**Your Little Spy**

**תוכנת מעקב ובקרה טלפונים ניידים**

מנחה: דר' עמית דביר

מגישים: ניר עינת ואריאל חדד



תשע"ה

יולי 2015

תוכן עניינים

[הקדמה 3](#_Toc425700808)

[המטרה 4](#_Toc425700809)

[אנו החלטנו לפתח אפליקציה אשר תעניק להורים (אך לא רק, גם מעסיקים, כל גורם אשר ימצא לנכון) אפשרות בקרה וקבלת דוחות שימוש במכשיר הנייד המנוטר עליו מותקנת התוכנה, לקבל סיכום יומי של הודעות SMS ושיחות שבוצעו אל המכשיר המנוטר, במטרה להעניק תמונת מצב לגבי המנוטר, בנוסף נכללות סטטיסטיקות של קבלת SMS יומי ושיחות של אנשי הקשר המנוטרים במכשיר, כל אלו מאפשרים בעצם לקבל תמונת מצב לגבי אופן השימוש ותוכנו של האדם המנוטר. 4](#_Toc425700810)

[מצב קיים 4](#_Toc425700811)

[למה דווקא פיתוח בשפת JAVA IDT 4](#_Toc425700812)

[דרישות מערכת 5](#_Toc425700813)

[הוראות הפעלה 5](#_Toc425700814)

[תיאור המערכת כללי 7](#_Toc425700815)

[רכיבי המערכת 8](#_Toc425700816)

[תוכנת השרת 8](#_Toc425700817)

[אפיון המערכת 8](#_Toc425700818)

[תיאור המערכת בעזרת Use Cases עיקריים 9](#_Toc425700819)

[תיאור הרשמה למערכת 9](#_Toc425700820)

[כניסה לאתר הניהול 10](#_Toc425700821)

[תיאור פלטפורמת הפיתוח 10](#_Toc425700822)

[מחלקות עיקריות 11](#_Toc425700823)

[הטכנולוגיות 23](#_Toc425700824)

[תקשורת בין Client – Server 23](#_Toc425700825)

[הסבר על TCP/IP 23](#_Toc425700826)

[מהו Socket? 23](#_Toc425700827)

[איך זה עובד בתוכנה? 24](#_Toc425700828)

[ממשק גרפי Android ADT (Layout) 24](#_Toc425700829)

[JavaMail API 24](#_Toc425700830)

[MYSQL|PHP API 25](#_Toc425700831)

[בסיס הנתונים 25](#_Toc425700832)

[טבלאות בסיס הנתונים 25](#_Toc425700833)

[טבלת ה- Users 25](#_Toc425700834)

[טבלת parents 26](#_Toc425700835)

[טבלאות clients 26](#_Toc425700836)

[תיאור המסכים 27](#_Toc425700837)

[תוכנת משתמש קצה 27](#_Toc425700838)

[מסך הכניסה 27](#_Toc425700839)

[מסך רישום משתמש חדש 28](#_Toc425700840)

[מסך הראשי של המשתמש 29](#_Toc425700841)

[מסך ההודעות 30](#_Toc425700842)

[מסך שיחות 31](#_Toc425700843)

[מסך אודות 32](#_Toc425700844)

[חלון יצירת קשר 33](#_Toc425700845)

[Literature 35](#_Toc425700846)

[Similar Applications 35](#_Toc425700847)

[Relevant web sites for the project 36](#_Toc425700848)

# הקדמה

מהי רוגלה?

רוגלה : spyware או תוכנת ריגול, היא [תוכנית מחשב](https://he.wikipedia.org/wiki/%D7%AA%D7%95%D7%9B%D7%A0%D7%99%D7%AA_%D7%9E%D7%97%D7%A9%D7%91) העוקבת בחשאי אחר הרגלי הגלישה של המשתמש במחשב שבו היא מותקנת ומעבירה מידע על הרגלים אלו לאתרים ששתלו אותה. הרוגלות נועדו לזהות את העדפותיו של המשתמש ואת תחומי העניין שלו, בדרך כלל כדי להתאים לו [פרסומות](https://he.wikipedia.org/wiki/%D7%A4%D7%A8%D7%A1%D7%95%D7%9D) אישיות בעת הגלישה ולהפיק מהן רווח כלכלי.

הרוגלות מצורפות לתוכנות תמימות למראה (למשל, [יוטורנט](https://he.wikipedia.org/wiki/%D7%A7%D7%90%D7%96%D7%94) – תוכנית [שיתוף קבצים](https://he.wikipedia.org/wiki/%D7%A9%D7%99%D7%AA%D7%95%D7%A3_%D7%A7%D7%91%D7%A6%D7%99%D7%9D) נפוצה) לעתים בעידוד מפיצי התוכנות ולעתים שלא בעידודם. בהסכימו להתקין את ה[תוכנה](https://he.wikipedia.org/wiki/%D7%AA%D7%95%D7%9B%D7%A0%D7%94), מסכים המשתמש גם להתקין הרוגלה, לרוב מבלי להיות מודע לכך (שכן רוב המשתמשים כלל אינם טורחים לקרוא את הסכם השימוש).

תוכנות ריגול שנועדו לרגל ריגול טהור (ולא בהכרח להפיק רווחים) מסוגלות לדעת כמעט כל דבר שביצע משתמש המחשב או לחלופין במקרה שלנו הסמארטפון, החל מהרגלי הגלישה שלו, וכלה במידע אישי כמו יומן או אף סיסמאות או נתונים על חשבון הבנק שלו שהקליד. דוגמאות לתוכנות כאלו הן [סוס טרויאני](https://he.wikipedia.org/wiki/%D7%A1%D7%95%D7%A1_%D7%98%D7%A8%D7%95%D7%99%D7%90%D7%A0%D7%99_(%D7%AA%D7%95%D7%9B%D7%A0%D7%94)) או תוכנת key log הרושמת בקובץ את כל המקשים שעליהם הקיש המשתמש. לכן יש להיזהר על אחת כמה וכמה משימוש רגיש במחשבים ציבוריים כמו [קפה אינטרנט](https://he.wikipedia.org/wiki/%D7%A7%D7%A4%D7%94_%D7%90%D7%99%D7%A0%D7%98%D7%A8%D7%A0%D7%98).

הרוגלה מתגלה לעתים עקב המטרד שהיא גורמת, בדמות חלונות פרסומת שקופצים תוך כדי הגלישה ואשר מקורם אינו באתר שאליו פונה הגולש. בשוק המחשבים פותחו תוכנות לנקות את המחשב מרוגלות שמותקנות בו. משתמשים מתקדמים נוהגים להתקין במחשביהם תוכנת [רחרחן (סניפר)](https://he.wikipedia.org/wiki/%D7%A8%D7%97%D7%A8%D7%97%D7%9F), המסוגלת לעקוב אחר תעבורת הרשת במחשב ולרשום מידע זה בקובץ. בדרך זו אפשר לזהות אם תוכנה שהותקנה זה עתה היא באמת תוכנה תמימה או שמא רוגלה, וכן אם קיימת במחשב פעילות של סוס טרויאני.

כמו כן, בהתאם להתעצמות הטכנולוגיה בשוק המכשירים הניידים החכמים התפתחו רוגלות בקצב מסחרר גם למכשירי הסמארטפון כמו במחשבים, שכן אנו שומרים את אותו המידע אם לא רגיש יותר במכשיר הנייד האישי שלנו.

# המטרה

# אנו החלטנו לפתח אפליקציה אשר תעניק להורים (אך לא רק, גם מעסיקים, כל גורם אשר ימצא לנכון) אפשרות בקרה וקבלת דוחות שימוש במכשיר הנייד המנוטר עליו מותקנת התוכנה, לקבל סיכום יומי של הודעות SMS ושיחות שבוצעו אל המכשיר המנוטר, במטרה להעניק תמונת מצב לגבי המנוטר, בנוסף נכללות סטטיסטיקות של קבלת SMS יומי ושיחות של אנשי הקשר המנוטרים במכשיר, כל אלו מאפשרים בעצם לקבל תמונת מצב לגבי אופן השימוש ותוכנו של האדם המנוטר.

# מצב קיים

המצב הקיים היום בשוק הינו מצב של תוכנות רבות לא מאושרות אשר מציעות ממשק לא נוח, תוכנות זוללות משאבים ולא מנוהלות נכון, וכמו כן פרוצות לחלוטין אשר מספקות שירות דומה של ריגול אחר המכשיר המנוטר ומתן לוגים בסיסיים למשתמש.

# למה דווקא פיתוח בשפת JAVA IDT

כיום בשוק ישנם 2 ספקים מרכזיים בעולם הסמארטפונים אשר מחזיקים ב 2 מערכות ההפעלה הנפוצות ביותר למכשירים אלו והן:

* APPLE – אשר מריצה את מערכת ה IOS (מערכת סגורה)
* GOOGLE – אשר מריצה את מערכת ה ANDROID (מערכת פתוחה עם שלל API'ים חינמיים ועוד).

אנו בחרנו לפתח את האפליקציה שלנו בשפת JAVA ל ANDROID מהסיבה הפשוטה שברוב התואר תכנתנו בשפת JAVA, ובנוסף למדנו באוניברסיטה קורס פיתוח אפליקציות ב ANDROID אשר העניק לנו בסיס והקדמה לשפה.

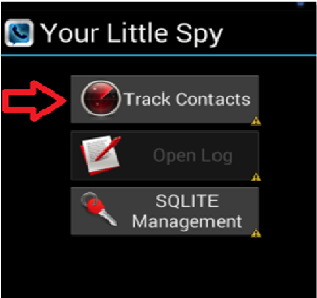
# דרישות מערכת

כדי להפעיל את אפליקציית משתמש הקצה ותוכנה השרת יש צורך במספר דרישות בסיסיות לסוגי הפלטפורמות השונות:

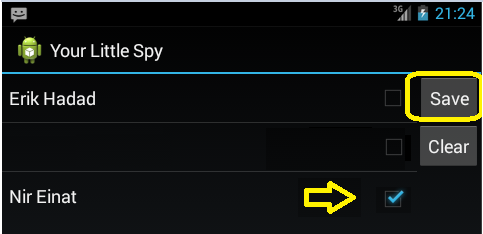
* מכשיר אנדרואיד בגרסא 4.2 ומעלה (JELLY BEAN)
* גישה לאינטרנט דרך כל מחשב PC או MAC. (על מנת לצפות בדוחות בצורה נוחה)

# הוראות הפעלה

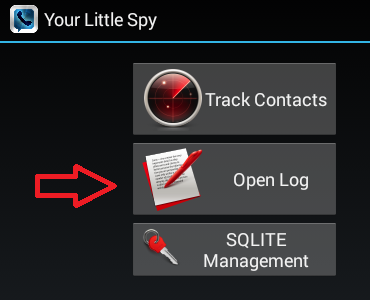
1. הממשק של האפליקציה .כאשר נפתח לנו האפליקציה המסך הראשי מציג לנו אופציה לסמן את אנשי הקשר המנוטרים במכשיר.



במקרה הנ"ל אנו בוחרים את שם איש הקשר המופיע מטה: אותו אנו רוצים לנטר , ומסמנים אותו. ולאחר מכן אנו לוחצים על שמור.

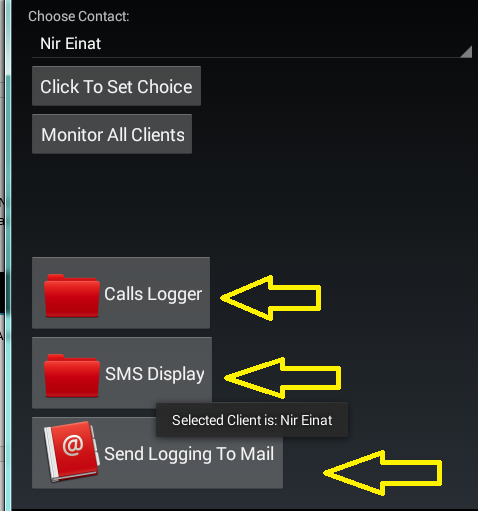


1. אחרי ששמרנו את השם שאותו אנו רוצים לנטר, ניתן לחזור אחורה למסך הראשי , וניגש למסך האפשרויות הראשי Open Log:



1. במסך הבא אנו נקבל שליטה מלאה על אנשי הקשר המנוטרים, ונוכל להציג כלוג את השיחות הנקלטות או / ו את ה SMS'ים הנשלחו למכשיר המנוטר.

בנוסף: נוכל לבצע שליחה ידנית לכתובת מייל ספציפית שנגדיר (שימוש ב INTENT קיים עבור MAIL CLIENT שמוגדר כברירת מחדל על מנת לשלוח אליו את הנתונים. (מדובר ב METHOD אשר שולח את כלל הנתונים כסטרינג אחיד.)



# תיאור המערכת כללי

המערכת שלנו עובדת על מערכת הפעלה של אנדרואיד שהיא מיועדת לסמארטפונים

היא מבוססת על ליבת Linux, שפת הפיתוח שאנו מפתחים בה היא java.

המערכת בנויה מ 2 רכיבי Database והם:

* Online Database (SQLITE אשר בשימוש של מכשיר האנדרואיד ואליו נכתבים הנתונים)
* OFFLINE Database (שהוא השרת שלנו העובד עם MYSQL SERVER)

תרשים המתאר את אופן הפעולה:



# רכיבי המערכת

## תוכנת השרת

לתוכנת השרת מספר תפקידים חשובים במערכת, כגון:

* מתווכת בין המשתמשים ושולחת את ההודעות בניהם.
* מנהלת את בסיס הנתונים עליו שמורים נתוני המשתמשים כולל רשימות החברים.
* אחראית על מענה לבקשות המשתמשים, למשל:
* הצגת נתונים של משתמש אחר
* הוספת חשבון (הורה) מנטר
* ועוד...

לשרת יש חשיבות רבה מאוד בתפעול מערכת אשר בלעדיה לא היה ניתן לנהל את מערך המידע והלוגים.

# אפיון המערכת

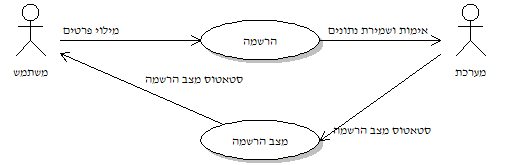
המערכת כוללת מספר רכיבים עיקריים:

* ניהול חשבון משתמש – רכיב זה מאפשר למשתמש לנהל את חשבונו האישי כולל את פרטיו האישיים, כמו כן, ניתן גם באמצעות רכיב זה לעדכן את הפרטים ולשנות את הסיסמא האישית.
* הרשמה למערכת – רכיב אשר בו המשתמש נרשם למערכת ובו המשתמש ממלא את פרטיו האישיים (תעודת זהות) את שם המשתמש הייחודי לצורך הכניסה ואת סיסמתו האישית..
* מסכי לוגים – בהם יוצג כל המידע הרלוונטי על המכשיר המנוטר.

# תיאור המערכת בעזרת Use Cases עיקריים

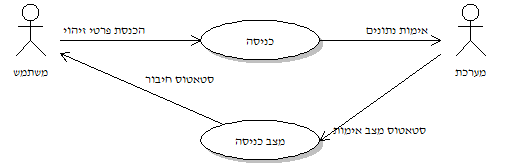
## תיאור הרשמה למערכת

1. המשתמש ממלא את פרטי תעודת הזהות שלו + סיסמא + סלולארי של המשתמש המנוטר
2. המערכת מבצעת בדיקת תקינות הנתונים ואימות מול מסד הנתונים.
3. המשתמש מקבל הודעה על הצלחה או כישלון ההרשמה למערכת.



## כניסה לאתר הניהול

1. המשתמש מכניס את שם המשתמש שהוא תעודת זהות ואת הסיסמא.
2. המערכת מבצעת בדיקת תקינות הנתונים ואימות מול מסד הנתונים.
3. כניסה המשתמש למערכת או קבלת הודעה על כישלון האימות.



# תיאור פלטפורמת הפיתוח

קיימות מספר בעיות אנו נפרט את המרכזיות בהן:

* אילוץ האפליקציה לעבוד ברקע של המנוטר בכל זמן נתון: - משמעות הדבר שנירצה שבכל עת מרגע הפעלת האפליקציה יהיה ניטור שוטף של ההודעות והשיחות.
* שמירת כלל הנתונים על אנשי הקשר גם בסגירת האפליקציה: כאשר האפליקציה סגורה או לחלופין בוצע RESTART למכשיר הסלולארי אנו לא רוצים לאבד את המידע. (שימוש ב DB המקומי של המכשיר SQLITE לאכסון הנתונים וטעינתם בכל פעם שהאפליקציה עולה.

וכמו כן, הפרויקט פותח תחת סביבת העבודה Android Studio של חברת Google אשר סביבה זאת מבוססת על קוד פתוח ותומכת בפיתוח על גבי פלטפורמות שונות ובשפות תכנות מגוונות וכוללת בתוכה ממשק ידידותי לעיצוב מסכי ממשק המשתמש.

בנוסף אנו משתמשים בתוכנת ה- WAMP אשר מבוססת על קהילת קוד פתוח ומאפשרת בקלות רבה ליצור שרת מסדי נתונים מסוג MySql וכמו כן גם PHP.

# מחלקות עיקריות

מחלקת CLIENT נועדה בעצם ליצור משתמש אשר יכיל את הפרטים המזהים שלו כמו כן יכיל את הנתונים אשר יחזיק את המידע הרלוונטי לגביו ה DATA.

בנוסף מספר מתודות אשר השתמשנו בהן.

**package** com.example.broadcastreciever;

/\*\*

\* The core client class which has all the needed params for a well working monitored client :).

\* **@author** Niro

\*

\*/

**public** **class** Client

{

String Name;

String Number;

String Data;

String SMS\_Data;

String no;

**int** Monitored;

**int** SMS\_Sum;

**int** Calls\_Sum;

**public** Client(String Name,String Number)

{

**this**.Name=Name;

**this**.Number=Number;

**this**.Data="";

**this**.SMS\_Data = "";

**this**.Monitored=0;

**this**.SMS\_Sum=0;

**this**.Calls\_Sum=0;

**this**.no="no";

}

**public** Client()

{

**this**.Name="";

**this**.Number="";

**this**.Data="";

**this**.no="";

}

**public** **void** setMonitored(**int** num)

{

**this**.Monitored = num;

}

**public** **void** setStatus(String no)

{

**this**.no = no;

}

**public** **void** setName(String Name)

{

**this**.Name=Name;

}

**public** **void** setNumber(String Number)

{

**this**.Number=Number;

}

**public** String getNumber()

{

**return** **this**.Number;

}

**public** String getName()

{

**return** **this**.Name;

}

**public** String getStatus()

{

**return** **this**.no;

}

**public** **void** setData(String string)

{

**this**.Data += string+"\n\n";

}

**public** String getData()

{

**return** **this**.Data;

}

**public** String getSMSData()

{

**return** **this**.SMS\_Data;

}

**public** String toString()

{

**return** **this**.getName();

}

**public** **void** setSMSData(String data)

{

**this**.SMS\_Data += data+"\n\n";

}

**public** **int** getMonitored() {

// **TODO** Auto-generated method stub

**return** **this**.Monitored;

}

**public** String printAllClientDetails()

{

**return** **this**.getName()+", "+**this**.getNumber()+", "+**this**.getData()+", "+**this**.getSMSData()+","

+ " "+**this**.getMonitored()+ ", "+**this**.getStatus();

}

**public** **void** setSMS\_Sum()

{

**this**.SMS\_Sum++;

}

**public** **void** setSMS\_Sum(**int** num)

{

**this**.SMS\_Sum = num;

}

**public** **int** getSMS\_Sum()

{

**return** **this**.SMS\_Sum;

}

**public** **void** setCalls\_Sum()

{

**this**.Calls\_Sum++;

}

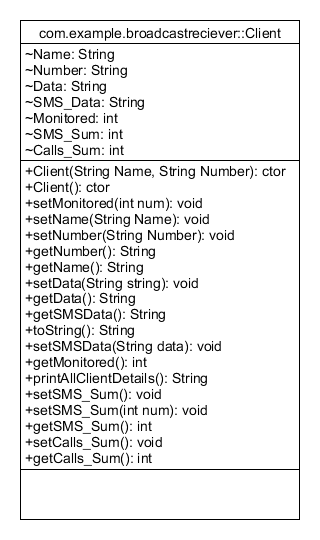
**public** **int** getCalls\_Sum()

{

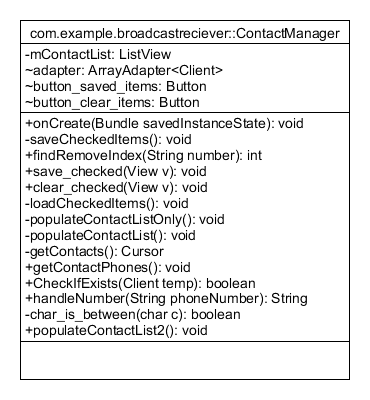
**return** **this**.Calls\_Sum;

}

}



מחלקת CONTACT MANAGER אשר אחראית על יצירת רשימת אנשי הקשר אשר על המכשיר המנוטר ובנוסף אחראית ליצור קשר שוטף עם ה DB על מנת לעדכן את הנתונים עדכנית.



מחלקת MAIN DATA אחראית על יצירת 2 מערכי רשימות (KEY VALUE) שה VALUE הוא בעצם אובייקט מידע.

המחלקה משתמש בתבנית העיצוב של סינגלטון על מנת לוודא יצירת מנוטר אחד בלבד ובנוסף אחראית על בניית מבני הנתונים ואכלוסו.

כמו כן קיימות שיטות אשר המחלקה משתמשת בהן ומתוארות מטה.

\* This Class creates the Main\_Data (2 Array Lists which hold all the information regarding Monitored Clients and general clients USING the SINGLETON PATTERN)

\* **@author** Niro

**public** **class** Main\_Data

{

**private** **static** Main\_Data *instance*=**null**;

**public** ArrayList<Client> data = **new** ArrayList<Client>();

**public** ArrayList<Client> Monitored\_Clients = **new** ArrayList<Client>();

**private** Main\_Data(){}

**public** **static** Main\_Data getInstance()

{

**if** (*instance* == **null**)

{

*instance* = **new** Main\_Data();

**return** *instance*;

}

**else**

{

**return** *instance*;

}

}

**public** **void** setData(Client v, String phoneNumber)

{

}

**public** String getArrayList()

{

String s="";

**for** (Client v : **this**.data)

{

s+=v.getName()+" Number: "+v.getNumber()+" Data: "+v.getData()+"\n";

}

**return** s;

}

**public** Client getClient(String phoneNumber)

{

// for (Client v : Main\_Data.getInstance().Monitored\_Clients)

// {

// if (phoneNumber.equals(v.getNumber())==true) //OLD Code

// {

// return v;

// }

// }

**for** (Client v : Main\_Data.*getInstance*().Monitored\_Clients) //New CODE

{

**if** (v.getNumber().contains(phoneNumber)==**true**)

{

**return** v;

}

}

**return** **null**;

}

**public** Client getClientbyName(String Name)

{

**for** (Client v : Main\_Data.*getInstance*().Monitored\_Clients)

{

**if** (Name.equals(v.getName())==**true**)

{

**return** v;

}

}

**return** **null**;

}

**public** **boolean** numinList(String phoneNumber)

{

**boolean** ans = **false**;

**for** (Client v : Main\_Data.*getInstance*().Monitored\_Clients)

{

// if (phoneNumber.equals(v.getNumber())==true)

// {

// return true;

// }

Log.*d*("MYDEBUG", "phoneNumber is: "+phoneNumber);

Log.*d*("MYDEBUG", "client saved num is: "+v.getNumber());

Log.*d*("MYDEBUG", "phone contains client phone is: "+phoneNumber.contains(v.getNumber()));

**if**(phoneNumber.contains(v.getNumber())==**true**)

{

**return** **true**;

}

}

**return** ans;

}

**public** **boolean** numinListForSMS(String phoneNumber)

{

**boolean** ans = **false**;

**for** (Client v : Main\_Data.*getInstance*().Monitored\_Clients)

{

// if (phoneNumber.equals(v.getNumber())==true)

// {

// return true;

// }

String newPhoneNumber = phoneNumber.substring(4,phoneNumber.length());

Log.*d*("MYDEBUG", "newPhoneNumber is: "+newPhoneNumber);

Log.*d*("MYDEBUG", "client saved num is: "+v.getNumber());

Log.*d*("MYDEBUG", "phone contains client phone is: "+newPhoneNumber.contains(v.getNumber()));

**if**(v.getNumber().contains(newPhoneNumber)==**true**)

{

**return** **true**;

}

}

**return** ans;

}

**public** String getLogging() //Will return all calling logs of monitored clients!

{

String data = "";

**for** (Client v : **this**.Monitored\_Clients)

{

data+=v.Data+"\n";

}

**return** data;

}

**public** ArrayList<String> getListNames()

{

ArrayList<String> ans = **new** ArrayList<String>();

**for** (Client v : data)

{

**if** (v!=**null**)

{

ans.add(v.getName());

}

}

**return** ans;

}

**public** ArrayList<String> getMonitoredListNames()

{

ArrayList<String> ans = **new** ArrayList<String>();

**for** (Client v : Monitored\_Clients)

{

**if** (v.getMonitored()==1)

{

ans.add(v.getName());

}

}

**return** ans;

}

**public** Client [] getContactArray() //Returns String array for populating the list

{

Client [] ans = **new** Client[data.size()];

**int** i=0;

**for** (Client v : data)

{

ans[i] = v;

i++;

}

**return** ans;

}

**public** **boolean** isExists(Client other)

{

**boolean** ans = **false**;

**for** (Client v : Monitored\_Clients)

{

**if** (v.Number.equals(other.Number)==**true**)

{

ans = **true**;

**break**;

}

}

**return** ans;

}

**public** String getAllMonitoredData()

{

String data="";

**for** (Client v : Monitored\_Clients)

{

data +=v.getData() + v.getSMSData() + "Number of Total SMS Messages:" + v.getSMS\_Sum();

}

data += "Total Number of monitored clients are: "+**this**.Monitored\_Clients.size();

data += "\nTotal Number of SMS captured: +"+getTotal\_SMS();

data += "\nTotal Number of Calls captured: +"+getTotal\_Calls();

**return** data;

}

**public** **void** clearAll()

{

**for** (Client v : Monitored\_Clients)

{

v.setMonitored(0);

}

}

**public** String printMonitored()

{

String s="";

**for** (Client v : Monitored\_Clients)

{

s+=(v.getName()+":"+v.getMonitored());

}

**return** s;

}

**public** **void** setMonitored\_Clients(ArrayList<Client> other)

{

**this**.Monitored\_Clients = other;

}

**public** **int** getTotal\_SMS()

{

**int** ans=0;

**for** (Client v : **this**.Monitored\_Clients)

{

ans+=v.getSMS\_Sum();

}

**return** ans;

}

**public** **int** getTotal\_Calls()

{

**int** ans=0;

**for** (Client v : **this**.Monitored\_Clients)

{

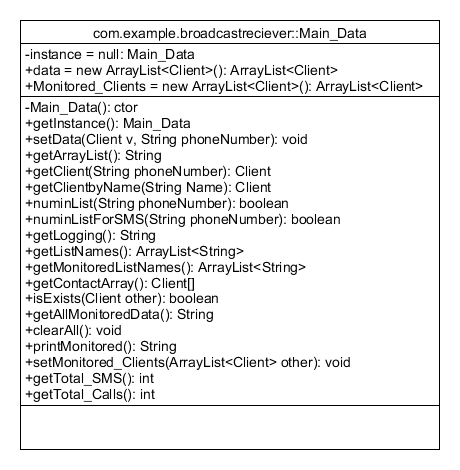
ans+=v.getCalls\_Sum();

}

**return** ans;

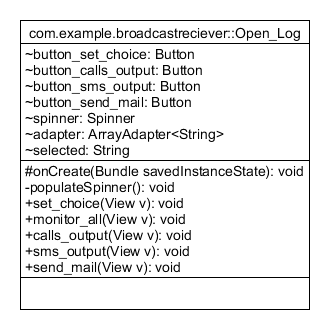
}

}



מחלקת OPEN LOG אחראית על הצגה של כלל הלוגים במערכת על פי בחירת המשתמש באיש הקשר שבו הוא חפץ.

המחלקה היא בעיקר הצגת נתונים וכלל המתודות שלנו הם הדפסות.



package com.example.broadcastreciever;

import android.os.Bundle;

import java.io.Serializable;

import android.app.Activity;

import android.content.Intent;

import android.util.Log;

import android.view.View;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.Spinner;

import android.widget.Toast;

/\*\*

\* The main UI for the user to handle and see the logging.

\* @author Niro

\*

\*/

public class Open\_Log extends Activity {

/\*\*

\*

\*/

Button button\_set\_choice;

Button button\_calls\_output;

Button button\_sms\_output;

Button button\_send\_mail;

Spinner spinner;

ArrayAdapter<String> adapter;

String selected;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_open\_\_log);

spinner = (Spinner) findViewById(R.id.spinner);

button\_calls\_output = (Button) findViewById(R.id.button\_calls);

button\_sms\_output = (Button) findViewById(R.id.button\_sms);

button\_set\_choice = (Button) findViewById(R.id.button\_set\_choice);

button\_send\_mail = (Button) findViewById(R.id.button\_mail);

try

{

populateSpinner();

}

catch(Exception e)

{

Log.w("MYDEBUG", e);

}

}

private void populateSpinner()

{

adapter = new ArrayAdapter<String>(this, android.R.layout.simple\_spinner\_item,Main\_Data.getInstance().getMonitoredListNames());

spinner.setAdapter(adapter);

}

public void set\_choice(View v)

{

selected = spinner.getSelectedItem().toString();

Log.w("MYDEBUG", "Selected Client is: "+selected);

Toast.makeText(this, "Selected Client is: "+selected, Toast.LENGTH\_LONG).show();

button\_calls\_output.setEnabled(true);

button\_sms\_output.setEnabled(true);

button\_send\_mail.setEnabled(true);

}

public void monitor\_all(View v)

{

Intent i = new Intent(this,MonitorAll\_Output.class);

i.putExtra("All\_Clients", selected);

startActivity(i);

}

public void calls\_output(View v)

{

Intent i = new Intent(this,Calls\_Output.class);

i.putExtra("Client", selected);

startActivity(i);

}

public void sms\_output(View v)

{

Intent i = new Intent(this,SMS\_Output.class);

i.putExtra("Client", selected);

startActivity(i);

}

public void send\_mail(View v)

{

Client c = Main\_Data.getInstance().getClientbyName(selected);

Intent intent = new Intent(Intent.ACTION\_SEND);

intent.setType("text/html");

intent.putExtra(Intent.EXTRA\_EMAIL, "nire@bezeqint.co.il");

intent.putExtra(Intent.EXTRA\_SUBJECT, "Logging from Sniffer");

intent.putExtra(Intent.EXTRA\_TEXT, c.getData()+"\n"+c.getSMSData());

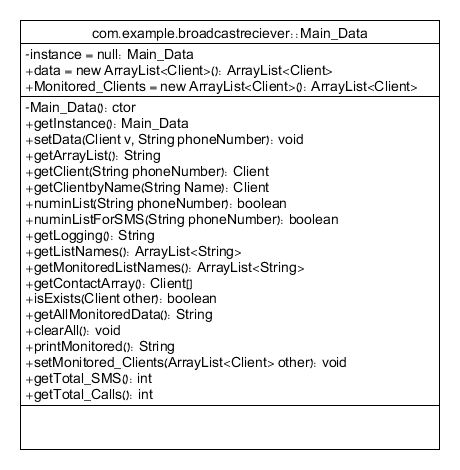
startActivity(Intent.createChooser(intent, "Send Email"));

Toast.makeText(this, "Email Sent!", Toast.LENGTH\_LONG).show();

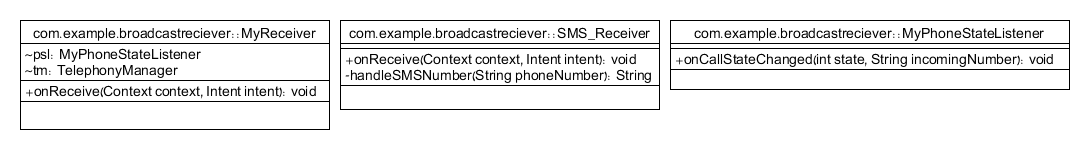
}

}

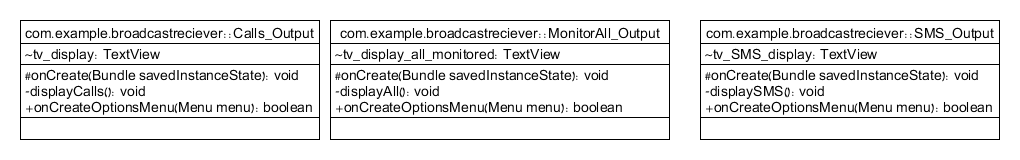
מחלקת Main Data בעצם מכילה מתודות עזר בעיקר להשגת נתונים ממערך האובייקטים של האנשים המנוטרים ובכלל, מספר שיטות GET ושיטות ניקוי המערכים והדפסות.



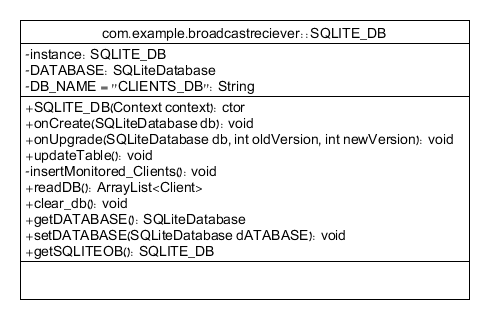
מחלקת Broadcast Receiver אחראית על לכידת הארועים בתוכנית, בעצם כאן מאתחלים את התוכנית להאזנה של ארועים במכשיר הנייד.

7 

מחלקות הדפסה:



מחלקת ניהול ויצירת ה Database:



# הטכנולוגיות

## תקשורת בין Client – Server

### הסבר על TCP/IP

זהו פרוטוקול תקשורת, או משפחה של פרוטוקולים המאפשרים חיבור של מספר רב של רשתות מחשבים כולל רשתות קטנות ופנימיות ועד לרשתות גדולות ביותר כגון רשת האינטרנט.

פרוטוקול זה מאפשר להתגבר על מגבלות טכנולוגיות וליצור רשתות מחשבים בטכנולוגיות שונות של תשתית.

דוגמא טובה לכך היא התשתיות השונות הקיימות, תשתית טלפונית כגון "בזק" או תשתית כבלים כגון "הוט" אשר כל אחת עובדת בטכנולוגיה שונה אבל המחשבים מתקשרים באותו פרוטוקול – TCP/IP.

למודל הפרוטוקולים TCP/IP יש ארבע שכבות שתפקידן ליצור את התקשורת, השכבות הן:

1. **Application (שכבת היישום)** - פרוטוקול התקשורת שמופעל על ידי תוכנת הקצה, עיבוד נתונים, ניהול תהליכים.
2. **Transport (שכבת התעבורה)** - שמירה על סדר הנתונים ובקרת זרימה. בניגוד לפרוטוקול TCP, המבטיח גם כי כל הנתונים שנשלחו מצד אחד יגיעו אל הצד השני, שכבת התעבורה של המודל לא מתחייבת לספק שירות זה. למשל פרוטוקול UDP, המממש שכבה זו, לא מוודא הגעה של הנתונים.
3. **Network (שכבת הרשת)** - העברת נתונים על פי כתובות לוגיות (IP).
4. **Physical (השכבה הפיזית)** - העברת אותות בינאריים, וחלוקת נתונים על פי כתובות פיזיות.

### מהו Socket?

Socket זה נקודת הקצה בצד אחד של חיבור בין שני יישומים שרצים ברשת.

Socket מקושר למספר Port כך ששכבת ה- TCP תוכל לזהות את היישום שהמידע מיועד אליו.

מחלקות Socket משמשות לייצג את החיבור בין יישום המשתמש קצה ליישום השרת.

חבילת ה Java.net מספקת שני מחלקות - Socket ו Server Socket - שיכולות לממש את החיבור מצד המשתמש קצה ומצד השרת בהתאמה.

### איך זה עובד בתוכנה?

***צד השרת:*** על השרת מוגדר Server Socket אשר מאזין ל- Port המוגדר לצורך חיבור תוכנת משתמש הקצה, תפקידו של ה- Socket הוא להאזין לחיבורים חדשים שמתחברים לשרת ומנותבים ל-Thread חדש אשר בו מתבצעת האזנה לבקשות/הודעות ממשתמשי הקצה על ידי ה- Socket שמקשר בין המשתמש קצה לשרת.

***צד משתמש הקצה:*** על המשתמש קצה מוגדר Socket אשר מוגדרים בו כתובת ה- IP של השרת והפורט בו השרת מאזין. בזמן הכניסה לתוכנה/אפליקציה מתבצע התחברות לשרת דרך ה Socket המוגדר.

כל השליחות וקבלת הבקשות/הודעות מתבצעות דרך אותו Socket שנפתח בזמן ההתחברות לשרת.

במקרה ומתרחש ניתוק מהשרת, יש אפשרות למשתמש קצה ליצור Socket חדש לצורך התחברות מחדש לשרת ע"י לחיצה על הכפתור המתאים המוגדר לשם כך.

## ממשק גרפי Android ADT (Layout)

ADT מכילה ערכת כלים גרפיים של רכיבים שימושיים לתוכנה כגון: כפתורים, רשימות, פס גלילה, תוויות טקסט ועוד...

כמו כן, רכיבים אלו אינם מוגבלים למכשיר (מדובר על אנדרואיד בלבד) וניתן להשתמש בהם גם כאשר מפעילים את האפליקציה בגרסאות שונות.

בנוסף לרכיבים אלו ניתן לבצע התאמה אישית לצרכים של המפתח ע"י ביצוע הורשה מהרכיב עצמו וביצוע השינוי הרצוי ע"י כתיבת קוד.

## JavaMail API

על מנת לאפשר למשתמשים לשחזר את הסיסמה המוגדרת בבסיס הנתונים שבשרת, נעשה שימוש ברכיב INTENT ייעודי אשר אחראי על יצירת מייל מהאפליקציה אליו אנו מעבירים נתונים בתצורה של Intent.PutExtra שהוא באפר שמחזיק מידע ועובר דרך האובייקט.

בעצם זוהי דרך קלה ונוחה לשליחת דוא"ל.

## MYSQL|PHP API

על מנת לקשר את הסמארפטון ל DB חיצוני אשר יכיל את כל המידע ללא תלות וקשר למכשיר הסלולרי היינו צריכים "מתווך" ובמקרה הזה השתמשנו ב PHP ובשרת ה WAMP.

בעזרת כלי זה יצרנו סקריפטים של התחברות מהאפליקציה אל השרת שלנו שמריץ DATABASE של MYSQL, בעזרת סקריפטים של PHP שהטמענו בעצם בנינו אוייבקטים של JSON אשר האפליקציה שולחת REQUEST ומקבלת חזרה RESPONSE עם המידע הרלוונטי.

בעזרת ה PHP הצלחנו לתווך בין ה CLIENT ל SERVER.

# בסיס הנתונים

בסיס הנתונים שעל תוכנת השרת בנוי על MySql שהוא בסיס נתונים המבוסס "קוד פתוח" בבעלות החברה Oracle אשר ניתן להורדה מהאינטרנט ושימוש בחינם. בסיס נתונים זה מבוסס על שפת SQL (Structured Query Language).

תפקידו של בסיס הנתונים הוא לאחסן ולנהל את המידע של המשתמשים וכמו כן בבניית רשימות המשתמשים הרשומים.

בחרנו להשתמש בבסיס נתונים זה עקב הקלות היחסית בהתממשקות של PHP לבסיס הנתונים MySql.

## טבלאות בסיס הנתונים

### טבלת ה- Users

טבלה זו משתמשת לאחסון הנתונים האישיים של המשתמש.

|  |  |  |
| --- | --- | --- |
| **שם עמודה** | **סוג תוכן** | **תיאור** |
| **User\_id** | varchar | שם מזהה ייחודי לזיהוי המשתמש במערכת. המשתמש מכניס את התעודת זהות שלו וע"י זה הוא נכנס למערכת |
| **password** | varchar | סיסמת כניסה של המשתמש למערכת. |
| **Email** | varchar | דוא"ל של המשתמש, משמש בעיקר לצורך שיחזור סיסמא ( כרגע לא בשימוש ). |

# 

# טבלת parents

טבלה זו משמשת לאחסון השם משתמש שם ההורה והמספר הסלולארי של המנוטר

|  |  |  |
| --- | --- | --- |
| **שם עמודה** | **סוג תוכן** | **תיאור** |
| **parent\_user\_id** | Varchar | שם מזהה ייחודי לזיהוי המשתמש במערכת. המשתמש מכניס את התעודת זהות שלו וע"י זה הוא נכנס למערכת |
| **Child\_Number** | Varchar | שם מזהה יחודי לזיהוי המשתמש המנוטר . המספר הסלולארי של המשתמש המנוטר מוכנס מהחלון של הרשמה |

### טבלאות clients

טבלה זו משמשת עבור שמירת נתוני המשתמש המנוטר שבאים מהסלולארי

|  |  |  |
| --- | --- | --- |
| **שם עמודה** | **סוג תוכן** | **תיאור** |
| **Name** | Varchar | שם מזהה ייחודי של המשתמש המנוטר. |
| **Number** | Varchar | שם המשתמש ייחודי של סלולארי של המשתמש המנוטר. |
| **Data** | Text | שומר את המידע ( לוג ) של השיחות ( Call ) שמתקבלות מהסלולארי של המנוטר |
| **SMS\_Data** | Text | שומר את המידע ( לוג ) של ההודעות ( SMS ) שמתקבלות מהסלולארי של המנוטר |
| **Monitored** | Text | מציין במידה ואיש הקשר מנוטר או לא מנוטר |
| **Total\_SMS** | Text | שומר את הכמות של הסכום הודעות של המשתמש המנוטר |

# תיאור המסכים

## תוכנת משתמש קצה

### מסך הכניסה

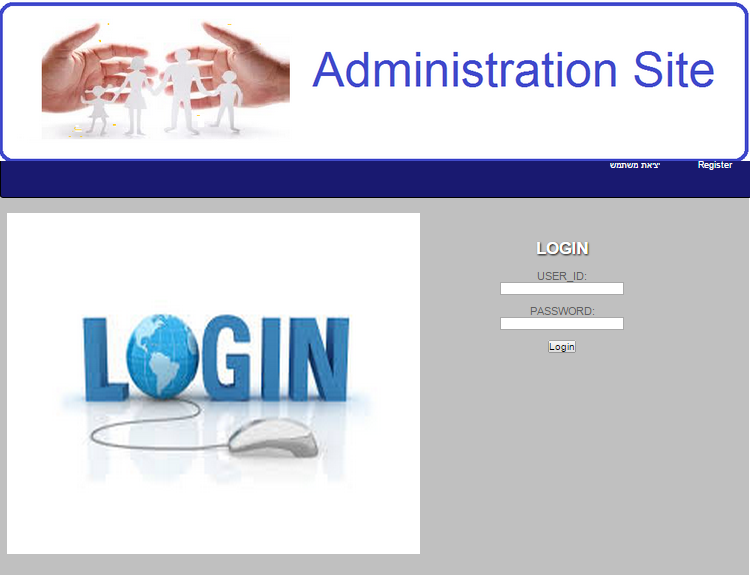
בזמן הפעלת האתר, מופיע למשתמש מסך כניסה למערכת שבו המשתמש נדרש להכניס את פרטי ההתחברות שלו:

* **שם משתמש**
* **סיסמא**

אשר שמורים בבסיס הנתונים על השרת.

בזמן הכניסה לאתר, התוכנה מבצעת את ההתחברות לשרת ומאמתת את הנתונים אשר הוזנו ע"י המשתמש. במקרה של הצלחת התחברות המשתמש מקבל אישור ומסך ההתחברות מתחלף במסך הראשי של המשתמש.

במקרה של כשלון המשתמש מקבל הודעה מתאימה על המסך.

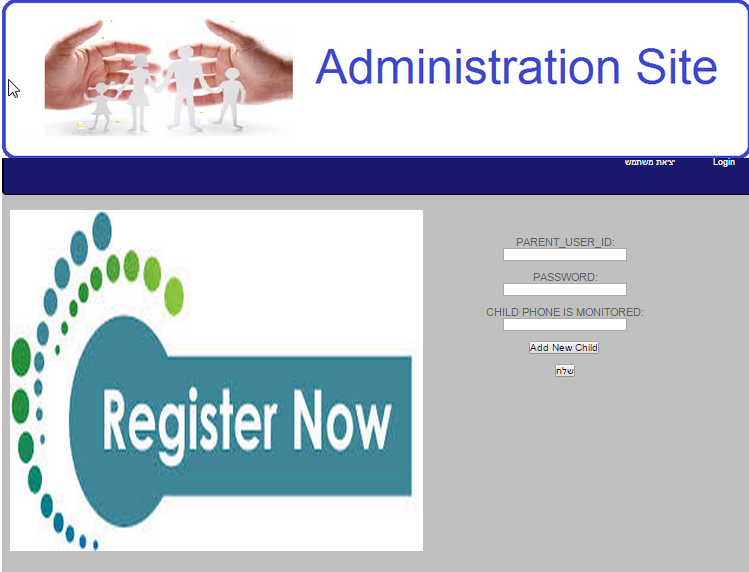
במידה שהמשתמש אינו רשום למערכת, יש באפשרותו להיכנס למסך הרישום ע"י לחיצה על כפתור register. 

### מסך רישום משתמש חדש

משתמש שעדיין לא רשום בבסיס הנתונים על השרת ועדיין אין לו פרטי התחברות יכול לבחור להיכנס לעמוד הרישום וכאן להזין את פרטיו האישיים:

* **Parent\_user\_id (שם משתמש)** – שם ייחודי אשר חייב להיות מינימום 9 תווים המשמש לזיהוי והתחברות למערכת.
* **Password )סיסמא)** – סיסמא שהמשתמש בוחר לעצמו עבור כניסה למערכת – חייב להיות מינימום 6 תווים, הסיסמא שמורה בבסיס הנתונים על השרת.
* **Child phone is monitored (מספר הטלפון של המשתמש המנוטר)** – אופציונאלי, המשתמש יכול להכניס את שמו הפרטי.

בסיום תהליך הרישום יכול המשתמש לחזור למסך ההתחברות ולהתחבר למערכת.



### מסך הראשי של המשתמש

לאחר התחברות מוצלחת למערכת נפתח למשתמש החלון הנ"ל, ובנוסף נפתחות האפשרויות הבאות בסרגל התפריטים העליון כגון:

* אודות האתר
* חלון עזרה
* יצירת קשר
* התנתקות מהמשתמש וחזרה למסך ההתחברות. במקרה והמשתמש בחר באפשרות זאת והפרטי החיבור שלו שמורים עבור החיבור האוטומטי אז לאחר ביצוע פעולה זאת החיבור האוטומטי מתבטל להתחברות הבאה.



כמו כן ניתן לראות בחלון את השדות הבאים:

* בחלק העליון של המסך מופיע חיווי עם שם המשתמש המחובר ברגע נתון

**בחלק השמאלי** של המסך מופיע שני לחצנים

* **לחצן SMS ( הודעות )** – לחיצה על כפתור זה פותחת חלון שבתוכו מידע של לוגים

של ההודעות SMS שיש לך על המשתמש המנוטר

* **לחצן CALL** – לחיצה על כפתור זה פותחת חלון שבתוכו המידע של לוגים של השיחות שיש לך על המשתמש המנוטר

### מסך ההודעות

כאשר ההורה או המעסיק לוחץ על הלחצן של **לחצן SMS**  הוא יעבור לחלון הבא

ששמה הוא הראה את הלוג של ההודעות



במסך הזה אנו רואים:

את השם של המנוטר – במקרה שלנו זה ניר עינת

מספר הסלולארי שלו – 0542348200

הודעה SMS – ששמה זה מכיל שעה ותאריך מתי התקבל ההודעה ואת התוכן של ההודעה

### מסך שיחות

כאשר ההורה או המעסיק לוחץ על הלחצן של **לחצן CALL**  הוא יעבור לחלון הבא

ששמה הוא הראה את הלוג של השיחות



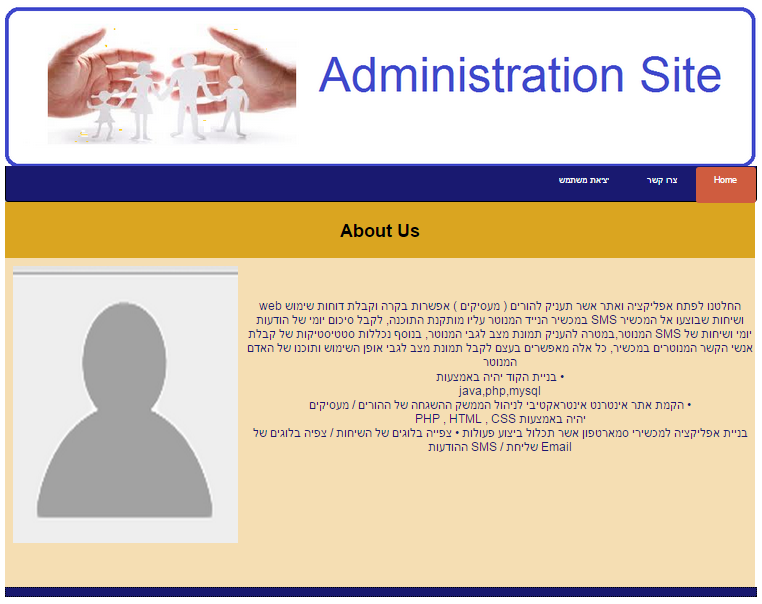
במסך הזה אנו רואים:

את השם של המנוטר – במקרה שלנו זה ניר עינת

מספר הסלולארי שלו – 0542348200

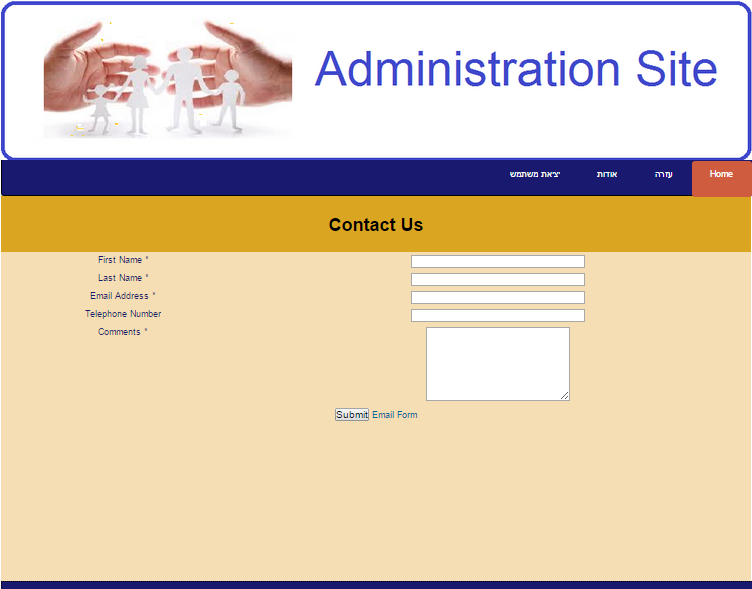
הודעה שיחה – ששמה זה מכיל שעה ותאריך מתי התקבל השיחה

### מסך אודות



### חלון יצירת קשר

בחלון הזה יהיה אפשרות לשלוח אלינו את ההודעה כדי שיצרו איתנו קשר



מסך ABOUT + פניה של משתמש בנוגע לבאגים עדכונים וכו'

First Name - שם פרטי שלו

Last Name - שם משפחה שלו

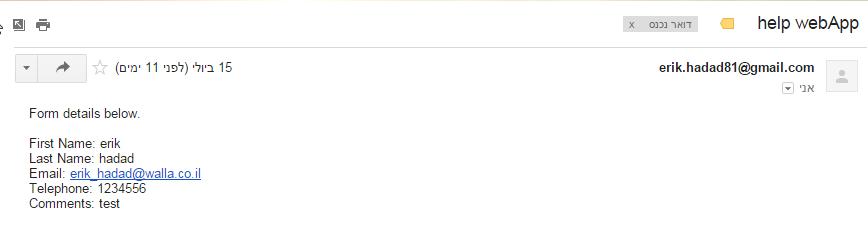
Email Address - המייל של ההורה שאליו נחזיר את התשובה של השליחה

Telephone Number - מספר הסוללארי של ההורה או המעסיק שנוכל לחזור אליו

Comments – הודעה של הבעיה של להורה או למעסיק

וזאת הדוגמא שאנו נקבל כאשר זה מתקבל בשליחת מייל

פניה מגיעה ישירות לתיבת התואר:



# Literature

## Similar Applications

1. **Name:**  SpyPhone

**Link:** <http://www.spyphone.com>

**About:** As mentioned, there are a variety of applications that promise to make you spies in a moment, but it is important to choose a reliable and high-quality applications

One Company that offers high quality spy app is **SpyPhone**

SpyPhone is another common spyware, designed to most smartphones in use in, Using the software, installing it on top of the device is made in physical form, without a link or an SMS message, you can listen to telephone calls made on the device on which it is installed, read incoming messages to the device in and out it, to seek Call

History.

1. **Name:**  SpyBubble

**Link:** <http://www.prospybubble.com>

**About:**  SpyBubble is another common spyware, which helps in monitoring your spouse, child, employee, etc. you will need to install this program to their smartphone to start recording. After you installed you have the ability to read messages send and received by the user, and you will see Call Logs, how many calls were made to each number.

1. **Name:** Flexispy

**Link:** <http://www.flexispy.com/?AVGAFFILIATE=69975>

**About:** that offers high quality spy app is Flexispy

The Hacker can monitor Android smartphone and table using internet capability. It works on all kinds of devices like iPhones and iPads, BlackBerry and Nokia

## 

## Relevant web sites for the project

1. “Data Backup.” *Developer.android.com.* Developers*.* Retrieved from http://developer.android.com/guide/topics/data/

backup.html

1. “Definition: Two-factor authentication.” *Searchsecurity.techtarget.com.* Search Security. Retrieved from <http://searchsecurity.techtarget.com/definition/two-factor-authentication>
2. <http://javabook.co.il>

This is a Hebrew website by Haim Michael from HIT collage for teaching Java programming language by simple videos, example and good explanations for each subject.

1. <https://wiki.gentoo.org/wiki/MySQL/Startup_Guide>

This website is a startup guide for MySQL query language,

The website contains examples and explanation of queries in MySQL.

1. <http://www.w3schools.com/html/>
2. <http://www.tizag.com/mysqlTutorial/mysqlinsert.php>