

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](#_Toc418019495)

[מבוא ורקע כללי 4](#_Toc418019496)

[תיאור המוצר המוגמר 6](#_Toc418019497)

[שפת התכנות וסביבת העבודה 7](#_Toc418019498)

[ניסוח וניתוח הבעיה האלגוריתמית 8](#_Toc418019499)

[תיאור אלגוריתמים קיימים 9](#_Toc418019500)

[הפתרון הנבחר 11](#_Toc418019501)

[פיתוח הפתרון בשכלול הקוד עם שפת התכנות 15](#_Toc418019502)

[תיאור המודולים המרכיבים את המוצר 23](#_Toc418019503)

[פירוט מבני הנתונים 26](#_Toc418019504)

[פירוט מאגרי המידע 27](#_Toc418019505)

[השוואת העבודה עם פתרונות ויישומים קיימים 29](#_Toc418019506)

[מדריך התקנה למשתמש 31](#_Toc418019507)

[תיאור הממשק למשתמש 32](#_Toc418019508)

[מבט אישי על העבודה ותהליך הפיתוח 35](#_Toc418019509)

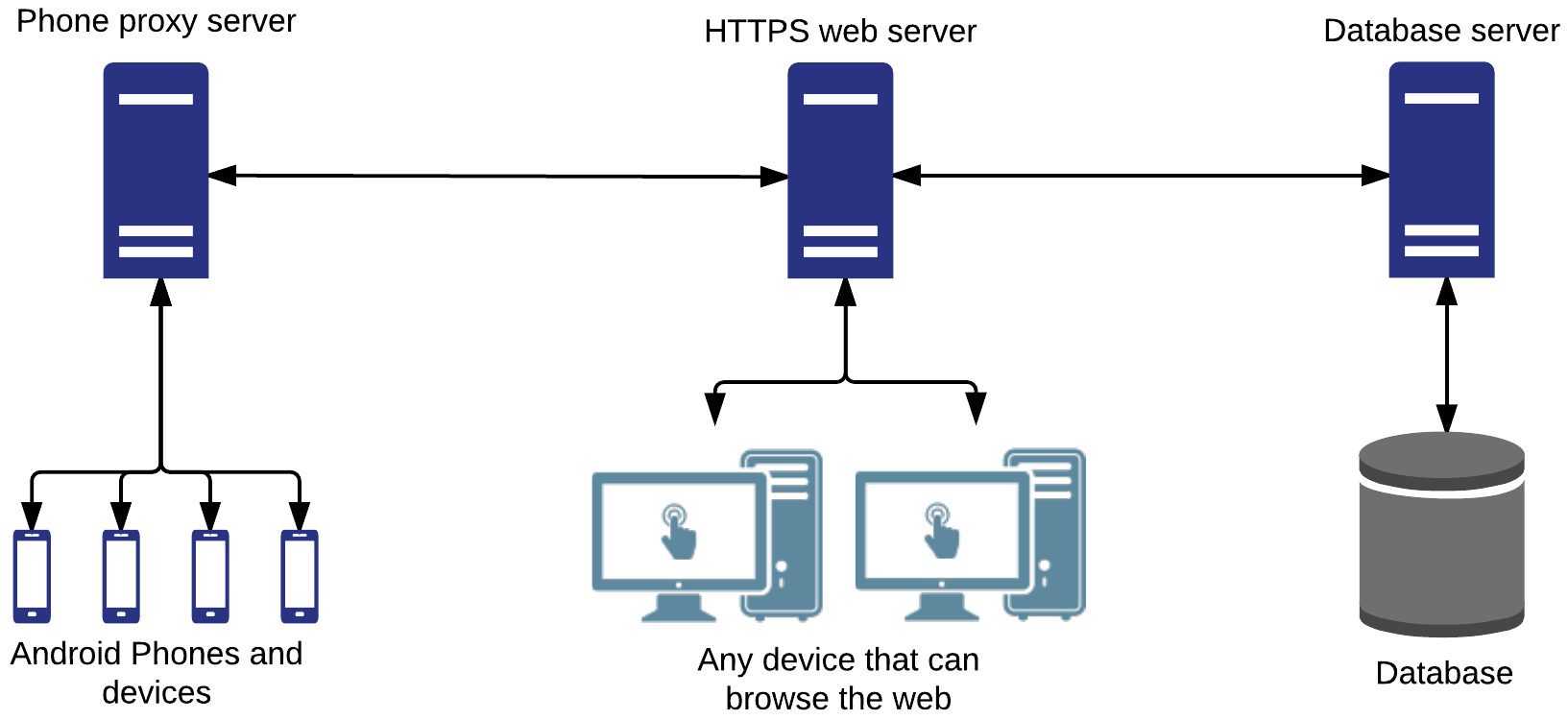
[ביבליוגרפיה 37](#_Toc418019510)

[קוד התוכנה 38](#_Toc418019511)

# תקציר כולל ורציונל הפרויקט

כיום לרוב האנשים יש שתי עמדות עבודה: עמדה ניידת ועמדה נייחת כאשר הטלפון הוא העמדה הניידת והמחשב הוא העמדה הנייחת. כאשר האנשים משתמשים במחשב, לרוב הטלפון אינו לידם והם לא יכולים לגשת למידע ולנהל אותו. הפתרון שלנו לבעיה זו הוא "RDroid", שירות שיתן ללקוח לנהל את הטלפון מהמחשב. הלקוח יתקין את אפליקציית השירות בטלפון ויתחבר למשתמש הפרטי שלו בעזרת שם וסיסמא. כאשר הלקוח ירצה לנהל את המכשיר, הוא יתחבר לאותו המשתמש דרך האתר ויוכל לנהל את המכשיר משם.

הסיבה שבחרתי לעבוד בפרוייקט על ניהול טלפונים מרחוק היא שלעיתים קרובות מצאתי את עצמי במצבים בהם לא הייתה לי דרך לגשת למידע חשוב שהיה לי על הטלפון כשהוא לא היה לידי או כאשר לא היה לי כבל חיבור לטלפון.

התוכנה שלי מספקת שירות להעברת קבצים בין המחשב לטלפון. התקשורת נעשית דרך שרת Proxy ושרת המשמש כאתר אינטרנט.

# מבוא ורקע כללי

הפרויקט שלי מבוסס על **שליטה מרחוק** (**Remote Control**). בין המחשבים ברשת נוצר חיבור עם השרת המרכזי, וחיבור זה מאפשר שליטה מרחוק עליהם (שליטה מוגבלת, בהתאם לאפשרויות שהתוכנה מספקת). השליטה מרחוק מאפשרת איסוף מידע ממחשבי הקצה והרצת פקודות עליהם (כגון סגירת תהליכים), וזהו הבסיס לפרויקט כולו.

הנושא המרכזי של הפרויקט הוא **ניתוח משאבי מערכת**. למערכת מחשב יש סוגים שונים של משאבים, כגון: זמן מעבד, זיכרון פיזי, דיסק קשיח, תעבורת רשת ועוד. את המשאבים חולקים ביניהם כל התהליכים הרצים במערכת.

הפרויקט שלי עוסק בניתוח של נתוני השימוש במשאבי המערכת. הן של נתוני השימוש הכולל במשאבי המערכת, והן של נתוני שימוש של תהליכים נפרדים במשאבי המערכת. מטרת ניתוח הנתונים היא לזהות תהליכים שגוזלים יותר מדי משאבי מערכת מיתר התהליכים ובכך מאטים את המחשב, ולפעול בהתאם למניעת מצבים כאלו.

לפרויקט יש מספר מרכיבים עיקריים ולהלן סקירת רקע לגביהם:

* מרכיב עיקרי בפרויקט שלי הוא **תקשורת**. תקשורת בעולם המחשבים מבוססת על Sockets, שהם נקודות קצה עבור זרם נתונים בתקשורת בין תהליכים על גבי רשת מחשבים. רוב הסוקטים כיום מבוססים על ה- Internet Protocol (IP).

בפרויקט שלי התקשורת בין השרת ללקוחות מתבצעת באמצעות Internet Sockets, כאשר בשכבת התעבורה יש שימוש בפרוטוקול ה- TCP, שייחודו בכך שהוא מוודא שכל הנתונים שנשלחו התקבלו ובסדר הנכון, ואחרת שולח אותם שוב. בפרויקט שלי יש חשיבות לכך שלא יהיה איבוד מידע ופרוטוקול ה- TCP/IPמתאים בדיוק למטרה זו.

* מרכיב חשוב נוסף בפרויקט הוא **קבלת מידע לגבי תהליכים**. המודול האחראי על קבלת המידע לגבי התהליכים הרצים במחשב משתמש ב- WMI. הוא פונה אליו באמצעות שאילתות SQL ומקבל מידע מפורט לגבי כל התהליכים שרצים במחשב, וגם מידע לגבי השימוש הכולל בזיכרון, בזמן מעבד ופעולות I/O.

כדי להבין איך זה עובד עלינו להבין ראשית מהו ה- WMI. ה- WMI הוא המימוש של Microsoft ל- Web-Based Enterprise Management, שזו יוזמה של התעשייה לפתח תקן של גישה למידע ניהולי בסביבה ארגונית. כלומר ה- WMI הוא ממשק של מערכת ההפעלה שמאפשר לקבל מידע לגבי רכיביה. בעזרתו ניתן לקבל מידע לגבי רכיבי המערכת, החומרה והתוכנה (התהליכים שרצים).

* בנוסף, הפרויקט משלב עבודה עם **מסד נתונים**. הנתונים שהתוכנה אוספת לגבי כל אחד ממחשבי הרשת נשמרים במסד הנתונים, שהעבודה איתו היא בטכנולוגיית ה- SQLite, שמממשת את רוב הסטנדרט של SQL (שהיא השפה המקובלת לטיפול ועיבוד מידע בבסיסי נתונים). להבדיל מרוב המערכות לניהול בסיסי נתונים, SQLite אינה תהליך עצמאי נפרד, אלא ספריה בתוך התהליך באופן אינטגרלי. ‏ זוהי בחירה פופולרית כאשר מפתחים בסיס נתונים מוטמע עבור צד לקוח כשירות ניהול מידע מקומי. מסד הנתונים שאותו מנהלת התוכנה שלי שומר מידע לשימוש מקומי בלבד, ולכן בחרתי להשתמש ב- SQLite שלא דורש הרצה של SQL Server.
* כמו כן, בפרויקט ישנה עבודה עם **קובץ הגדרות**. התוכנה מאפשרת למנהל הרשת לקבוע הגדרות שיכתיבו את אופן פעולתה בתרחישים השונים, כאשר ההגדרות נשמרות בקובץ XML.

ה- XML הוא תקן לייצוג טקסטואלי של נתונים במחשבים. ב- XML טבעי מאוד לסדר מידע באופן היררכי (עץ). הייצוג עצמו מתבצע תוך שימוש בתגיות (Tags) שמגדירות רכיבי מידע (Elements). בתוך רכיבי המידע ניתן לאכסן את המידע עצמו או רכיבים נוספים. קובץ ה- XML בפרויקט שלי מכיל את ההגדרות שהן אוסף כללים המכילים תתי-כללים במבנה היררכי.

# תיאור המוצר המוגמר

שם המוצר: Computer Resources Monitoring.

המוצר אמור לבצע ניטור ובקרה על השימוש במשאבים של מחשבים ברשת מקומית, לזהות פעילות חריגה של שימוש מופרז במשאבים, ולטפל בה בהתאם להוראות מנהל הרשת. המוצר למעשה נותן למנהל הרשת כלי לפיקוח ושליטה על הנעשה במחשבים שברשת.

המטרה המרכזית של המוצר היא מניעת עומס על המחשבים ברשת וזיהוי פעילות חריגה (העלולה להעיד על וירוסים במחשב). המוצר מבצע זאת ע"י ניטור ובקרה על מחשבי הרשת באמצעות חיבור לשרת מרכזי. הוא מקבל אליו מידע שוטף לגבי השימוש במשאבים של כל אחד מהמחשבים ומאפשר הגבלה ובקרה על השימוש במשאבים באמצעות התראות, סגירת תהליכים וכיבוי המחשב. המטרה היא לאפשר למנהל הרשת לפקח על הנעשה במחשבי הרשת – לקבל התראות לגבי פעילות חריגה ולקבוע דרכי פעולה שהתוכנה תבצע באופן אוטומטי במקרים חריגים שונים, ובכך תשמור על פעילות תקינה וחלקה של המחשבים.

השאלות שהמערכת עונה עליהן:

1. **כיצד להגן על המידע הרגיש מפני גורמים לא מורשים?** התוכנה מצפינה את כל התקשורת בין מחשבי הרשת לשרת, כך שגורם לא מורשה לא יוכל להפיק שום תועלת מהאזנה למידע הנשלח, הוא לא יוכל לגלות אילו תהליכים רצים על מחשבי החברה וכמה משאבים הם צורכים.
2. **כיצד לעבוד בו-זמנית עם כל המחשבים ברשת?** כאשר מחשבי הרשת מתחברים לשרת, הוא פותח Thread נפרד לכל אחד ומסוגל לתקשר במקביל עם כולם, לקבל מהם מידע ולשלוח להם פקודות, ולכל מחשב ברשת יש בצד השרת מסד נתונים נפרד שבו מאוחסן כל המידע שנאסף עליו.
3. **כיצד להחליט באילו פעולות יש לנקוט בתרחישים הבעייתיים השונים?** התוכנה מאפשרת למנהל הרשת לקבוע את כל ההגדרות שקובעות את אופן פעולתה, ובמסגרת ההגדרות ניתן למשל לקבוע whitelist של תהליכים שהתוכנה לא תתערב בפעולתם, תאפשר להם לצרוך כמה משאבים שהם צריכים ולא תשלח התראות לגביהם.
4. **מה לעשות במקרה שמחשב מסוים מפסיק להגיב לחלוטין?** כאשר מחשב כלשהו נתקע ומפסיק להגיב ולתקשר עם השרת, המערכת מנתקת את התקשורת איתו ושולחת באופן מיידי התראה על הבעיה למנהל הרשת ול- IT כדי שיוכלו להתייחס אליה (לגשת באופן פיזי למחשב ולטפל בו).

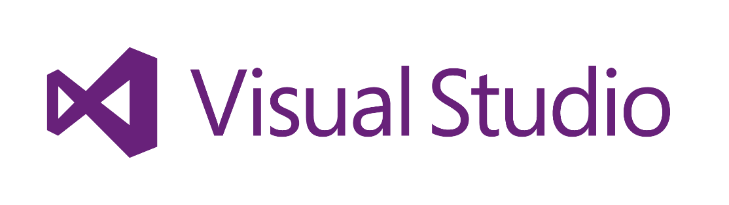
# שפת התכנות וסביבת העבודה

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 החליטו שזאת שפה נוחה והביצועים שלה לא נוראיים בהשוואה ל-C++ ו-C. התמיכה של אנדרואיד וסביבת העבודה Android Studio ב-Java היא מאוד טובה ולכן עבדתי איתה.

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.

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

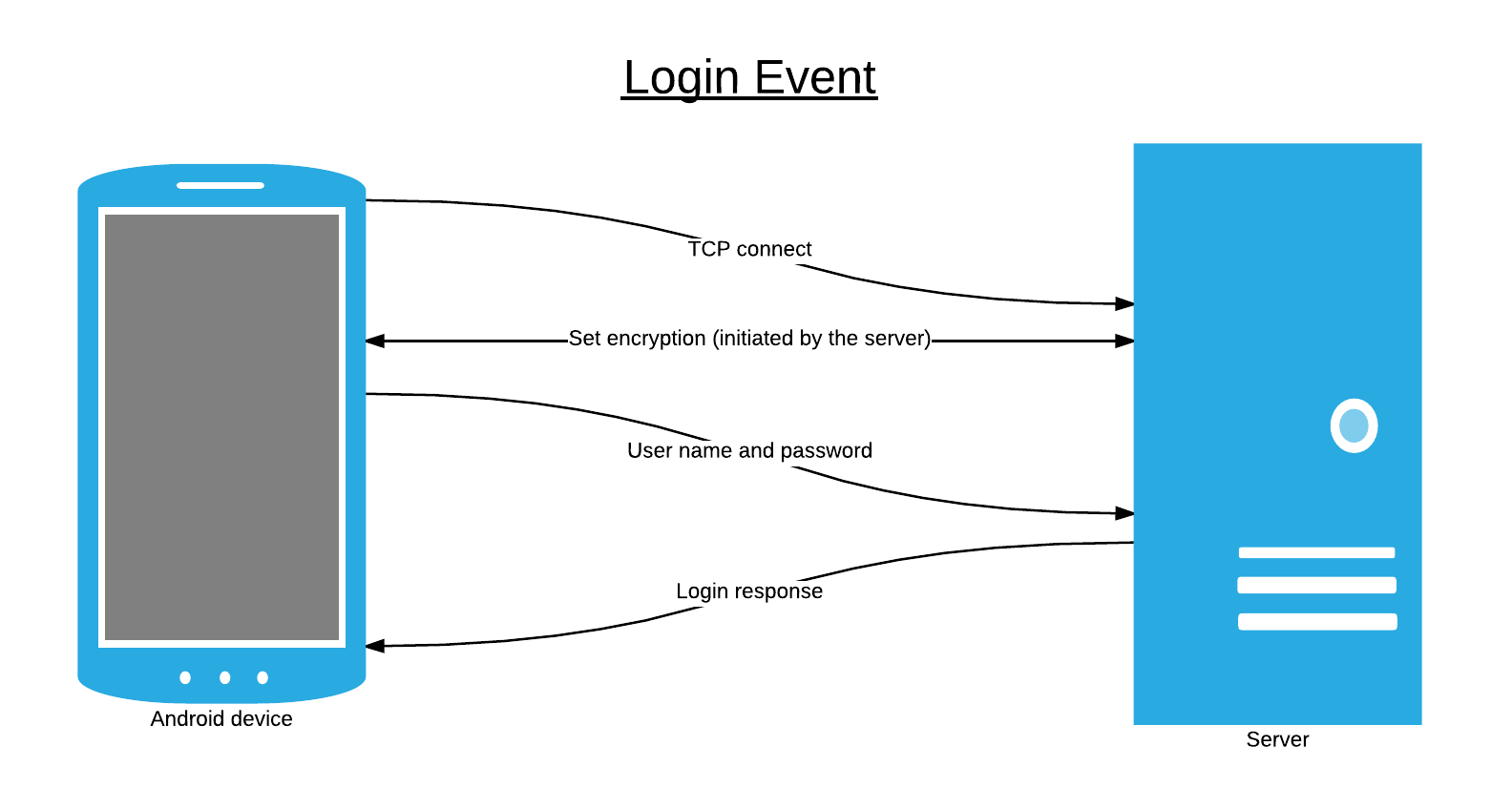
במהלך פיתוח המוצר נתקלתי בבעיות אלגוריתמיות רבות:

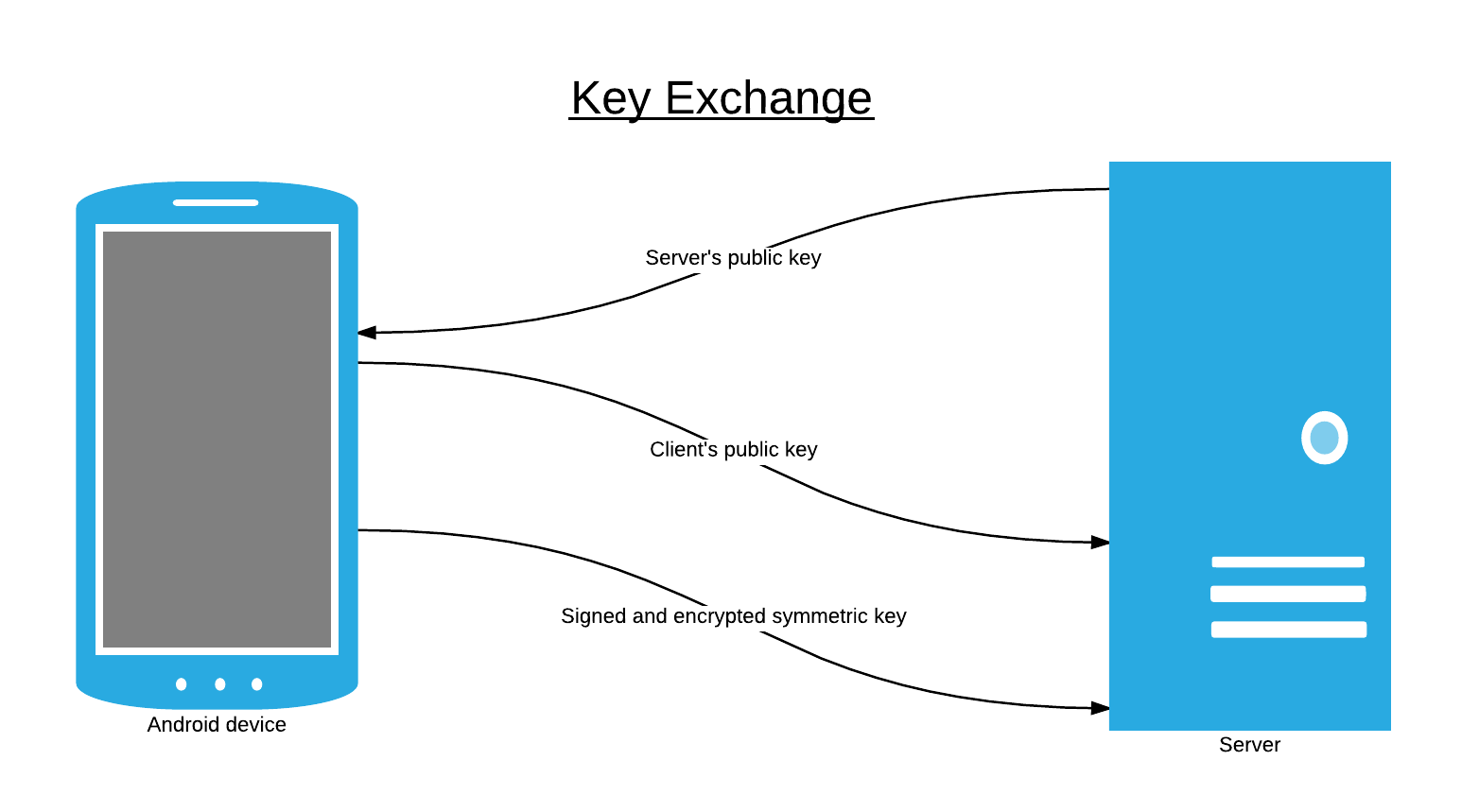
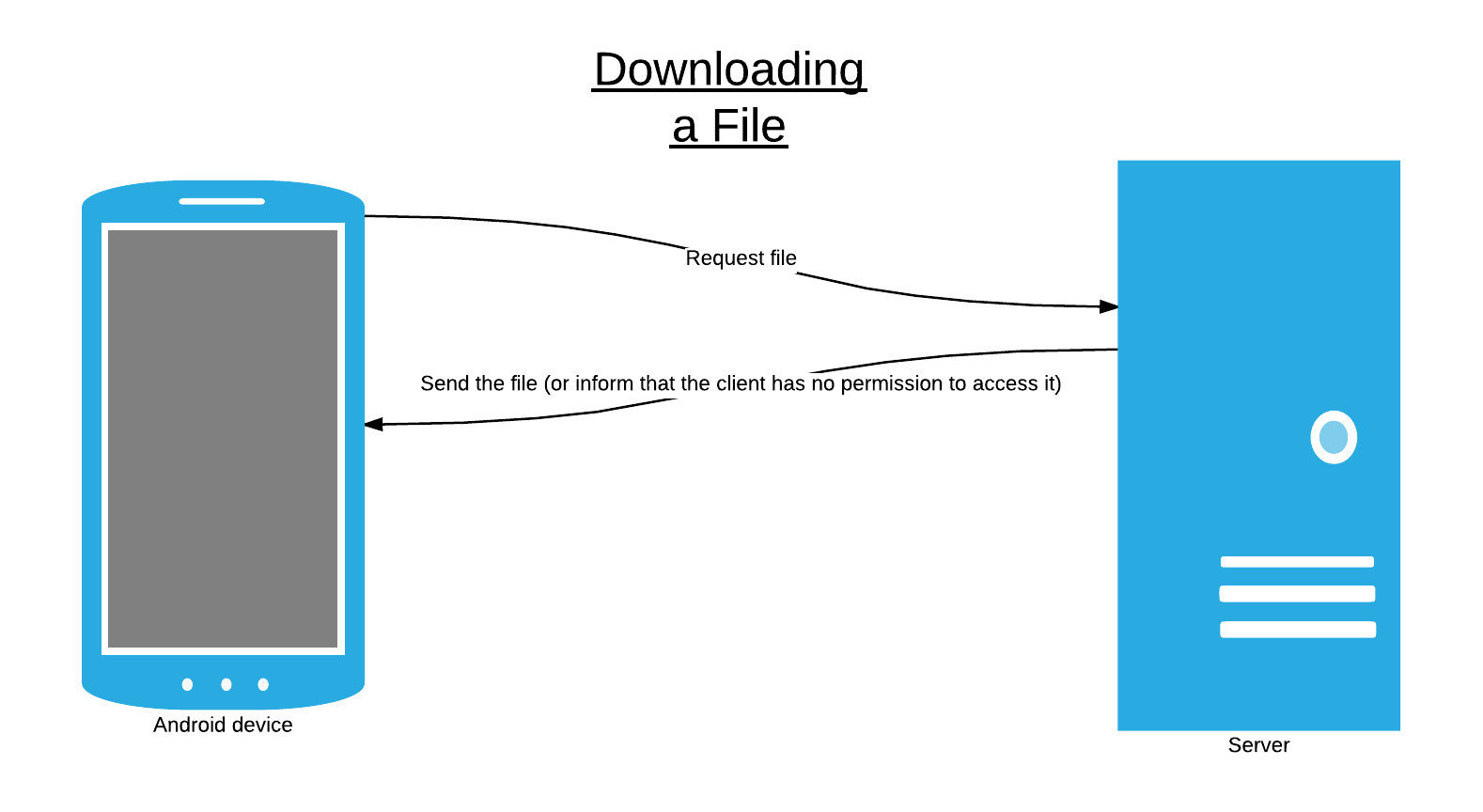
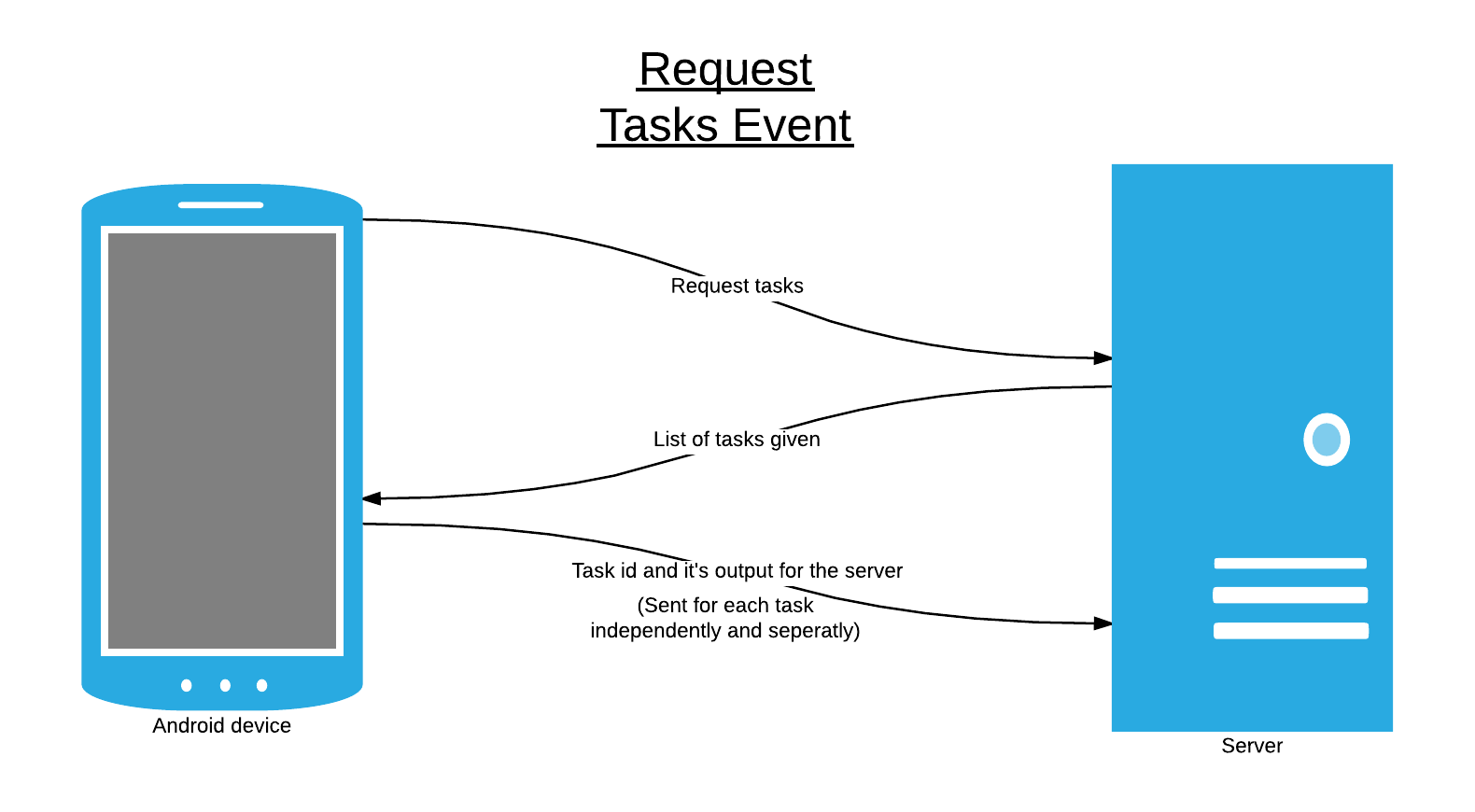
* **מימוש שרת מולטי-קליינט:**

הבעיה הראשון שעמדה לפניי בפיתוח המוצר 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:** השרת צריך לדבר עם הלקוחות בפרוטוקול מסוים, שמגדיר את אופן התנהלות התקשורת ביניהם. הייתי צריך לפתח פרוטוקול משלי שבו ידבר השרת עם הלקוחות שמתחברים אליו.
* **מימוש אבטחת מידע:** כדי למנוע מהמידע לעבור בין השרת ללקוחות באופן חשוף הייתי צריך לממש אבטחת מידע, כלומר להוסיף לפרוטוקול שלי הצפנה. 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

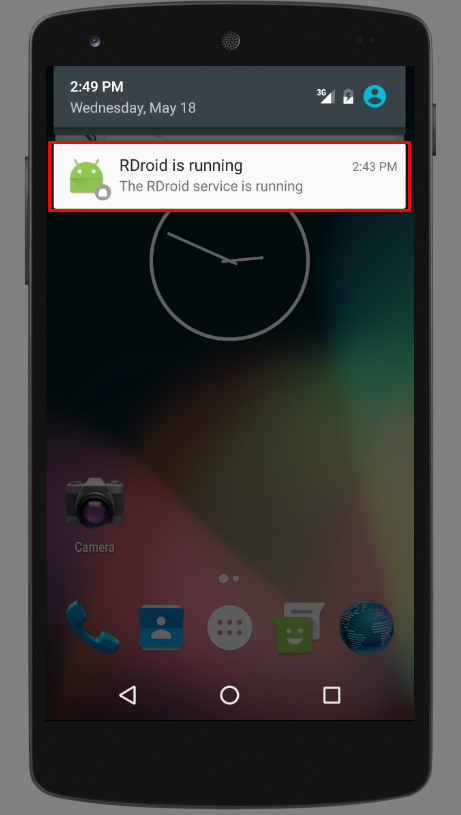
הנקודותיים מסמלות את סוף ה-id ותחילת ההודעה עצמה.

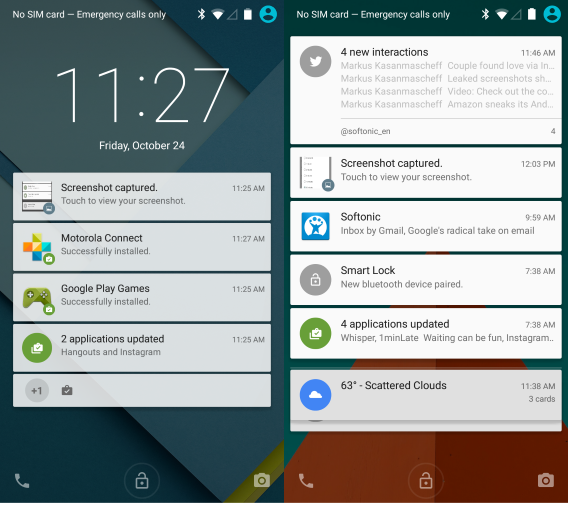
* **העברת קבצים:**

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.

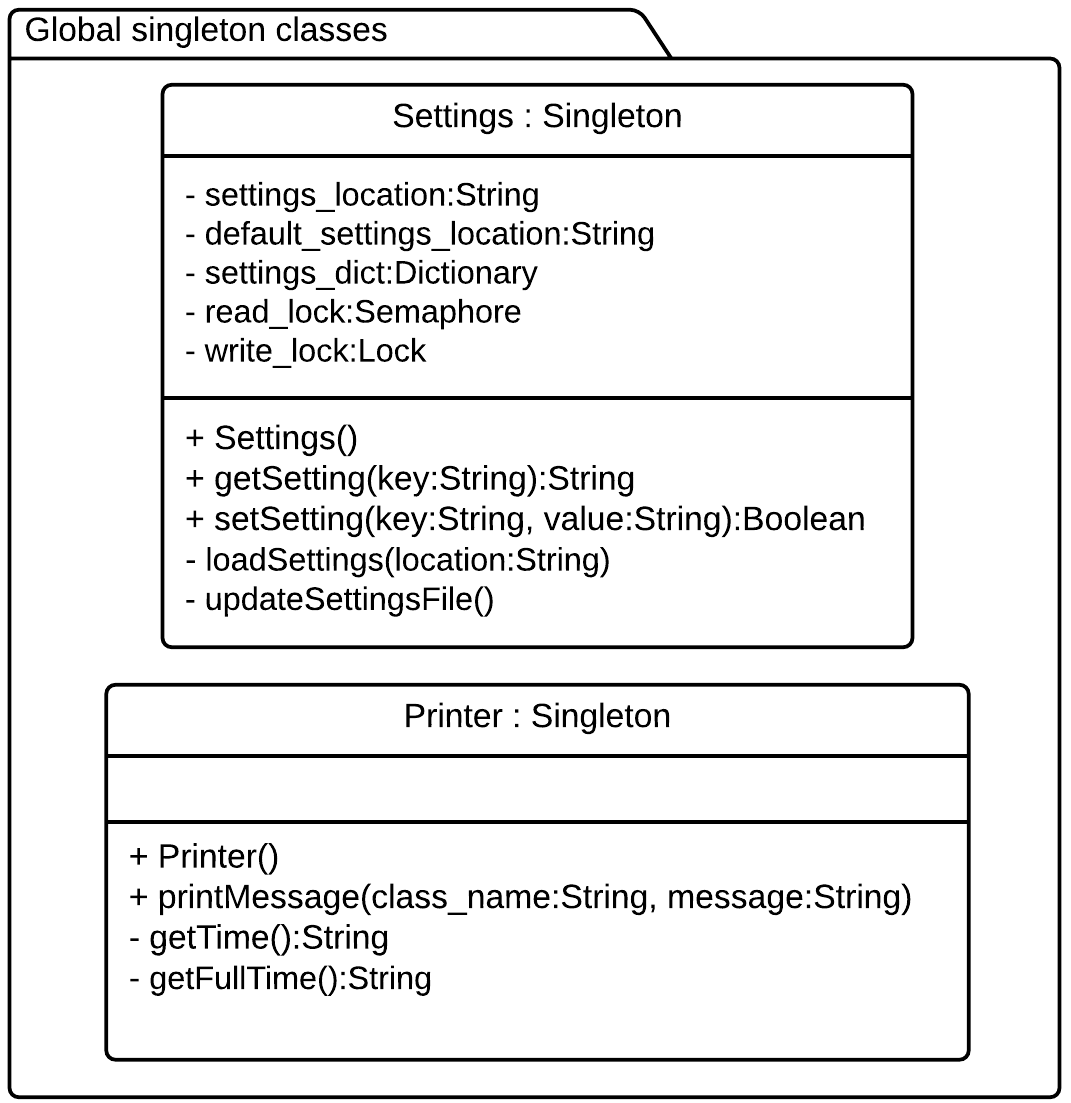
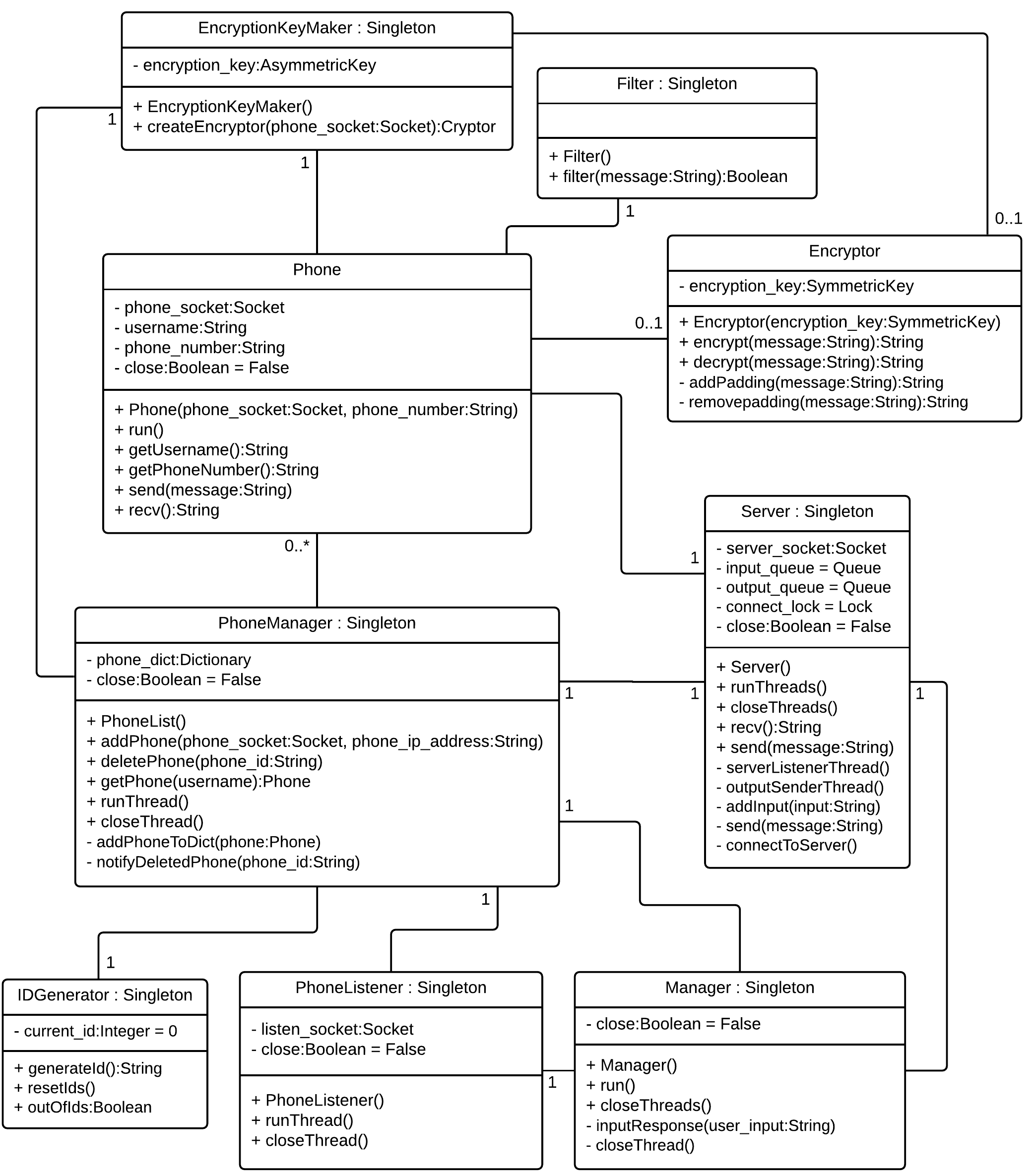
* **שמירת מידע באפליקציה:**

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.

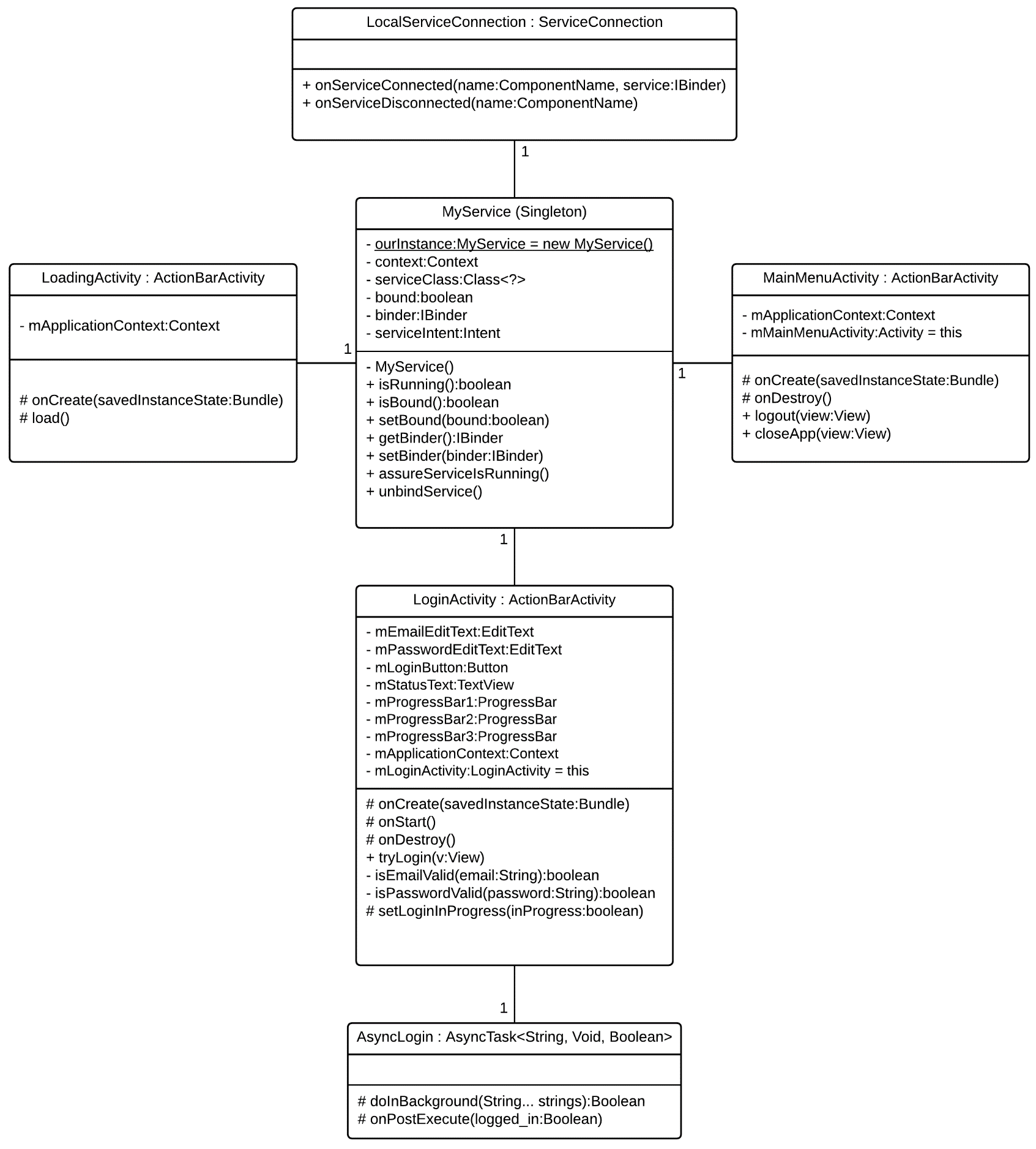
# תיאור מבנה המוצר

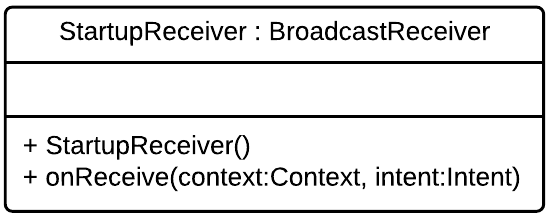
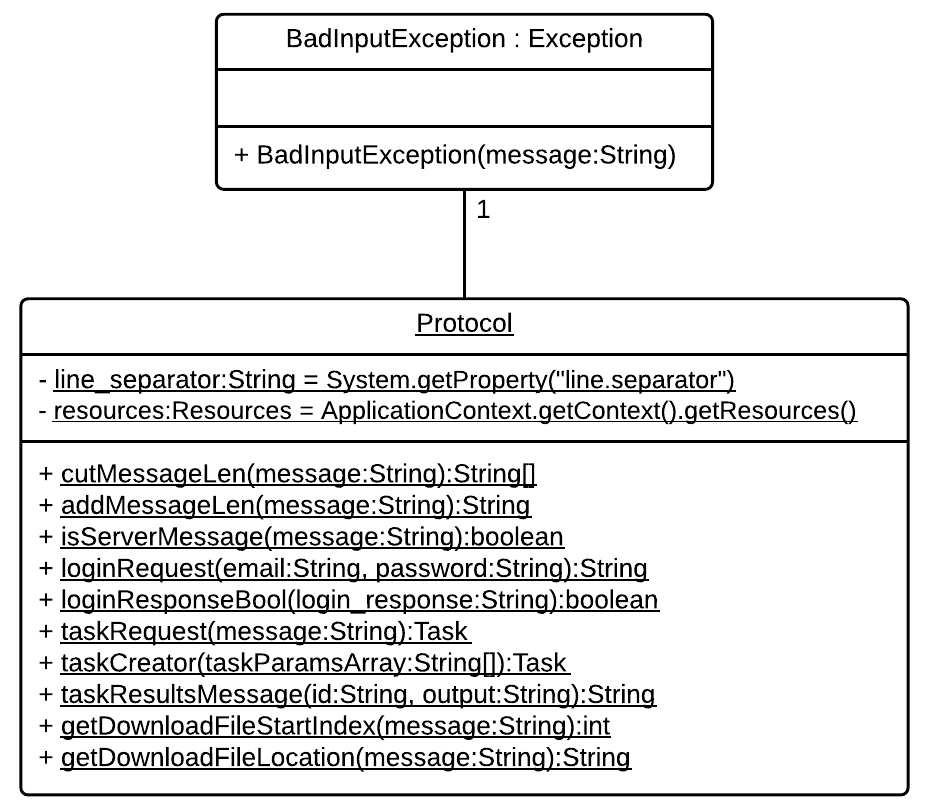
תרשימי UML של שרת ה-Proxy:

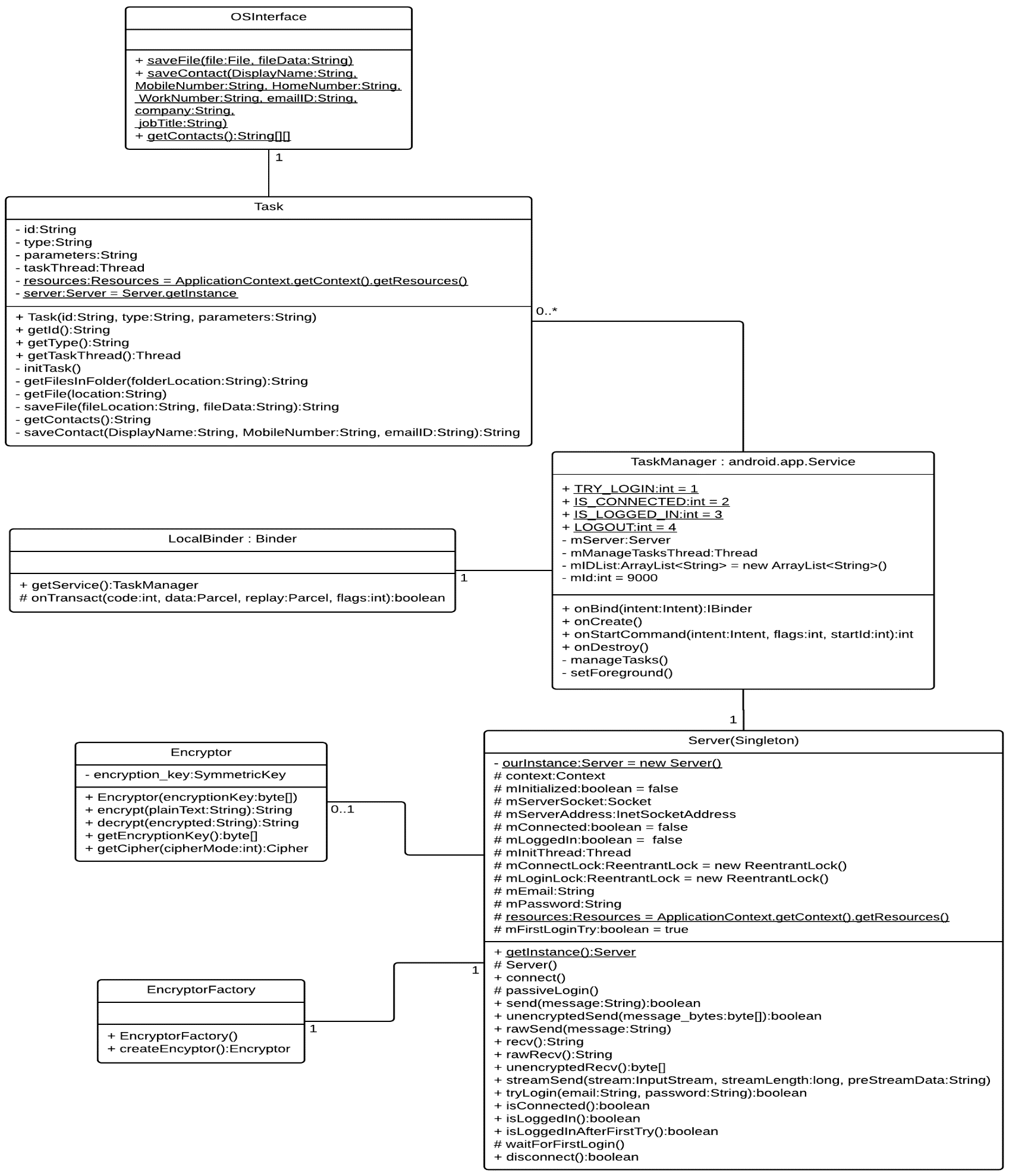


המודולים הם:

* **Manager** –

תרשימי UML של האפליקציה:





המודולים הם:

* MainForm – הטופס הראשי של ה- GUI שמציג את כל ה- Controls.
* PythonListener – אחראי על התקשורת עם מנוע ה- Python.
* ClientControl – Control שמייצג לקוח שהתחבר ומשמש להצגת נתונים על הלקוח והתראות.
* RuleControl – Control שמייצג כלל בתוך הגדרות התוכנה ומשמש לקביעת תכונות הכלל.
* SubRuleControl – Control שמייצג תת-כלל בתוך כלל ומשמש לקביעת תכונות תת-הכלל.
* Rule – מחלקה שמייצגת כלל.
* SubRule – מחלקה שמייצגת תת-כלל.
* ManageWhitelist – טופס שמאפשר לקבוע את ה- whitelist עבור תת-כלל מסוים.
* ManageBlacklist – טופס שמאפשר לקבוע את ה- blacklist עבור תת-כלל מסוים.

# פירוט מבני הנתונים

|  |  |  |  |
| --- | --- | --- | --- |
| **מודול** | **שם המבנה** | **סוג** | **תפקיד** |
| Process\_Stats.py | self.process\_attrs | רשימה של strings | מגדיר את התכונות של תהליך שלגביהן תיתן מידע מחלקת ProcessStats |
| this\_proc\_dict | מילון string-int/string | מילון שהמפתחות שלו הם שמות התכונות של התהליך, והערכים הם הערכים של תכונות אלה בתהליך הספציפי |
| proc\_results\_list | רשימה של מילונים | רשימת כל המילונים this\_proc\_dict עבור כל התהליכים במחשב |
| Client.py | all\_process\_stats | רשימה של מילונים | רשימה שמכילה את כל הנתונים של כל התהליכים במחשב שהלקוח מקבל לאחר קריאה לפונקציית GetStats() של מחלקת ProcessStats |
| Server.py | all\_process\_stats | רשימה של מילונים | אותה רשימה כמו הנ"ל שהשרת מקבל מהלקוח |
| g\_client\_thread\_list | רשימה של אובייקטים | רשימה של כל האובייקטים ממחלקת ClientThread שיצר השרת (כלומר מעין רשימה של כל הלקוחות) |
| Followed\_Action.py | IT\_EMAIL\_ADDRESSES | רשימה של strings | רשימה של כל כתובות המייל של התמיכה הטכנית (IT) של החברה |
| UNITS\_CONVERSION\_DICT | מילון string-int | מילון המסייע להמיר בין היחידות השונות של מידע (Bytes, KB, MB, GB) |
| RESOURCES\_MATCH | רשימה של tuples של strings | רשימה המכילה זוגות של מחרוזות ומתאימה בין שמות "רגילים" של משאבי המחשב לבין השמות שלהם כתכונות של תהליך |
| SYS\_PROCESSES | רשימה של strings | רשימה המכילה שמות של תהליכי מערכת שמהם יש להתעלם בעת ניתוח הנתונים |

# פירוט מאגרי המידע

* **מסד הנתונים:**

התוכנה יוצרת בצד השרת מסד נתונים אחד לכל לקוח. שם הקובץ של המסד מתקבל מהפעלת hash על הכתובת הפיזית של המחשב (שאליה נוסף prefix). במסד הנתונים יש טבלה בודדת בשם **ResourcesData** שמאחסנת את כל הנתונים הנאספים מאותו מחשב.

מבנה הטבלה הוא:

|  |  |  |
| --- | --- | --- |
| **שם העמודה** | **סוג** | **תפקיד** |
| ProcName | VARCHAR(255) | שם התהליך |
| ProcID | INT | מזהה התהליך (ID) |
| CreatingProcID | INT | מזהה התהליך של התהליך שיצר את התהליך הנוכחי |
| ElapsedTime | INT | הזמן שהתהליך רץ בשניות |
| CpuTimePercent | INT | אחוז השימוש ב- CPU |
| WorkingSetPrivate | BIGINT | השימוש בזיכרון (ב- Bytes) |
| IOBytesPersec | BIGINT | כמות ה- Bytes שנקראו או נכתבו בפעולות I/O לשנייה |
| IOOperationsPersec | BIGINT | כמות פעולות I/O לשנייה |
| Threads | INT | כמות תתי-התהליכים |
| Date | DATE | התאריך שבו נאסף המידע |
| Time | TIME | השעה שבה נאסף המידע |

* **קובץ הגדרות בפורמט XML:**

לשם קביעת אופן הפעולה של התוכנה בתרחישים השונים קיים בצד השרת קובץ הגדרות בשם **settings.xml.** את הקובץ קורא המנוע הכתוב ב- Python בכל פעם שהוא מנתח את משאבי המערכת, וה- GUI כותב לקובץ זה בכל פעם שהמנהל משנה את ההגדרות ושומר אותן.

מבנה הקובץ הוא:

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes** | **Tag Description** | **Parent Tag** | **Tag Name** |
| - | The root tag. | - | settings |
| ID, Name, CPUmin, CPUmax, RAMmin, RAMmax, IOmin, IOmax | Defines a new rule. | settings | rule |
| ID, CPUmin, CPUmax, RAMmin, RAMmax, IOmin, IOmax, IOPSmin, IOPSmax, ThreadsMin, ThreadsMax, Action, Type | Defines a new sub-rule of a rule. | rule | subrule |
| - | Defines the whitelist of a sub-rule. | subrule | whitelist |
| - | Defines the blacklist of a sub-rule. | subrule | blacklist |
| name | Defines a process in the whitelist or blacklist. | whitelist  **OR**  blacklist | process |

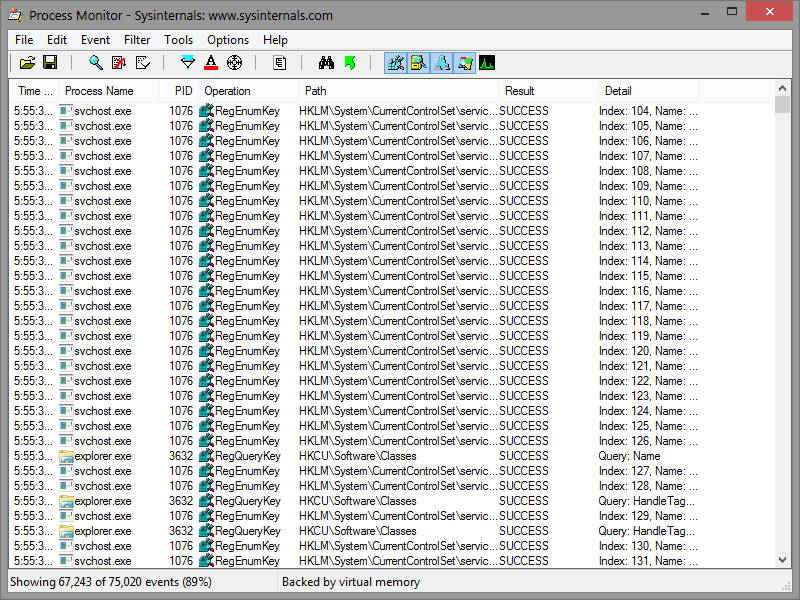
# השוואת העבודה עם פתרונות ויישומים קיימים

בעולם קיימות תוכנות רבות העוסקות במעקב אחר הפעילות המתרחשת במחשב ואחר התהליכים הרצים, ומציגות את המאפיינים שלהם (שימוש במעבד, שימוש בזיכרון, כתיבה וקריאה מהדיסק, חיבור לאינטרנט, תעבורת רשת וכו').

מספר תוכנות מסוג זה שעושות דברים הדומים בחלקם למה שעושה המוצר שלי:

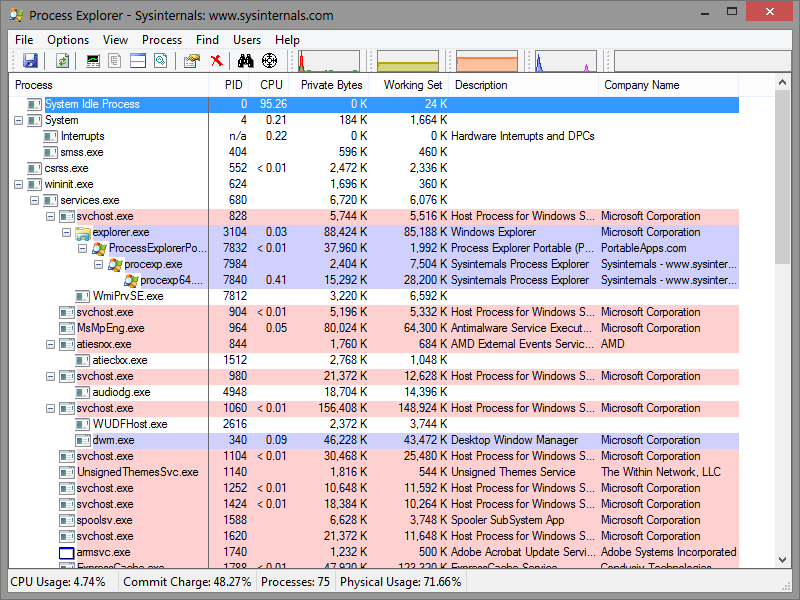
* **Process Monitor:**

זו תוכנה שנוצרה ע"י Sysinternals, שאותה רכשה Microsoft. היא מציגה בזמן אמת את פעילות מערכת המחשב – הפקודות (שמריצים תהליכים שונים) המתרחשות ברגע זה.

מעקב אחר תהליך יכול לשמש לזיהוי ניסיונות כושלים לקרוא ולכתוב מפתחות רישום. זה גם מאפשר סינון על מפתחות ספציפיים, תהליכים, מזהי תהליך וערכים. בנוסף, התוכנה מראה כיצד להשתמש בקבצי EXE וקבצי DLL, מזהה טעויות קריטיות בקבצי מערכת ועוד.

* **Process Explorer:**

זהו מנהל משימות וצג מערכת עבור Windows. גם כלי זה נוצר על ידי Sysinternals. הוא מספק את הפונקציונליות של מנהל המשימות של Windows יחד עם מערך עשיר של תכונות לאיסוף מידע על תהליכים הפועלים במערכת של המשתמש . הדבר יכול לשמש כצעד הראשון בניפוי בעיות תוכנה או מערכת.

Process Explorer יכול לשמש לאיתור בעיות. לדוגמה, הוא מספק אמצעי לחיפוש משאבים שמוחזקים על ידי תהליך. זה יכול לשמש לאיתור של מה שמחזיק את הקובץ פתוח ומונע את השימוש בו על ידי תכנית אחרת. דוגמה נוספת היא היכולת להראות את קווי הפקודה המשמשים להפעלת תכנית, המאפשרת לתהליכים אחרים זהים להיות מכובדים.

**ההבדל העיקרי** בין התוכנות הנ"ל לתוכנה שלי הוא שהתוכנה שלי מאפשרת שליטה מרחוק, ומספקת למנהל הרשת בקרה על המתרחש בכל המחשבים ברשת. בנוסף, קיים ממשק שמאפשר למנהל הרשת להגדיר כללים מדויקים מאוד שקובעים את אופן פעולת התוכנה, ועל פיהם היא מתערבת בנעשה במחשבי הרשת באופן **אוטומטי** כדי למנוע עומס וקריסות וכדי לשמור על נתוני שימוש תקינים.

# מדריך התקנה למשתמש

אילוצים ודרישות מערכת

* התוכנה פותחה עבור מערכת ההפעלה Windows. היא נבדקה ונוסתה בגרסת **Windows 7** בלבד, ועל כן השימוש בה בגרסה אחרת של Windows הוא באחריות המשתמש בלבד.
* בנוסף, יש לוודא שמותקן בכל מחשב קצה ובשרת **Python 2.7 (32-bit)**, ושמותקנות ספריות **PyWin32** ו- **PyCrypto** עבור גרסה זו של Python. בביליוגרפיה נמצאים לינקים להורדת Python ולהתקנת הספריות.
* כמו כן, צריכה להיות מותקנת במחשב השרת **.NET Framework 4.0** או גרסה עדכנית יותר (התוכנה נבדקה בגרסאות **4.0** ו- **4.5**).
* לבסוף, יש לוודא שקיים חיבור בין מחשבי הקצה למחשב עליו יותקן השרת (ובפרט יש לסדר את הגדרות ה- Firewalls שקיימים ברשת הפנימית כך שיאפשרו לתוכנה ליצור חיבורים).

התקנת והרצת התוכנה

* צד שרת:

1. יש להוריד את תיקיית **CRM Server** למחשב השרת.
2. בקובץ config.txt ניתן לקבוע את מספר ה- Port שהשרת יפתח כדי לקבל חיבורים מלקוחות (המספר בשורה הראשונה בקובץ) ואת מספר ה- Port שהשרת יפתח בשביל לתקשר עם ה- GUI (המספר בשורה השנייה בקובץ). ערכי ברירת המחדל הם 50,000 ו- 30,000.
3. כדי להריץ את השרת, יש להריץ את הקובץ **Computer Resources Monitoring.exe** שבתוך תיקיית **CRM Server**. קרוב לוודאי שבהרצה הראשונה תתקבל הודעת אזהרה מ- Windows Firewall ויש לאשר אותה.

* צד לקוח:

1. יש להוריד את תיקיית **CRM Client** לכל אחד ממחשבי הקצה שרוצים להריץ עליהם את התוכנה.
2. בקובץ emails.txt יש להכניס את כל כתובות המייל של התמיכה הטכנית (IT) של החברה. לכתובות מייל אלה התוכנה תשלח מיילים כאשר נבחרת אפשרות "Send Email to IT". כל כתובת מייל צריכה להופיע בשורה חדשה, ללא הוספת תווים נוספים להפרדה (כגון פסיקים).
3. כדי להריץ את התוכנה על מחשבי הקצה, יש ללחוץ לחיצה כפולה על הקובץ **Client.py** שבתוך תיקיית **CRM Client** במחשב זה. גם כאן קרוב לוודאי שבהרצה הראשונה תתקבל הודעת אזהרה מ- Windows Firewall ויש לאשר אותה.

# תיאור הממשק למשתמש

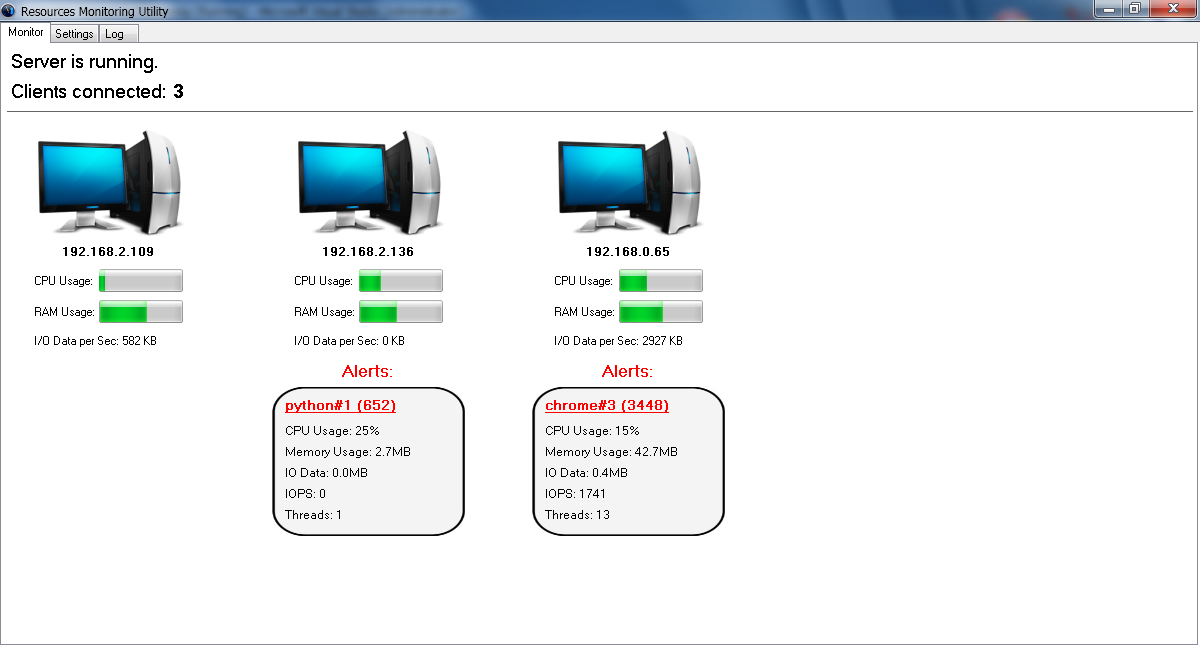
הממשק הגרפי אותו הינך (מנהל הרשת) רואה לאחר הרצת התוכנה מתחלק ל-3 חלקים:

1. ה- **Monitor**.

בחלק זה למעלה יש פס טקסט המתאר את מצבו של ה- Server (אם הוא עלה, רץ או נסגר). שאר המסך מציג את המחשבים שהתחברו לשרת. צבע מסך כחול מעיד על חיבור פעיל ואילו צבע אפור מעיד על מחשב שהתנתק מהשרת.

לכל מחשב מחובר יש כיתוב מתחתיו עם כתובת ה- IP שלו, ומתחתיה מוצגים נתוני השימוש שלו במשאבי מערכת (במעבד, בזיכרון הפיזי ופעולות I/O), אשר מתעדכנים כל פרק זמן קצוב.

מתחת לנתונים אלה מוצגות התראות אשר התוכנה שולחת כאשר היא מזהה פעילות חריגה במחשב הקצה המדובר. ההתראות מכילות את שם התהליך, מזהה תהליך ופירוט של השימוש במשאבים של התהליך. ההתראות העדכניות ביותר תמיד יהיו העליונות ביותר, וכדי לראות התראות ישנות יותר (כאשר מצטברות מספר התראות) יש לגלול עם העכבר למטה.



1. ה- **Settings**.

חלק זה מאפשר לך לצפות בהגדרות התוכנה הנוכחיות ולשנות אותן (לקבוע הגדרות משלך).

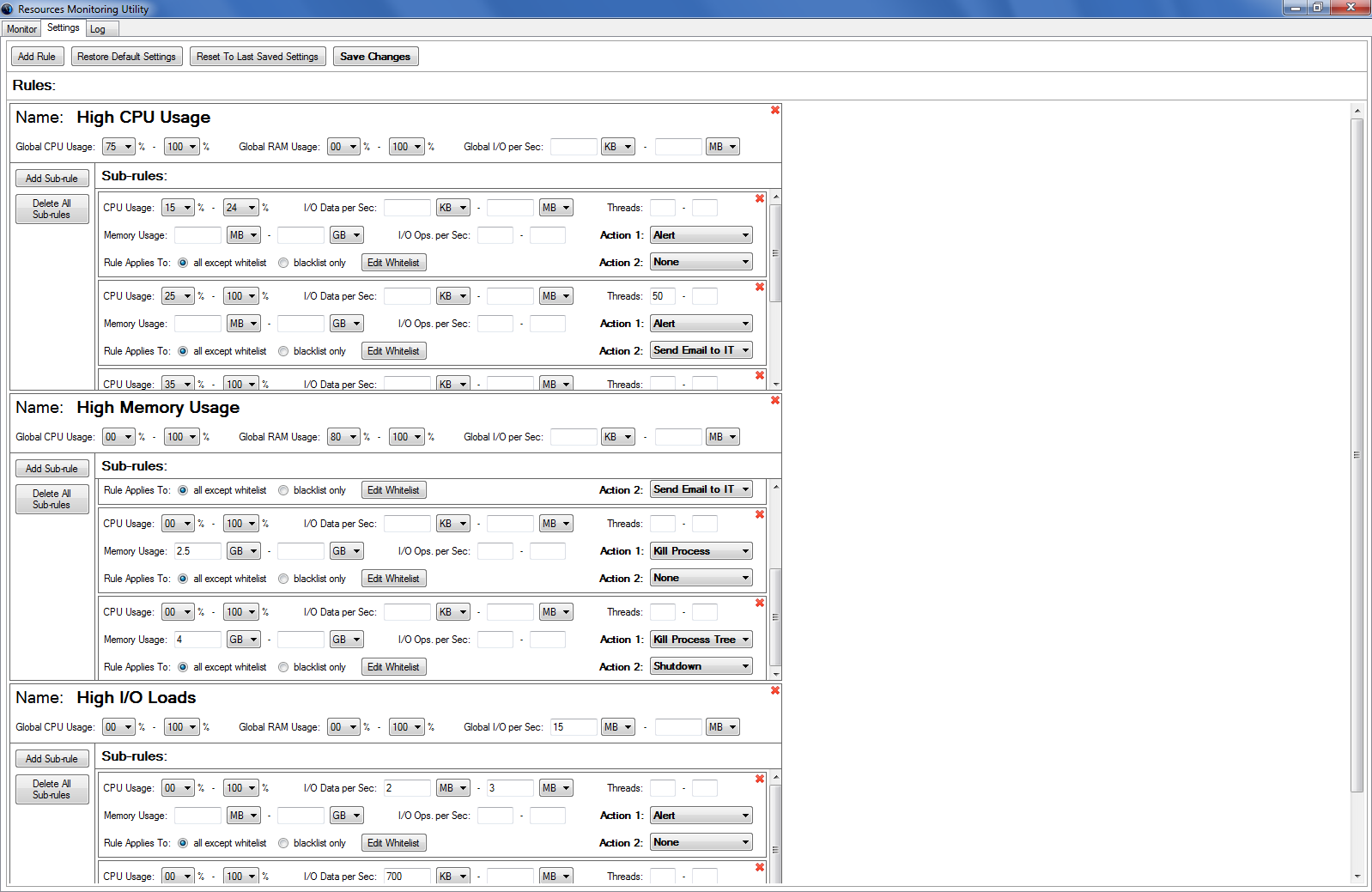
ההגדרות מורכבות מכללים. הכללים קובעים את טווחי השימוש הכולל במשאבי המערכת שלגביהם הם תקפים. למשל, כאשר יש שימוש ביותר מ-25% מזמן המעבד וביותר מ- 3GB של RAM, הכלל נכנס לתוקף.

לכללים יש תתי-כללים, שקובעים את טווחי השימוש במשאבי המערכת של תהליכים אינדיבידואליים שלגביהם הם תקפים. למשל, כאשר תהליך מסוים צורך יותר מ- 10% מזמן המעבד, יותר מ- 300 MB של RAM, ויש לו מעל 50 ת'רדים, אז תת-הכלל נכנס לתוקף (אם הכלל שמכיל אותו בתוקף), ונוקט כלפי תהליך זה בפעולה שנבחרה באופן אוטומטי.

הכפתורים בחלקו העליון של מסך זה מאפשרים להוסיף כלל חדש, להחזיר את ההגדרות לברירת מחדל, להחזיר את ההגדרות למצב האחרון שנשמר, ולשמור את השינויים שבוצעו. יש להקפיד ללחוץ על כפתור שמירת השינויים כדי שכל השינויים בהגדרות יישמרו.

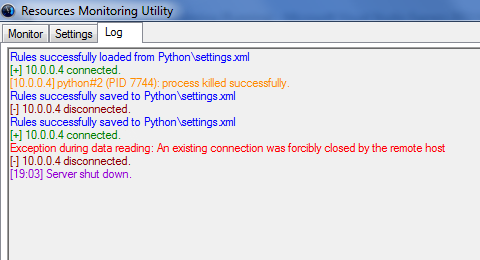
לכל כלל אפשר לתת שם, שבעזרתו יהיה קל יותר לזהות את הכלל ולהבין מה מטרתו הכללית. לכל כלל ניתן לקבוע לגבי אילו טווחים של שימוש במעבד, בזיכרון וב- I/O הוא תקף. הכפתורים בצד שמאל מאפשרים להוסיף תתי-כללים חדשים ולמחוק את כל תתי-הכללים של הכלל. האיקס האדום בפינה הימנית-עליונה מאפשר למחוק את הכלל.

לכל תת-כלל ניתן לקבוע את טווח השימוש במשאבים שלגביו הוא תקף (שימוש במעבד, בזיכרון, כמות I/O Data לשנייה, כמות פעולות I/O לשנייה ומספר Threads). בנוסף, ניתן לקבוע את הפעולות שהתוכנה תבצע כאשר תהליך מקיים את תת-הכלל. כמו כן, ניתן לבחור האם תת-הכלל תקף לכל התהליכים פרט לרשימה מסוימת (whitelist), או שהוא תקף רק לרשימה מסוימת של תהליכים (blacklist), ורשימות אלה כמובן ניתנות לעריכה. גם כאן האיקס האדום בפינה הימנית-עליונה מאפשר למחוק את תת-הכלל.



1. ה- **Log**.

חלק זה מציג תמצית של המתרחש "מאחורי הקלעים". הוא מודיע על התחברות והתנתקות של לקוחות, על שגיאות לא צפויות שהתרחשו (Exceptions), על נפילה של השרת וכו'. זה מאפשר צפייה רטרואקטיבית בפעילות של התוכנה, בכל השלבים והפעולות שננקטו. כמו כן, אם וכאשר מתרחשים דברים לא צפויים, זה עשוי לעזור לאתר את מקור הבעיה.



# מבט אישי על העבודה ותהליך הפיתוח

במהלך העבודה על הפרויקט למדתי לראשונה על ה- WMI ומאוד נהניתי לגלות שקיים ממשק כה נוח לקבלת מידע לגבי רכיבי המערכת. נהניתי ללמוד על האפשרויות המגוונות שהוא מספק ולעבוד איתן. כמו כן, נהניתי לגלות את מודול ה- sqlite3 ב- Python שהיה מאוד נוח ובעזרתו כתבתי בקלות רבה מחלקה לעבודה עם מסד נתונים. גם במהלך העבודה עם קבצי XML הופתעתי לטובה מקלות השימוש בהם, הן מבחינה לוגית והן מבחינה תכנותית.

פיתחתי הרגלי תכנות טובים במהלך העבודה על הפרויקט, הכוללים חלוקה למודולים ותכנון לוגי מראש, תיעוד מלא, שימוש בקבועים, במחלקות ובפונקציות, נתינת שמות משמעותיים למשתנים.

האתגרים והקשיים שנתקלתי בהם במהלך פיתוח הפרויקט הם:

* כאשר הוספתי הצפנה של המידע העובר בין השרת והלקוח, בתחילה בחרתי באלגוריתם האסימטרי El-Gamal, אך גיליתי שייצור של מפתח גדול מספיק הנחשב לבטיחותי (2048 ביטים) לוקח זמן רב מדי. כדי לפתור את הבעיה עברתי לאלגוריתם הנפוץ יותר RSA, שבו זמן ייצור המפתח הוא קצר בהרבה.
* לאחר מכן הבנתי שהצפנה באמצעות מפתח RSA בלבד היא לא ישימה, כי אין דרך בטוחה לתאם בין המפתחות של השרת והלקוח (הדרך היחידה היא להשתמש באותו מפתח קבוע תמיד וזה לא בטיחותי). כדי לפתור את הבעיה מימשתי בפרוטוקול שלי החלפת מפתחות מאובטחת שבמסגרתה השרת שולח ללקוח את מפתח ה- RSA הפומבי שלו והלקוח מצפין בעזרתו את מפתח ה- AES שלו ושולח לשרת. מאותו רגע שני הצדדים משתמשים במפתח ה- AES להצפנה ופיענוח.
* כשעבדתי על המודול שמקבל את נתוני השימוש במשאבים של כל התהליכים, הבדיקות שערכתי הראו שהשימוש הכולל ב- CPU שהתקבל היה מאוד לא מדויק. ערכתי חיפושים באינטרנט ומצאתי מספר דרכים שונות לקבל את הנתון הזה, וניסיתי את כולן כדי לבדוק איזו מהן מניבה את התוצאות הכי מדויקות, ואותה בחרתי בשביל המוצר המוגמר.
* קושי נוסף שהתעורר היה כיצד להחליט איזה נתון של שימוש בזיכרון לקחת בחשבון. ישנם מספר נתוני שימוש בזיכרון של תהליך (Private Bytes, Virtual Bytes, Working Set ועוד) ורציתי לבחור אחד (לשם פשטות) שישקף בצורה המיטבית את השימוש בזיכרון. לאחר מחקר וקריאה של חומר רקע באינטרנט הגעתי למסקנה שהנתון הכי מתאים לצרכים שלי הוא ה- Private Working Set ובו השתמשתי במוצר המוגמר.
* במהלך הבדיקות גיליתי שכאשר מספר מחשבים מתחברים לשרת והשרת מנסה בו-זמנית להכניס לתוך מסד הנתונים את המידע ששולח כל אחד מהם, מתעוררת בעיה, כי לא ניתן לכתוב לאותו קובץ מכמה Threads בו-זמנית. כדי לפתור את הבעיה שיניתי את הקוד כך שלכל מחשב יהיה מסד נתונים משלו בהתאם לכתובת הפיזית שלו.

הערכת הפתרון לעומת התכנון:

* התכנון המקורי היה שהמוצר יוכל לא רק לסגור תהליכים במקרה של זיהוי פעילות חריגה, אלא גם יוכל להגביל את המשאבים העומדים לרשותו של התהליך הבעייתי כצעד מקדים לפני סגירתו. במהלך המחקר גיליתי שהדבר ניתן למימוש רק ע"י עבודה ב- Kernel Mode והדבר כרוך בקשיים רבים. הגעתי למסקנה שכדי לעמוד בזמנים אין לי ברירה אלא לזנוח רעיון זה ולהשאיר רק את האפשרות של סגירת תהליך.
* בתכנון המקורי המוצר היה אמור ללמוד לבד את הרגלי השימוש במשאבי המחשב מהנתונים שהוא אוסף לאורך זמן, ועל סמך זה לזהות חריגויות ולפעול בהתאם. גם כאן בשל מורכבות הדבר (יש לפתח אלגוריתם של לימוד עצמי, מעין בינה מלאכותית) החלטתי לוותר על רעיון זה ולהסתפק במתן אפשרות למנהל הרשת לתת הגדרות מפורטות שעל פיהן יפעל המוצר בכל מקרה ומקרה.
* מלבד הנקודות שהוזכרו לעיל, המוצר המוגמר תואם בדיוק את התכנון המקורי.

ההמלצות לשיפורו הן אם כן:

* להוסיף אפשרות להגביל את השימוש של התהליכים במשאבים.
* להוסיף אלגוריתם לימוד עצמי, שילמד את הרגלי השימוש בכל אחד ממחשבי הרשת באמצעות המידע שנאגר לאורך זמן במסד הנתונים, ויזהה פעילות חריגה בהתאם לכך.
* בנוסף, אין ספק שיש מקום לשיפור המוצר מבחינת אבטחת מידע ומימוש פרוטוקול יותר מקצועי, יעיל ומאובטח.

# 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**