מסמך פרוייקט גמר בקורס פייטון

"רשת תחנות לשאיבת מיים"

# סיכום מנהלים:

פרוייקט מתאר רשת של תחנות לשאיבת מיים, אשר מדווחות לשרת המרכזי כל פרק זמן את מספר הנתונים, כגון סטטוס של תחנה, מצב אזעקות, וכו'.

פרוייקט כתוב בשפת Python 2.7 כולל התאמות הנדרשות למעבר ל-Python 3 תוך זמן קצר.

הפרוייקט מנהל את תחנות הקצה וגם שרת המרכזי, כאשר כל צד בזמנו מתפקד במבחינת רשת גם כ"Client" וגם כ-“Server”.

הפרוייקט בנוי בדגש על חוויית המשתמש ואבטחת מידע.

קוד של הפרוייקט נמצא בGitHub וניתן להורדה מלינק:

<https://github.com/damashnik/Python-Project>

# מבנה כללי:

בפרוייקט קיימים שתי ישויות המרכזיים: שרת (server) ותחנת קצה (client). לכל ישות מוגדרים שני תהלכים מרכזיים:

ישות "שרת":

* ממתין לבקשות מתחנות קצה – חיבור תחנה חדשה, שינוי מפתח או מחיקת תחנה
* תשאול תחנות קצה – פעם במספר דקות (לפי הגדרה בקובץ קונפיגורציה) שרת מתשאל כל תחנות הקצה המוגדרות, שולח להם שאילתא “Keep Alive”.

ישות "תחנת קצה"

* שולח בקשות לשרת, לפי דרישת המשתמש
* מענה לתשאול מהשרת “Keep Alive” – שולח חזרה מידע המופיע בקובץ status.txt

# מבנה הפרוייקט:

## תשתית:

פרוייקט רץ על גבי שרתי Linux , פותח על פלטפורמה VirtualBox עם מכונות מריצות Ubuntu Server 16.04, בנוסף הותקנו רכיבים הבאים:

OpenSSH – לתובת חיבור והעברת מידע בין תחנת פיתוח, שרת ותחנות קצה

OpenVPN – אופציונאלי, מאפשר הקמת VPN Channel לטובת חיבור מאובטח בין שרת ותחנות קצה

Sqlite3 – רק בצד השרת, לטובת אחסון המידע המגיע מתחנות קצה

Flask – רכיב WebServer לטובת ממשק משתמש לתחנות קצה

## בסיסי נתונים:

שרת מנהל רישום של תחנות קצה ורישום תשובות שהוא מקבל מתחנות קצה בבסיב נתונים מבוסס sqlite3.

כל הנתונים מאוחסנים בשתי טבלאות:

*clients* – טבלה מחזיקה נתונים של תחנות קצה רשומות:

`clientID` **int**(10) **NOT NULL**,  
`clientIP` **varchar**(16) **NOT NULL**,  
`clientKey` **varchar**(24) **NOT NULL**,  
`clientName` **varchar**(50) **NOT NULL**,  
`datetime` **varchar**(50) **NOT NULL**,  
**PRIMARY KEY**(clientID)

*reports* – טבלה מחזיקה תשובות של כל התשאולים של כל תחנות קצה:

`id` **int**(10) **NOT NULL**,  
`clientID` **int**(10) **NOT NULL**,  
`datetime` **varchar**(16) **NOT NULL**,  
`status` **int**(10) **NOT NULL**,  
`alarm1` **int**(1) **NOT NULL**,  
`alarm2` **int**(1) **NOT NULL**,  
**PRIMARY KEY**(id)  
**FOREIGN KEY** (clientID) **REFERENCES** clients (clientID)

כברירת מחדל משתמשים ב DataBase:

/home/user/water

מיקום ושם DataBase ניתנים לשינוי, יש לעדכן קובץ server.cfg שורה db.

הקמת בסיס נתונים על שרת מתבצעת בצורה הבאה:

1. In Terminal run the following command:
   1. sqlite3 /home/user/water
   2. .open <Project\_home\_dir>/SQL/create\_tables.sql
2. In order to check if database has been created successfully run the following commands from sqlite3 shell:
   1. .tables
   2. Select \* from clients;
   3. Select \* from reports;

## ניהול לוגים:

כל פעולה של שרת רושמת שורת לוג בקובץ:

/var/log/water-server/water.log

כל פעולה של תחנת קצה רושמת שורת לוג בקובץ:

/var/log/water-client/water.log

**הערה חשובה: עקב היבטי אבטחת מידע פרוייקט רץ עם הרשאות נמוכות של משתמש user. על מנת לאפשר כתיבה של לוגים יש לשנות הרשאות של ספריות /var/log/water-client ו-/var/log/water-server בהתאם. מומלץ להעביר ownership על ספריות אלו למשתמש user.**

## שרת:

רכיב שרת מורכב משני קבצים:

server.py – רכיב מאזין לפקודות של תחנות הקצה, ומבצע רישום של תחנות חדשות, שינוי מפתחות ומחיקה של תחנות קצה ישנות.

ברכיב זה נעשה שימוש במודולים הבאים:

**import** datetime  
**import** ConfigParser  
**import** socket  
**import** sqlite3

### ארכיטקטורת הרכיב:

שלב 1 - מתבצעת קריאה של קובץ קונפיגורציה, על מנת לשוף פרמטרים מקונפגים.

שלב 2 – שרת נכנס למצב האזנה ברשת, בפורט שמוגדר בקובץ קונפיגורציה

שלב 3 – שרת מקבל בקשת התחברות מתחנת קצה, כולל רשימת פרמטרים

שלב 4 – שרת מבצע את הפקודה שנשלחה (הוספת תחנה חדשה, שינוי מפתח או מחיקה של תחנה ישנה)

כל שלב מלווה בכתיבה לקובץ לוג.

שלב 4 מלווה בעדכון database.

server\_deamon.py – רכיב מופעל כל מספר דקות, ושולח בקשת KeepAlive לתחנות קצה, ומעבד את התוצאות.

ברכיב זה נעשה שימוש במודולים הבאים: