task used to identify the results that are sent back to the server after the task was done.

type - the type of the task.

parameter - used when the task needs parameters. example: can provide a file address that is used by the client to request a file to download.

TASK\_RESULTS (CLIENT):

sends the results of the tasks that were given to the client.

id - the id of the task.

output - the results of the task or the information that it needs to return.

NOTIFY\_SESSION\_ID (PROXY)

notifies the server of a new session id that is given to a phone by the proxy.

session\_id - the id of the session

NOTIFY\_SESSION\_DISCONNECT(PROXY)

notifies the server of a session that disconnected.

session\_id - the id of the session.

כל הפקטות מתחילות במספר המייצג את אורך הפקטה (ללא המספר) ואחריו התו : (שגם לא נכלל בספירה).

כל הפקטה חוץ מהחלק שכותב את אורכה מומרת ל-Base64 ובמידה ומסודרת הצפנה אז היא גם מוצפנת לפני ההמרה ל-Base64.

בנוסף, במעבר בין השרת הראשי לשרת הפרוקסי ובין שרת הפרוקסי לשרת הראשי כל פקטה תקבל בתחילתה את ה-id של ה-session. לדוגמא (id=123):

123:message

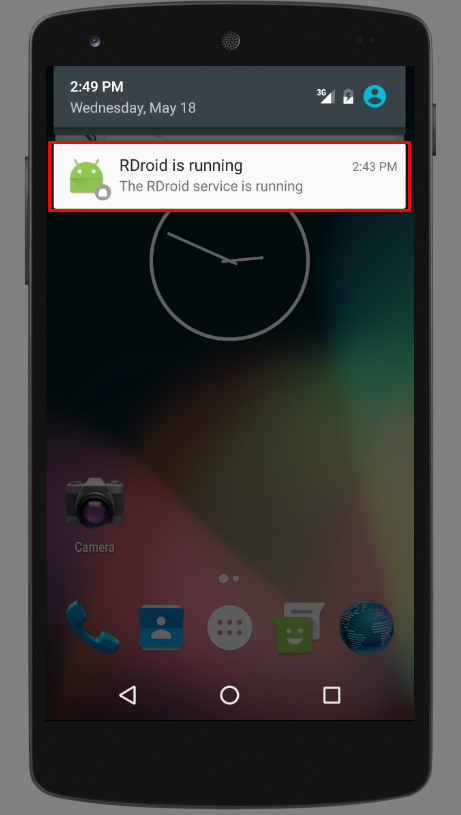
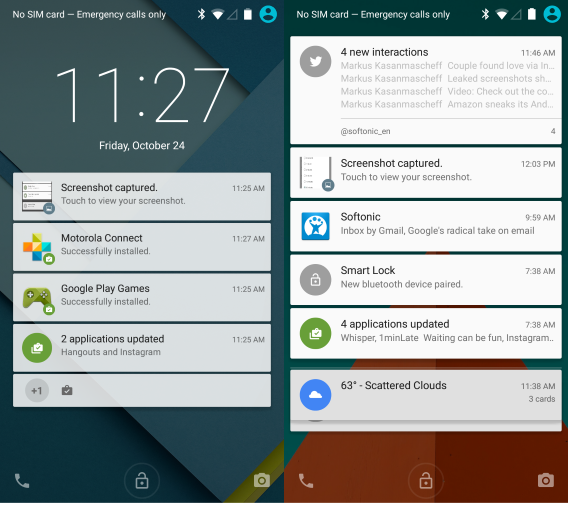
הנקודותיים מסמלות את סוף ה-id ותחילת ההודעה עצמה.

* **העברת קבצים:**

לקראת סוף הפרוייקט נתקלתי בבעיה בהעברת קבצים מכיוון שלא יכולתי לטפל במידע גדול. לעומת השרתים שפועלים על המחשב ויכולים להשתמש בהמון זיכרון Ram בכדי לשלוח קבצים בשלמותם, הטלפונים אינם מסוגלים להקצות להם הרבה זיכרון ולרוב גם מגבילים את הזיכרון שניתן לאפליקציות ושירותים. ישנם טלפונים יחסית חדשים שמקצים כ-30MB לאפליקציות. בכדי לשלוח קבצים בשלמותם צריך לשלוח אותם בחלקים. בשביל לשלוח את הקבצים השתמשתי ב-Streams שבעזרתם קיבלתי ושלחתי את המידע בחלקים. שליחת ההודעה בחלקים גרם לבעיה עם ההצפנה הסימטרית AES. היה צורך לשנות את המצב שימוש בהצפנה ל-CBC בכדי שיהיה ניתן לשלוח את ההודעות הללו ובסופו של דבר, היה יותר מדי קוד לשנות ולא מספיק זמן לעשות זאת בצורה מוצפנת.

* **הרצת האפליקציה ברקע:**

הפרוייקט שלי מבוסס על הצורך שהאפליקציה תעבוד כל הזמן ותגיב לשרת בכל עת. מכיוון שבאנדרואיד לא ניתן להריץ תהליך שהמשתמש לא יכול לראות, יצרתי שירות שמותחל ע"י האפליקציה ובפתיחת המכשיר. בכדי שהשירות ישאר דלוק הוא מייצר “Ongoing Notification” שהיא הודעה שמופיעה בחלקו העליון של המסך ולא נעלמת (אלא אם השירות נסגר או מגדיר את הנוטיפיקציה אחרת או שהמשתמש מגדיר שלא לראות את הנוטיפיקציה יותר).



* **אפשרות לקביעת הגדרות לשרת הפרוקסי:**

רציתי לאפשר קינפוג מהיר של שרת הפרוקסי בכדי שיהיה ניתן להתאים אותו לתנאים שונים. החלק העיקרי שניתן לקנפג הוא הכתובות IP ופורטים של שרת הפרוקסי והשרת הראשי, כך שיהיה ניתן להתאים את השרת במקרה שהשרת הראשי עובר לכתובת אחרת או משנה Port.

הסבר על שמירת ההגדרות בקובץ:

ההגדרות של השרת פרוקסי יישמרו במסמך טקסטואלי. ההגדרות יישמרו בצורה של מילון: כל הגדרה תשמר כמפתח וערכו. כל שורה במסמך תהיה הגדרה נפרדת. בכל שורה, המפתח הוא הטקסט עד ה-= וערכו הוא כל שאר הטקסט. שורות המתחילות ב-# הינן שורות הערה ולכן אינן נקלטות כהגדרות בתוכנה.

דוגמא:

#network

my\_ip=133.74.20.42

my\_port=8888

במקרה הזה שמור שכתובת השרת היא 133.74.20.42:8888.

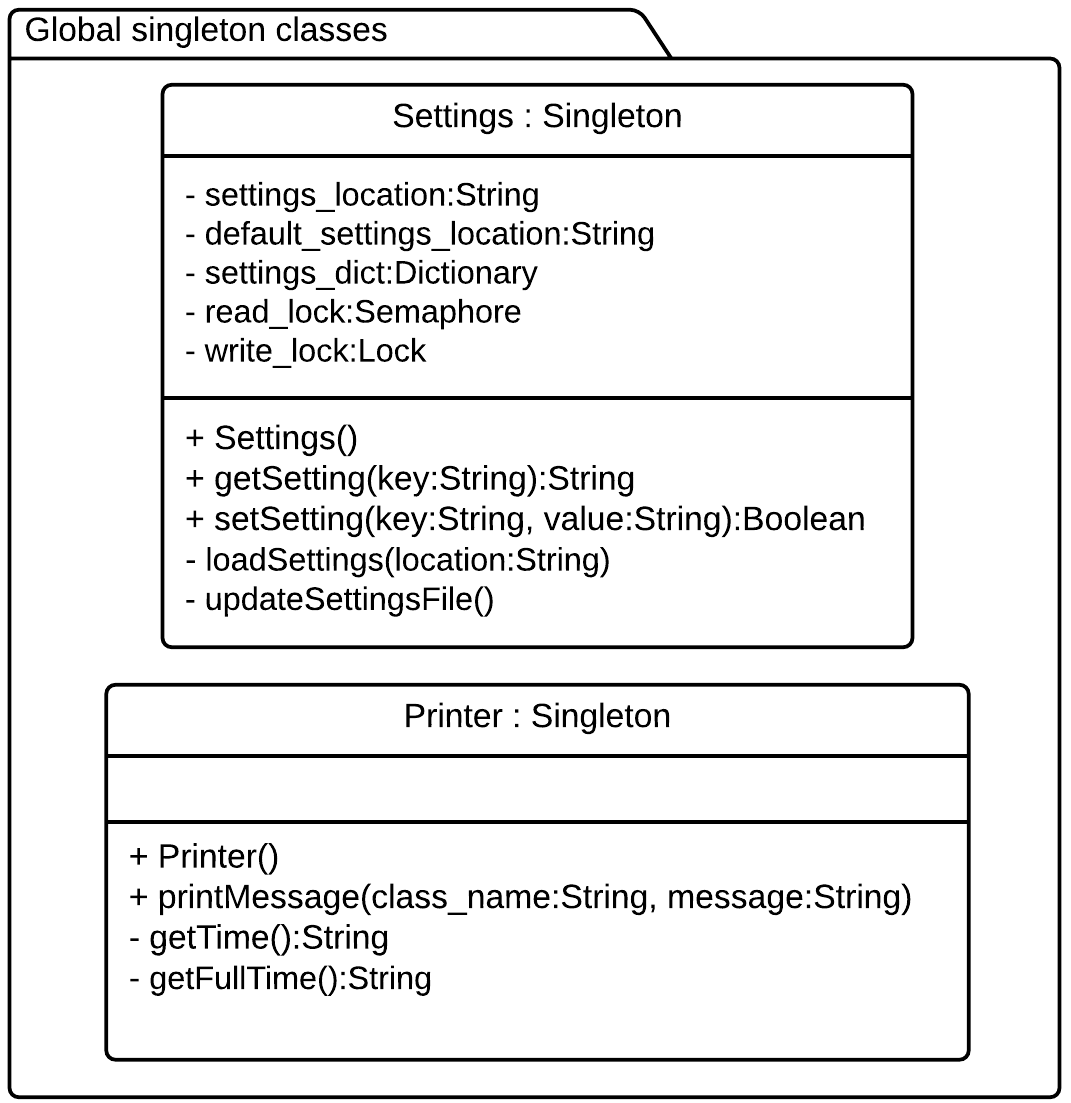
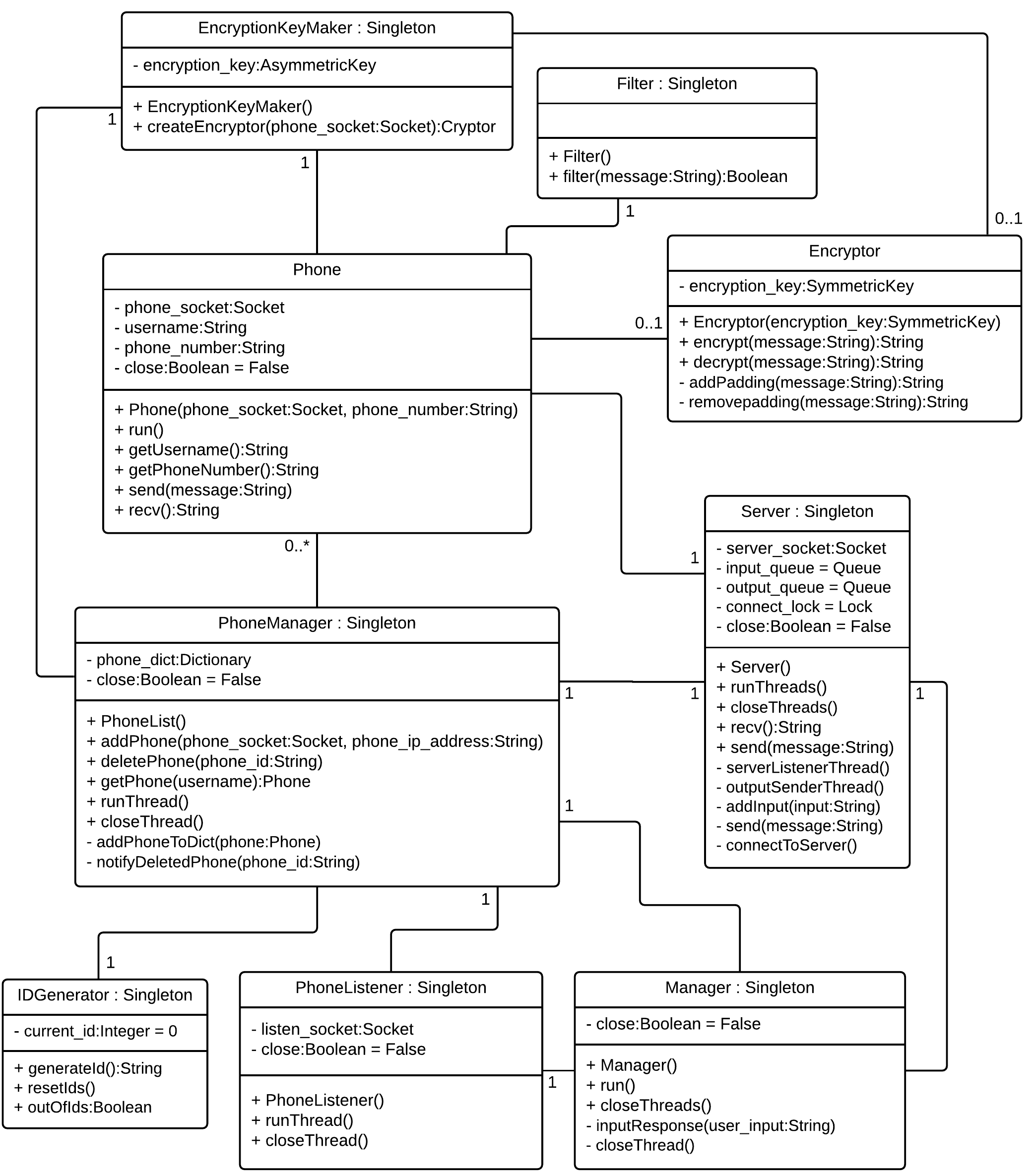
* **שמירת מידע באפליקציה:**

לאנדרואיד יש מערכת קבצים הדומה לזו של וינדוס. בנוסף למערכת הקבצים הזו יש עוד מערכת המנהלת את אחסון המידע של אפליקציות. המערכת הזאת שומרת את המידע של התוכנה בצורה של מילון ערכים. בנוסף ניתן להגדיר אם חלק מהמידע יהיה נגיש ע"י אפליקציות אחרות וכמובן לגשת למידע פומבי של אפליקציות אחרות.

בכדי לשמור את המידע חקרתי על השיטות השונות של שמירה ובסופו של דבר השתמשתי במערכת האחסון האישית לאפליקציה.

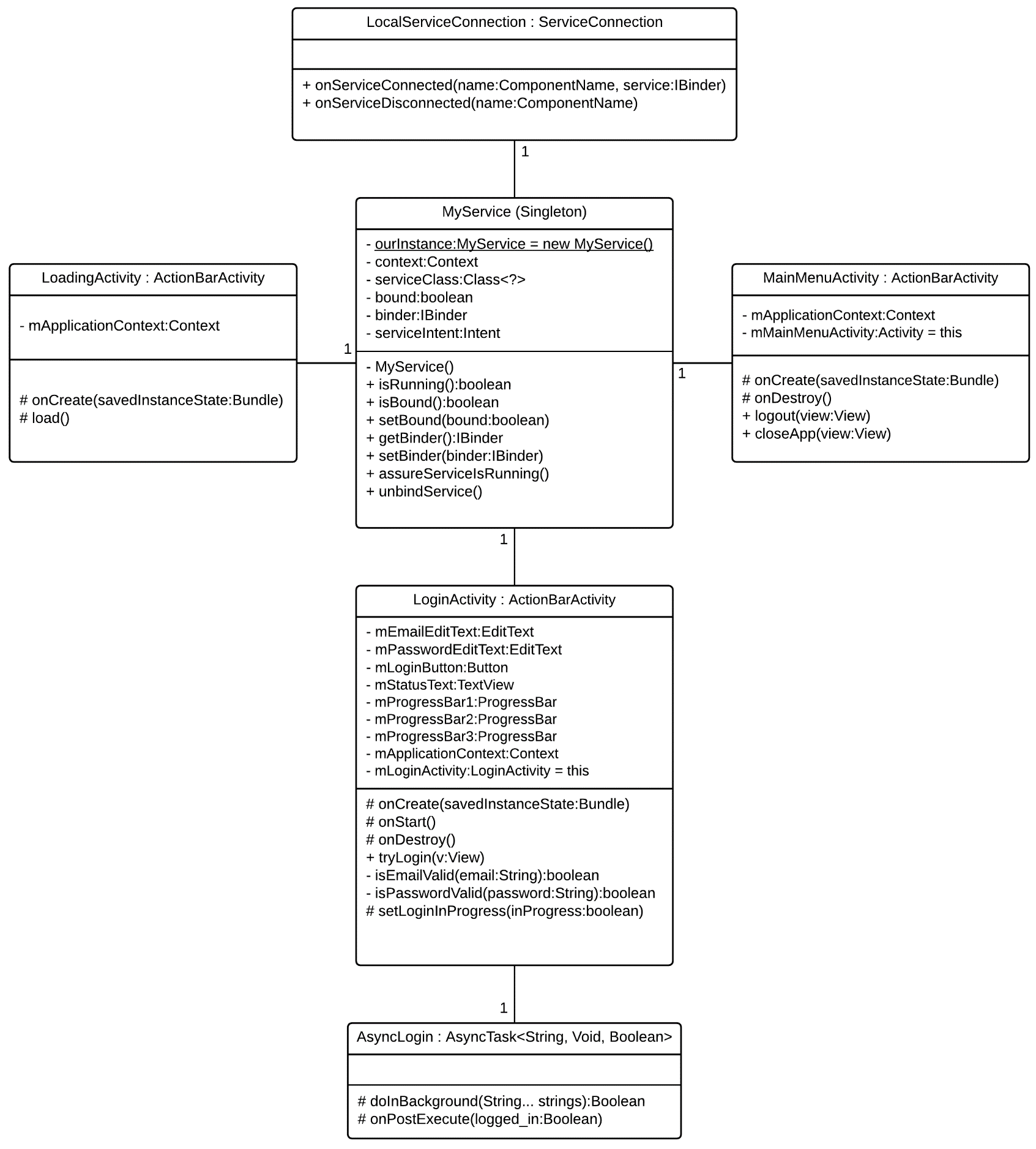
# תיאור מבנה המוצר

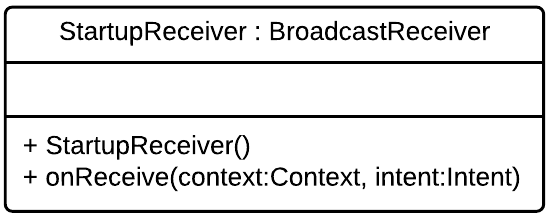
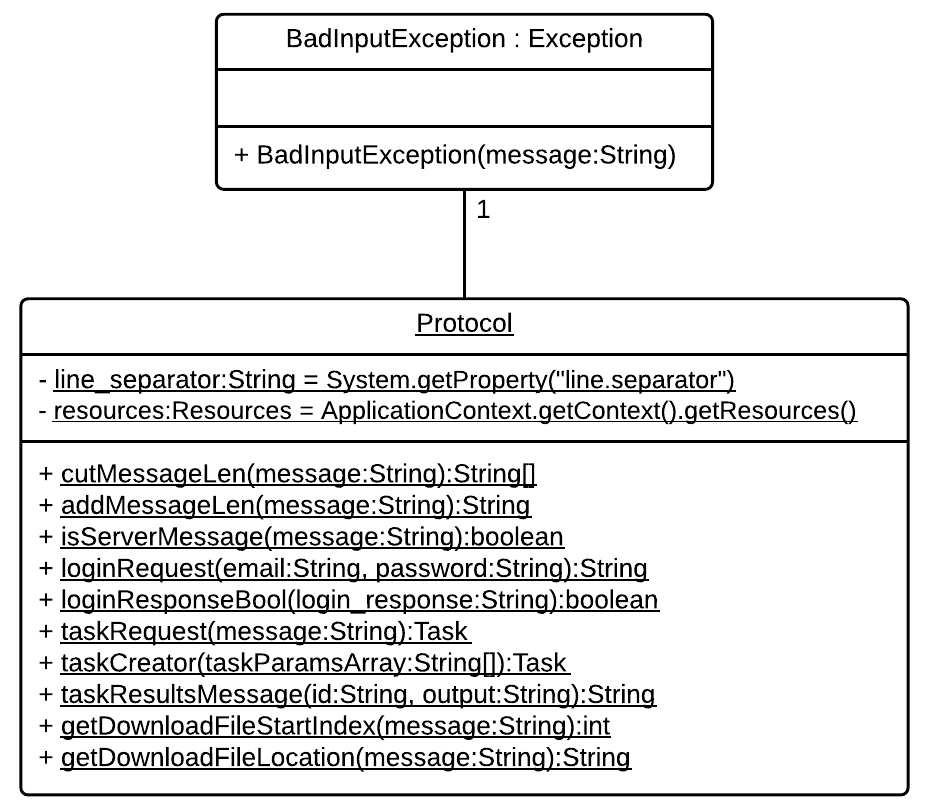
תרשימי UML של שרת ה-Proxy:

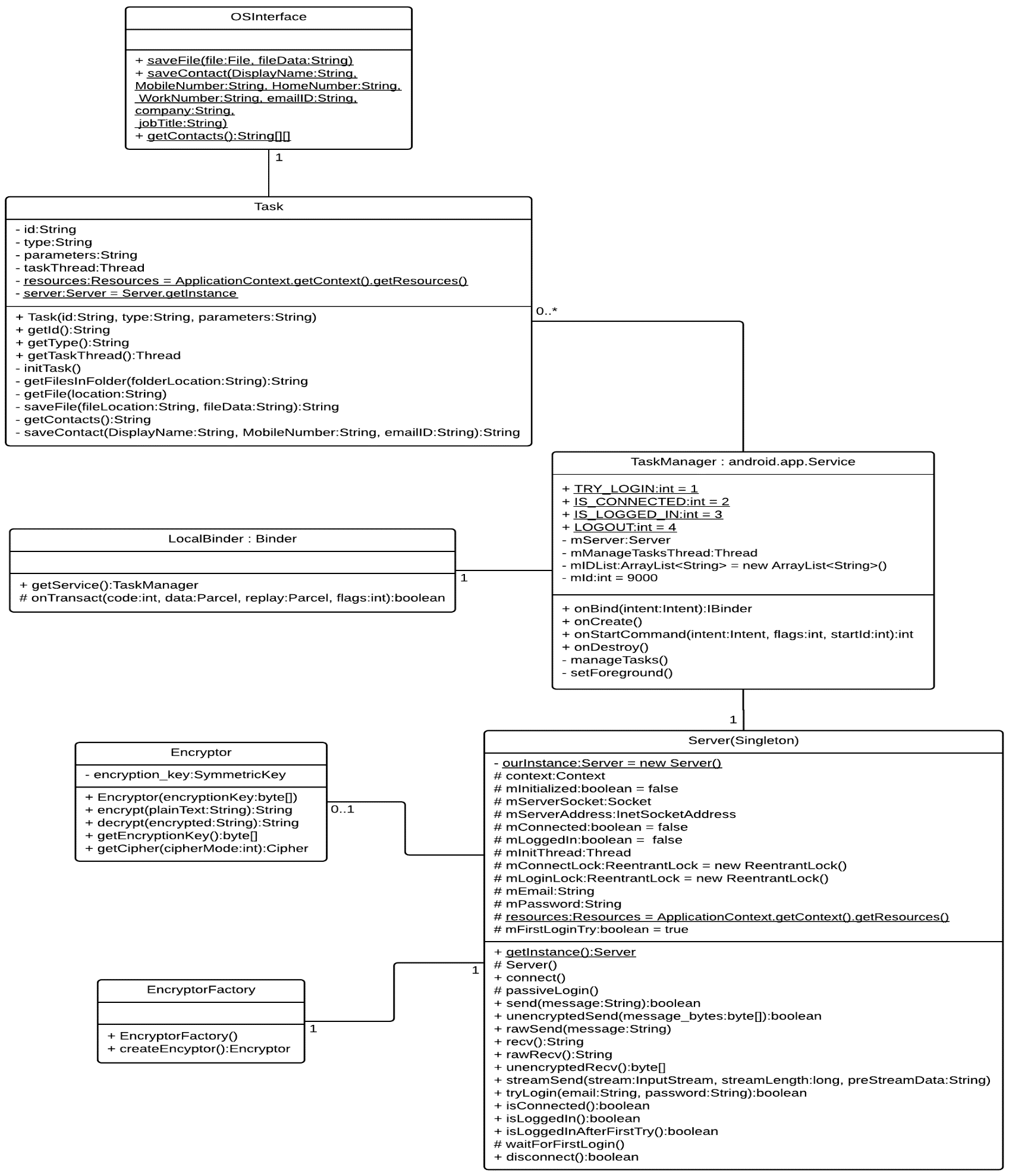


המודולים הם:

* **Main –** המודול שמתחיל את ריצת השרת. יוצר Manager חדש ומריץ אותו.
* **Manager** – המודול האחראי על התחלת Threads של תקשורת וניהול קלט מהמנהל (בממשק טקסטואלי).
* **Server** –המודול האחראי על ניהול התקשורת עם השרת הראשי.
* **Phone** – מודול האחראי על ניהול תקשורת עם טלפון.
* **PhoneListener** – מודול האחראי על קבלת חיבורים חדשים מטלפונים.
* **PhoneManager** – מודול האחראי על שמירת אובייקטי Phone הקיימים וניהול התקשורת איתם.
* **EncryptionKeyMaker** – אחראי על קביעת מפתח משותף עם טלפון ויצירת Encryptor למפתח הזה.
* **Encryptor** – אחראי על הצפנה ופיענוח של הודעות לפי המפתח שהוא הותחל איתו.
* **Filter** – פילטר להודעות שלא תואמות לפרוטוקול.
* **IDGenerator** – מייצר מזהים ייחודים שנועדו לזיהוי הטלפונים השונים.
* **Printer** – מדפיס הודעות לממשק של מנהל השרת (ממשק טקסטואלי). מדפיס את הזמן ואת המודול שמדפיס את ההודעה לפני ההודעה.
* **Settings –** אחראי על שמירה וקריאת הגדרות מקובץ ההגדרות.
* **Singleton –** Metaclass שהופך את מודולים ל-Singletons.

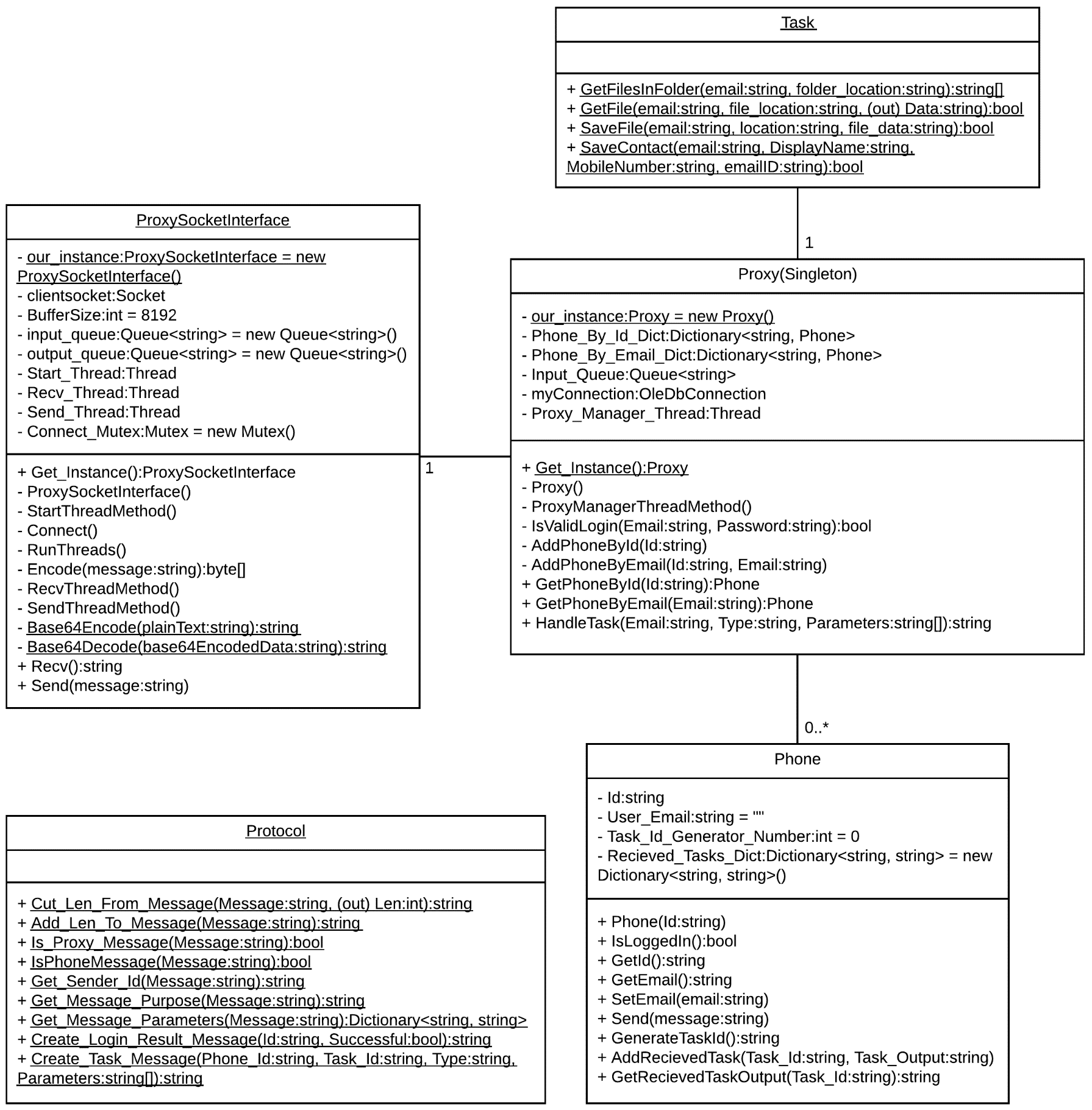
תרשימי UML של האפליקציה:





המודולים הם:

* **ה-Activities** – באפליקציה יש 3 Activities. התפקיד של המודולים הינו לנהל את ממשק המשתמש.
* **MyService** – מודול האחראי על ניהול התקשורת של האפליקציה עם ה-Service.
* **StartUpReciever** – מודול שמתחיל את ה-Service כאשר המכשיר נדלק (מריץ את TaskManager).
* **Protocol –** אחראי על כתיבת וניתוח הודעות לפי הפרוטוקול.
* **TaskManager –** אחראי על קבלת Tasks מ-Server, התחלתם ושליחת התוצאות שלהם חזרה לשרת. אחראי גם על התקשורת עם האפליקציה.
* **Server –** אחראי על תקשורת עם השרת.
* **EncryptorFactory –** אחראי על יצירת Encryptor להצפנת הודעות עם השרת.
* **Encryptor –** אחראי על הצפנה ופיענוח של הודעות עם השרת.
* **Task –** אחראי על ביצוע מטלות הנשלחות מהשרת והחזרת פלט מהן.
* **OSInterface –** ממשק להרצת פעולות של מערכת ההפעלה.

תרשים UML של הספריות לשרת הראשי:

המודולים הם:

* **Proxy –** אחראי על ניהול התקשורת עם שרת הפרוקסי, מעקב אחרי אילו טלפונים מחוברים ואילו התחברו למשתמש.
* **ProxySocketInterface –** אחראי על התקשורת עם שרת הפרוקסי.
* **Phone –** מייצג טלפון שמחובר לשרת הפרוקסי.
* **Task –** ממשק המפשט את התקשורת עם הטלפונים בכך שהוא מייצר הודעות של משימות ומחזיר את התשובה של הטלפון.
* **Protocol –** מייצר ומנתח הודעות לפי הפרוטוקול.

# מדריך התקנה למשתמש

אילוצים ודרישות מערכת

* השרתים פותחו עבור מערכת ההפעלה Windows. הם נבדקו ונוסו בגרסת **Windows 10** בלבד, ועל כן השימוש בהם בגרסה אחרת של Windows הוא באחריות המשתמש בלבד. האפליקציה עובדת על כל מכשיר אנדרואיד מגרסא 5 ומעלה.
* בנוסף, יש לוודא שמותקן בכל מחשב קצה ובשרת**Python 2.7** ושמותקנת בו ספריית **PyCrypto** עבור גרסה זו של Python. בביליוגרפיה נמצאים לינקים להורדת Python ולהתקנת הספרייה.
* כמו כן, צריכה להיות מותקנת במחשב השרת **.NET Framework 4.0** או גרסה עדכנית יותר (התוכנה נבדקה בגרסאות **4.0** ו- **4.5**).

התקנת והרצת התוכנה

1. צריך להוריד את הקוד של השרתים והאפליקציה מ-GitHub.
2. צריך להתאים את הגדרות השרתים בכדי שיתחברו אחד לשני וירוצו על הפורטים הרצויים, לעשות Port Forwarding לפורטים אם צריך, וגם באפליקצייה צריך להתאים את כתובת ה-ip והפורט למיקום של השרתים בקובץ strings.xml.
3. צריך ליצור קובץ Apk של האפליקציה עם השינויים בקובץ strings.xml ולהתקין אותו במכשיר.

# תיאור הממשק למשתמש

ממשק המשתמש הינו מאוד מינימליסטי.

אני לא אפרט על הממשק של האתר מכיוון שאני לא עיצבתי אותו.

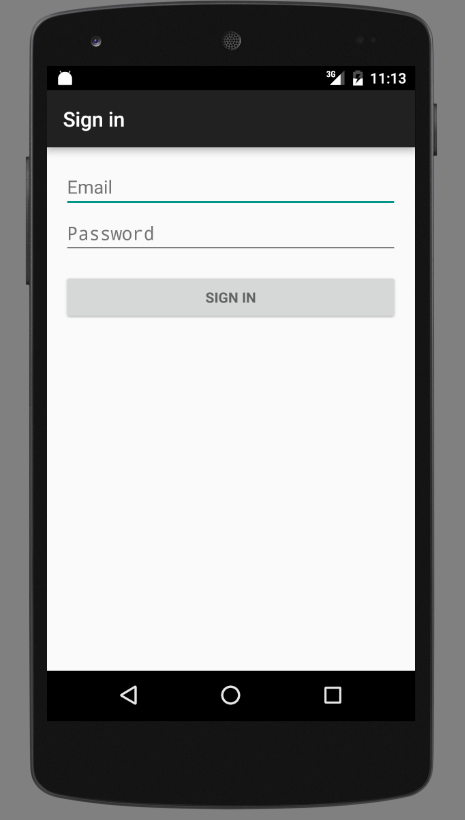
שרת הפרוקסי:

שרת הפרוקסי כתוב בפייטון ולכן יש לו ממשק טקסטואלי בסגנון CMD. השרת מדפיס הודעות על פעולתו וניתן לכתוב לו stop כדי שיסגור את עצמו.

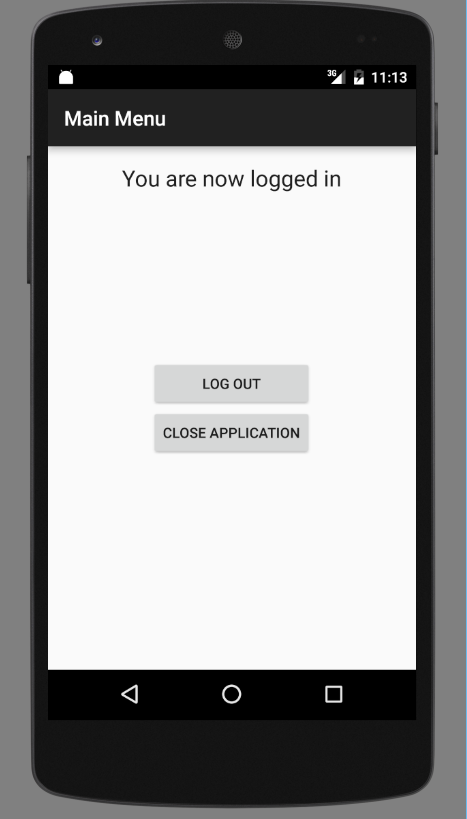
האפליקציה:

אפליקציות בטלפונים מורכבות וויזואלית מ-Activities, שהן כמו Forms ב-Windows, ועמודים באתרי אינטרנט. באפליקציה שלי ישנן 3 Activities:

* **Loading Activity** – נועדה בשביל להוות ממשק למשתמש בזמן שהאפליקציה מתחילה לרוץ. אם ישנם שם משתמש וסיסמא בזיכרון, ה-Activity מתחילה את Main Menu Activity, אחרת, היא מתחילה את Login Activity.
* **Login Activity** – נועדה בכדי שהמשתמש יוכל להזין שם משתמש וסיסמא כדי להתחבר לחשבון שלו. ב-Activity יש שתי תיבות טקסט להזנת שם משתמש וסיסמא וכפתור להיתחברות. אם המשתמש מתחבר בהצלחה, ה-Activity שומרת בזיכרון את שם המשתמש והסיסמא.

תמונה:

* **Main Menu Activity** – מהווה ממשק למשתמש שבו הוא יכול לבצע פעולות אחרי ההתחברות. יש בה שני כפתורים: כפתור להיתנתקות מהמשתמש שמעביר את האפליקציה ל-Login Activity וכפתור לסגירת האפליקציה (שהוא קצת מיותר).

תמונה:

# מבט אישי על העבודה ותהליך הפיתוח

בעבודתי הייתי צריך לכתוב אפליקציה למערכת הפעלה שונה ובשפה שונה. לאורך העבודה על הפרויקט השקעתי המון זמן בלמידת Android ואיך לעבוד בסביבת העבודה Android Studio.

האתגרים והקשיים שנתקלתי בהם במהלך פיתוח הפרויקט הם:

הערכת הפתרון לעומת התכנון:

* מלבד הנקודות שהוזכרו לעיל, המוצר המוגמר תואם בדיוק את התכנון המקורי.

ההמלצות לשיפורו הן אם כן:

* בנוסף, אין ספק שיש מקום לשיפור המוצר מבחינת אבטחת מידע ומימוש פרוטוקול יותר מקצועי, יעיל ומאובטח ומבחינת עיצוב הממשק כדי שיהיה יפה וברור יותר.

# ביבליוגרפיה

1. אתר הדרכה לסייבר שיוסד על ידי המנחה שלי מיכאל:  
   [https://sites.google.com/site/coursecyber](https://sites.google.com/site/coursecyber/)
2. הפורום הגדול ביותר לשאלות ותשובות בנושאי תכנות:  
   [http://stackoverflow.com](http://stackoverflow.com/)
3. MSDN, הרשת הרשמית של Microsoft למפתחים:  
   <http://msdn.microsoft.com/library>
4. התיעוד הרשמי של Python:  
   [https://docs.python.org/2.7](https://docs.python.org/2.7/)
5. הוויקי הרשמית של Python:  
   [https://wiki.python.org/moin](https://wiki.python.org/moin/)
6. אתר ללימוד Pythonו-Java:  
   <https://www.codecademy.com>
7. להורדת Python:

<https://www.python.org>

1. להתקנת ספריית PyCrypto בשביל Python:

<http://www.voidspace.org.uk/python/pycrypto-2.6.1>

1. להורדת Android Studio:

<https://developer.android.com/studio/index.html>

1. להורדת Visual Studio:

<https://www.visualstudio.com>

**קוד התוכנה**