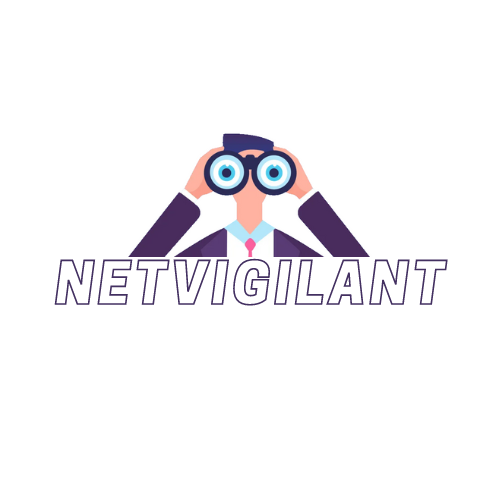
A picture containing graphical user interface

Description automatically generated



**שם:** שלי בן ציון

**ת.ז.:** 327889606

**שם המנחה:** גולן מור

**חלופה: הגנת סייבר ומערכות הפעלה**

**יוני 2023**

**תוכן עניינים**

**תקציר ורציונל הפרוייקט** 3

**מבוא ורקע כללי** 4

**מטרת הפרוייקט** 6

**שפת התכנות וסביבת העבודה** 7

**ניסוח וניתוח הבעיה האלגוריתמית** 8

**תיאור אלגוריתמים קיימים** 9

**מבנה וארכיטקטורה , הפתרון הנבחר** 10

**מימוש הפתרון בשכלול הקוד עם שפת התכנות** 13

**תיאור המודולים של מערכת התכנה** 19

**תיעוד הקוד** 20

**השוואת העבודה עם פתרונות ויישומים קיימים** 24

**הערכת הפתרון לעומת התכנון והמלצות לשיפורו** 25

**תיאור של הממשק למשתמש – הוראות הפעלה** 26

**מבט אישי על העבודה ותהליך הפיתוח** 28

**ביבליוגרפיה** 29

**קוד התוכנית** 30

# תקציר ורציונל הפרויקט

בכל ארגון שבו מחשבים קיים מנהל מערכת (System Administrator). תפקידו להיות אחראי על התחזוקה והתפקוד השוטף של מערכת המחשבים בארגון. לעקוב אחרי כל כך הרבה פרמטרים בכל כך הרבה מחשבים יכולה להיות עבודה קשה, ועל כן המוצר שלי, "NETVIGILANT", בא לעזור למנהל המערכת לעקוב אחרי הפעילות של כל המחשבים ברשת.

למנהל המערכת תהיה האפשרות לראות איזה מחשב במערכת דלוק ואיזה לא. עבור המחשבים הכבויים תהיה לו האפשרות להדליק אותם דרך המחשב שלו. עבור המחשבים הדלוקים תהיה לו האפשרות לצפות במידע חשוב על הביצועים, הרשת והחומרה של אותם מחשבים. המוצר יזין לתוך מסד נתונים את חלק מהמידע וינתח אותו. בנוסף הוא יתריע למנהל המערכת במקרה וחלק מהמידע חורג מתווך מוגדר מראש.

שירות נוסף שהמוצר ייתן הוא עדכון גרסה של אפליקציות ותוכנות על המחשבים במערכת מרחוק, בנוסף להדגרה של תהליכים שאסור שירוצו על המחשבים במערכת.

**מדוע בחרתי בפרויקט הזה?**

בחרתי לעשות את הפרויקט הזה כפרויקט הגמר שלי כחלק ממגמת הנדסת תוכנה מכמה סיבות.

* רציתי לעסוק במשהו שישלב הרבה אלמנטים שונים כך שאוכל ללמוד ולהתנסות בהרבה דברים בדרך. בשביל הפרויקט שלי השתמשתי למשל בתקשורת, התממשקתי עם מערכת ההפעלה, רכשתי ידע על החומרה של המחשב ועוד. מאוד שמחתי להזדמנות ללמוד תוך כדי עשיה, "ללכלך את הידיים" ולבנות משהו שמיש.
* רציתי ליצור כלי שיוכל לשמש ארגונים ולהקל על ניהול מערכת המחשבים שלהם.

# מבוא ורקע כללי

כל חברה בימינו עובדת עם מחשבים רבים. בלי אותם מחשבים, החברה לא הייתה יכולה להתקיים כי בעזרתם עושים הכל – מבצעים פעולות, מתקשרים עם שאר העולם, שומרים מידע ועוד. ולכן לכל חברה יש בעל תפקיד "מנהל מערכת". יש למנהל המערכת עבודה מרובה. עליו:

* לוודא שכל המחשבים עובדים בצורה תקינה.
* מידי פעם לעדכן גרסאות של כל המחשבים והתוכנות בהם.
* לבצע הגדרות והתקנות של ציוד.
* לעקוב אחרי ביצועי המערכת.
* להבטיח שכל המחשבים עובדים קשורה.

ועוד הרבה.

אם נדמיין חברה גדולה, עם מלא מחשבים ובניין ענק עם כמה קומות, אנו מבינים שהעבודה שלו לא קלה. יש לו הרבה אחריות והוא חייב לעקוב אחרי תפקוד כל המחשבים ברשת תמיד. תפקוד לקוי מצידו יכול להוביל להשלכות שישפיעו על החברה כולה.

# מטרת הפרויקט

**מה המוצר המוגמר אמור לבצע:**

המוצר אמור לעזור למנהל מערכת מחשבים לנהל את המערכת בצורה יעילה ונוחה. אצלו במחשב יהיה השרת, איתו הוא יוכל לראות מי כל המחשבים ברשת והאם הם דלוקים.

עבור כל המחשבים הדלוקים הוא יוכל לראות מידע שימושי כגון הביצועים של כל מחשב, מידע על הusers במחשב, מידע על החומרה שלו ועוד. בנוסף, מנהל המערכת יהיה מסוגל לבצע עדכונים של תוכנות ואפליקציות על המחשבים מהמחשב שלו.

עבור כל המחשבים הכבויים (שלא מחוברים לשרת באותו רגע), מנהל המערכת יהיה מסוגל להדליק אותם מהמחשב שלו, מרחוק (WOL).

מנהל המערכת יהיה מסוגל גם לקבל התרעות במקרה והנתונים – טמפרטורה של הcpu, שימוש cpu, או זיכרון עוברים רף מסוים אשר הוא מגדיר. יתר על כן, הוא יוכל להגדיר "forbidden processes", תהליכים שאסור שירוצו על המחשבים ברשת, ויקבל התרעה ואפשרות לראות את רשימת התהליכים הרצים על אותו המחשב ומידע עליהם, במקרה ואותם תהליכים רצים.

עבור כל הלקוחות התוכנה תרוץ ברקע.

**דרישות מרכזיות:**

דרישות פונקציונליות:

* המוצר יאפשר למנהל המערכת לראות את כל המחשבים ברשת הפנימית.
* עבור כל מחשב דולק, מנהל המערכת יוכל לראות את המידע הבא:
* כתובת ip וכתובת mac.
* מידע על החומרה של המחשב.
* מידע על הביצועים של המחשב.
* מידע על התהליכים הרצים על המחשב.
* מידע על הusers במחשב.
* מידע על הnetwork interface.
* מידע על הפורטים במחשב.
* Network information.
* מידע על הdrives במחשב.
* המוצר יתריע למנהל המערכת בעזרת חלון pop up והודעה למייל כאשר אחד מהנתונים – טמפרטורה של הcpu, שימוש הcpu או זיכרון עברו רף מסוים.
* המוצר יאפשר למנהל המערכת להגדיר תהליכים שאסור שירוצו על המחשבים ברשת ויתריע לו בעזרת חלון pop up והודעה למייל במקרה ואחד מהתהליכים המוגדרים כאסורים רצים.
* המוצר יאפשר למנהל המערכת לקבל רשימה של התהליכים הרצים על מחשב מסוים ומידע עליהם.
* עבור כל מחשב דולק, מנהל המערכת יהיה מסוגל לבצע עדכוני גרסה של תוכנות ואפליקציות עליו.
* עבור כל מחשב כבוי, מנהל המערכת יהיה מסוגל להדליק אותו מרחוק.

דרישות לא פונקציונליות:

* ממשק אינטואיטיבי וקל לעבודה עבור מנהל המערכת.
* יהיה מספר לא מוגבל של לקוחות ברשת.
* המידע שיתקבל על המחשבים יהיה מדויק ומהימן.
* השרת והלקוח יפעלו בצורה אוטומטית כשהמחשב נדלק.
* עבור הלקוחות, המוצר יעבוד לגמרי בקרע.

אילוצים:

* המוצר יעבוד רק על windows 10 או 11.
* עבור כל מחשב ברשת יהיה צריך להגדיר את קוד הלקוח אינדיבידואלית.

**תרחישים במערכת:**

1. שימוש ראשוני – שרת:

* מנהל המערכת יצטרך לעדכן את כתובת המייל שלו בקובץ בשם admin\_info.txt.
* יגדיר גם רף cpu, טמפרטורה וזיכרון באותו הקובץ.

2. שימוש ראשוני – לקוח:

* על מנהל המערכת להגדיר בקבצי server\_ip.txt את כתובת הip של המחשב עליו רץ השרת

3. שימוש שותף – שרת:

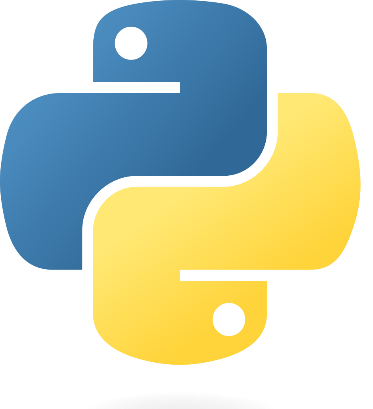
* מנהל המערכת יראה בGUI את כל המחשבים ברשת.
* הוא יהיה מסוגל לצפות המידע עבור כל מחשב –
  + מידע על החומרה
  + מידע על הרשת
  + מידע על הדיסקים
  + מידע על המשתמשים
  + מידע על כרטיס הרשת
  + מידע על הפורטים הפתוחים
* יהיה מסוגל להדליק מרחוק מחשבים כבויים על ידי לחיצה ימנית עליהם.
* יהיה מסוגל לעדכן גרסה של תוכנות ואפליקציות על מחשבים מרחוק.
* יהיה מסוגל להגדיר תהליכים שאסור שירוצו על המחשבים ברשת.
* יקבל התרעת pop up והתראה במייל במקרה ואחד המחשבים עבר את הרף שמנהל המערכת הגדיר בקובץ של הcpu או הזיכרון או הטמפרטורה.

4. שימוש שוטף – לקוח:

* עבור הלקוח הכל יקרה לגמרי ברקע.

# שפת התכנות וסביבת העבודה

Python -

שפת התכנות העיקרית שהשתמשתי בה כדי לכתוב את הפרויקט היא python 3.11, תוך שימוש בסביבת הרצה של Visual Studio Code. Python היא שפת תנות עילית המאפשרת תכנות מונחה עצמים, משפות התכנות הפופולאריות ביותר שקיימות. מאוד נוח לעבוד איתה בעיקר בגלל התמיכה הרחבה שלה במספר רב של ספריות ומודלים שונים. בחרתי בשפה הזו בגלל הנוחות שבכתיבה וקריאה שלה, ובגלל התמיכה שלה בספריות ומודלים שעזרו לי לבנות את הפרויקט. איתה כתבתי את כל הפרויקט – את התקשורת בין השרת והלקוח, את השגת המידע על כל לקוח ברשת, ואת ממשק המשתמש.

SQL –

בשביל לנהל את מסד הנתונים שלי השתמשתי בספריה של python בשם SQLite. הספרייה הזו מאפשרת להריץ פקודות בSQL בתוך קוד בpython כדי לנהל מסד נתונים. כאמור, SQL היא שפה בה אפשר להשתמש כדי לעבד מידע במסדי נתונים, המאפשרת שליפה והוספה של רשומות ונתונים לטבלאות, יצירה של טבלאות ושינוין. השתמשתי בה כדי לעדכן את מסד הנתונים שלי בעת כניסה של לקוח חדש/עדכון סטטוס החיבור של לקוח ישן, ובשביל לשמור את המידע שכל לקוח שולח לשרת במסד הנתונים.

# ניסוח וניתוח הבעיה האלגוריתמית

בעיה ראשונה:

הדרך להשיג מידע על המחשבים – כחלק מהפרויקט שלי הייתי צריכה למצוא דרך להשיג מידע שונה אודות הביצועים, החומרה, הרשת ועוד, עבור כל מחשב במערכת.

בעיה שניה:

לאפשר כמה clients במקביל – בגלל שהשרת שלי צריך להיות מחובר לכל הלקוחות ברשת במקביל, הייתי צריכה למצוא דרך לאפשר זאת.

בעיה שלישית:

עדכון הdatabase – עבדתי עם sqlite3 כדי לעדכן את הdatabase. הייתי צריכה לעדכן אותו מהמון threads שונים, בהתחשב שזה שלכל client במערכת אני יוצרת שני threads – אחד להידברות עם השרת והשני לעדכונים שוטפים של בדיקות תקינות. sqlite3לא יכול לעבוד עם אותו חיבור לdatabase מthread שונה מהthread בו הוא נוצר, ולכן היה עלי למצוא פתרון לעדכון הdatabase מthreads שונים.

בעיה רביעית:

ריצה אוטומטית – עבור השרת והלקוחות, המוצר שלי אמור להתחיל לרוץ אוטומטית ברגע שהמחשב נדלק. כך הלקוחות מתחברים ישר כשהם יכולים לשרת ומנהל המערכת יכול לראות את כל המחשבים הדלוקים ברשת. הייתי צריכה למצוא דרך לאפשר זאת.

# תיאור אלגוריתמים קיימים

פתרונות קיימים לבעיה ראשונה:

Python היא שפה שתומכת בספריות רבות, לכן היו הרבה פתרונות לבעיה זו.

1. Psutil (python system and process utilities)

זוהי ספריה בpython שיכולה לספק מידע על תהליכים שרצים על המחשב ועל הביצועים שלו, למשל cpu, memory, disks, network, sensors ועוד. הספרייה הזו מאוד שימושית לבדיקת הביצועים של המחשב וניהול הריצה של תהליכים במחשב. הספרייה אכן תומכת גם בwindows אך חלק מהפונקציות שלה לא תומכות בו.

הספרייה לא מגיעה אוטומטית עם python ולכן יש להתקין אותה באמצעות הרצת הפקודה הבאה בcmd:

pip install psutil

הספרייה יכולה למשל לספק את שימוש הcpu בעזרת הפונקציה psutil.cpu\_times() אך לאחר שניסיתי אותה כמה פעמים ובכמה דרכים שונות, שמתי לב שהמידע שהיא מחזירה הוא לא קונסיסטנטי, לא תואם למציאות והרבה פעמים היא מחזירה פשוט 0.0.

חוץ מזה, היה לי חשוב להשיג מידע על החיישנים של המחשב בעיקר כדי לגלות את טמפרטורת המחשב – מידע שחשוב שמנהל מערכת ייחשף אליו כדי לוודא שטווח הטמפרטורות של המחשב תקין. לעומת זאת, כל מה שקשור בחיישני טמפרטורה וחיישני מאווררים למשל בספריה psutil, לא תומך בwindows, אלא בlinux.

1. WMI (Windows Management Instrumentation)

זהו מודל שעוטף את WMI ומאפשר לנו להשתמש ביכולות שלו. WMI הוא יישום של Microsoft למודל שמאפשר קבלה של כמעט כל פיסת מידע מכל מחשב אם הוא רץ עם ההרשאות המתאמות. כך אנחנו מקבלים דרך לגשת ולנהל מידע ומשאבי מערכת שונים. בעזרתו אנחנו יכולים להתממשק בpython למערכת ההפעלה (בין היתר) ולקבל מידע שונה עליה, למשל, מידע על המערכת (חומרה, גרסת מערכת ההפעלה ועוד), מידע על המשתמשים ואבטחה (שמות הusers של המחשב, סיסמאות ועוד), מידע על תהליכים שרצים על המחשב, ועוד המון מידע שימושי כשמדובר על פיקוח על רשת.

גם את הספרייה הזו יש להתקין באמצעות הרצת הפקודה הבאה בcmd:

Pip install wmi

1. Open hardware monitor

זוהי תוכנה open source שדרכה אפשר לראות מידע שונה על תפקוד המחשב, למשל, חישני טמפרטורה, מידע על הcpu והgpu, חומרה, וולטים ועוד. אפשר לראות דרכה את טמפרטורת הcpu בעזרת חיישני הטמפרטורה שנמצאים במעבדי intel וADM, שתי החברות הנפוצות בעולם למעבדים. התוכנה חינמית ואכן תומכת ב.windows

כאשר מורידים אותה מהאתר, מקבלים תיקייה דחוסה שבה נמצא קובץ הexe אך גם נמצא קובץ dll. אפשר להשתמש בdll כדי להשיג את המידע הנחוץ לנו בpython.

כדי להשתמש בdll בתוך הקוד, אפשר להשתמש בספריה wmi (Windows Management Instrumentation) של python. הספרייה מספקת ממשק של מערכת ההפעלה שדרכו ניתן להשיג מידע על רכיבים שונים. במקום להתממשק אליה נוכל להתממשק אל התוכנה כך:

w = wmi.WMI(namespace="root\OpenHardwareMonitor")

וכך לגשת למידע הנדרש.

כשניסיתי את הדרך הזו ראיתי שהמידע המסופק אכן תואם למציאות וקונסיסטנטי. הדבר היחיד שלא הסתדר הוא שכאשר התוכנה הייתה סגורה, חלק מהערכים המוחזרים היו None תמיד. לכן יש צורך לפתוח את התוכנה ברקע וכך להשתמש במידע שבה.

התוכנה כן יחסית ישנה, העדכון האחרון שלה היה ב2020, אבל היא עונה על חלק מהצרכים בצורה טובה.

פתרונות קיימים לבעיה שניה:

אני יוצרת שרת tcp בפרויקט תוך שימוש בספרייה "socket" של python. מכיוון ואני צריכה שכמה מחשבים יתחברו במקביל לשרת שלי, הייתי צריכה לבחור בדרך אחת מהדרכים הבאות כדי לאפשר זאת:

1. Asyncio

ספרייה לכתיבת קוד במקביל בעזרת פקודות של async/await. הספרייה משמשת כבסיס למספר מסגרות אסינכרוניות של Python המספקות רשתות ושרתי אינטרנט בעלי ביצועים גבוהים, ספריות חיבור למסד נתונים, ועוד. בעזרתה אנחנו נמנעים משימוש בthreads מרובים אך היא מורכבת יותר ליישום ולניפוי באגים בהשוואה לפתרונות אחרים הקיימים.

1. Select

מודל שמאפשר גישה לפונקציות select() וpull() שזמינות ברוב מערכות ההפעלה. כך אנו מנהלים רשימה של לקוחות פעילים ומפקחים על הפעולות שלהם בעזרת הפונקציות האלו. כשלקוח למשל שולח מידע או כשמתקבל חיבור חדש, הsocket המתאים הופך להיות מוכן, ואפשר לטפל בו בהתאם. כך אנחנו נמנעים מיצירת threads מרובים והגישה הזו יותר חסכנית במשאבים, אך יותר קשה לניהול ותחזוקה.

1. Threading

בעזרת מודל זה אנחנו יכולים להקנות לכל לקוח thread משלו. כאשר מתקבל חיבור חדש, נוצר thread חדש אשר מטפל באותו הלקוח. הגישה הזו בהחלט נוחה ומאפשרת חיבור של כמה לקוחות במקביל, אבל כאשר מדובר על מספר רב של לקוחות היא יכולה להיות "בזבזנית" במשאבי המערכת.

פתרונות קיימים לבעיה שלישית:

כדי להשתמש בספרייה sqlite3 לצורך ניהול database, עלינו ליצור connection לdatabase, וממנו ליצור cursor שמאפשר לנו להריץ פקודות בSQL כדי לשלוף נתונים ולהכניס אותם לטבלאות השונות בdatabase. את הconnection והcursor לא ניתן להעביר בין threads וחייבים להשתמש בהם רק בthread בו הם נוצרו, sqlite לא תומך בmultithreading. מכיוון ואני משתמשת בmultithreading ויוצרת thread חדש עבור כל לקוח שמתחבר לשרת, הייתי צריכה למצוא דרך לנהל זאת תוך עמידה במגבלות של הספרייה ושמירה על הערכים של clean code.

פתרונות:

1. הפתרון הראשון שחשבתי עליו הוא ליצור connection וcursor חדש עבור כל מתודה בשרת שצריכה לגשת לdatabase. כך אני לא עוברת על המגבלות של הספרייה ובכל מקום שצריכה להיות הידברות עם הdatabase נוצר חיבור משלו. אמנם השיטה הזו מובילה לקוד מסורבל ולא נוח לעבודה איתו.
2. הפתרון השני עליו חשבתי הוא יצירה של thread שכל האחריות שלו היא עדכון של הdatabase. יצרתי רשימה גלובלית שאליה כל thread יכול היה להוסיף את מה שהיה צריך. כמובן שבשביל זה אני השתמשתי בlock, דרך להבטיח שברגע שthread מנסה לגשת לרשימה ולשנות אותה אף thread אחר לא יכול יהיה להפריע לו באמצע, באמצעות כך שאני עושה למנעול acquire לפני הכתיבה וrelease אחריה. הthread שאחראי על עדכון הdatabase קורא מהרשימה בצורה של FIFO (first in first out) כדי שהמשימות יתבצעו בסדר שבה נכנסו לרשימה, ולפי המידע ששלף הוא מעדכן את הdatabase. כך יש לי רק חיבור אחד לdatabase שקיים באופן קבוע.
3. הפתרון השלישי עליו חשבתי הוא ליצור מחלקה שכל תפקידה היא ניהול הdatabase. היא תאתחל אותו ואת הטבלאות בו במידה ולא אותחלו כבר, והיא תהיה אחראית על הוספה ועדכון המידע בו. כך אפשר להכיל בתוך השרת instance של המחלקה, ולהשתמש בה מכל thread בעזרת lock.

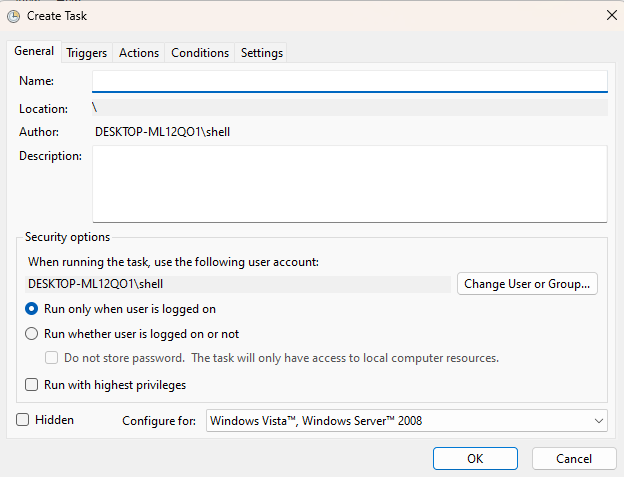
פתרונות קיימים לבעיה רביעית:

1. Task scheduler

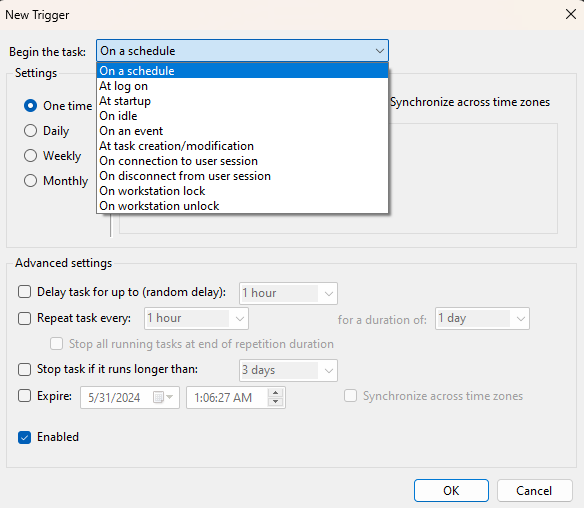
זהו כלי של windows שמאפשר להגדיר tasks – משימות שעל המחשב לבצע ומתי עליו לבצע אותן. הוא יכול למשל להפעיל אפליקציה, לשלוח הודעת מייל או להציג הודעה על המחשב.

כאשר פותחים אותו על המחשב (נמצא אוטומטית במחשבים בעלי מערכת הפעלה windows) אפשר ללחוץ על הכפתור Create task ובכך ליצור משימה חדשה.

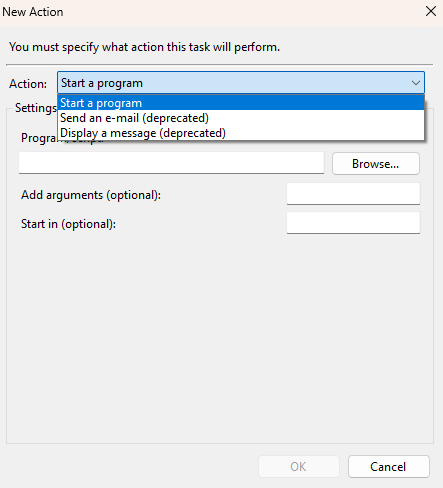
לאחר מכן נפתח חלון בו אפשר להגדיר את שם המשימה ולשים לה תיאור. בנוסף אפשר לתת לה פריווילגיות של admin ולהגדיר את המשימה כמוסתרת (hidden).



לאחר מכן ניתן לעבור לחלק של Triggers ולהגיד מתי המשימה צריכה לקרות:



עכשיו נגדיר מה המשימה בactions:



אפשר גם להוסיף תנאים (בחלון conditions) ולשנות את ההגדרות (בחלון settings).

כך ניתן להגדיר משימה שתתחיל את הקוד התוכנית כאשר המחשב נדלק. חשוב גם לתת למשימה את הפריווילגיות הגבוהות ביותר, כי הלקוחות והשרת שבניתי דורשים פריווילגיות גבוהות כדי לגשת למידע על המחשב.

1. Registry

זהו מאגר המרכזי של ההגדרות המשמשות את מערכת ההפעלה. הוא מכיל בתוכו הגדרות ממאפיינים של מערכת ההפעלה ונתונים על חומרה ועד מסך הרקע של שולחן העבודה.

# הפתרון הנבחר

פתרון בעיה ראשונה:

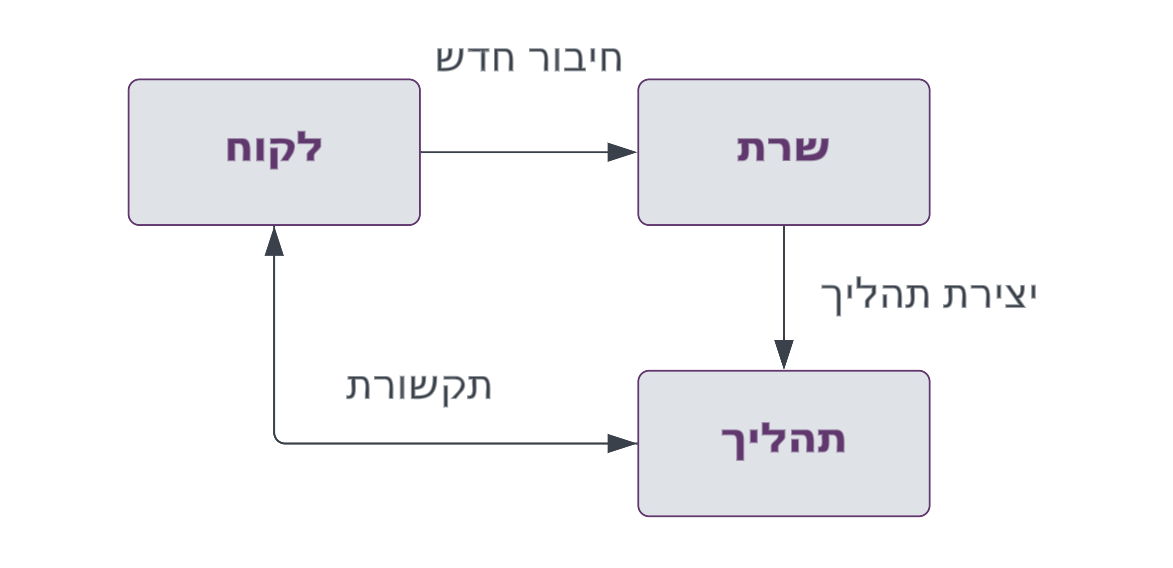
בשביל לקבל מידע אודות המחשב בclients בחרתי להשתמש בכל הפתרונות.

יצרתי מודל בשם snmp\_server.py ובתוכו כתבתי פעולות שנותנות לי מידע שונה אודות המחשב. קראתי לו כך מכיוון וSNMP הוא פרוטוקול לניהול רשת (Simple Network Management Protocol) שמשמש בעיקר רשתות TCP, הוא מספק אמצעי לניתור ובקרה של התקני רשת, איסוף סטטיסטיקות, ביצועים ואבטחה ברשת.

בתוכו יש לי כמה פונקציות וכל אחת משתמשת בחלק מפתרון אחר. בגלל ששמתי לב שכל אחד מהפתרונות מספק לי מידע לא מושלם, החלטתי לשלב את כולם לפתרון אחד ולהשתמש בכל הספריות שציינתי כדי לאסוף מידע שרלוונטי לי.

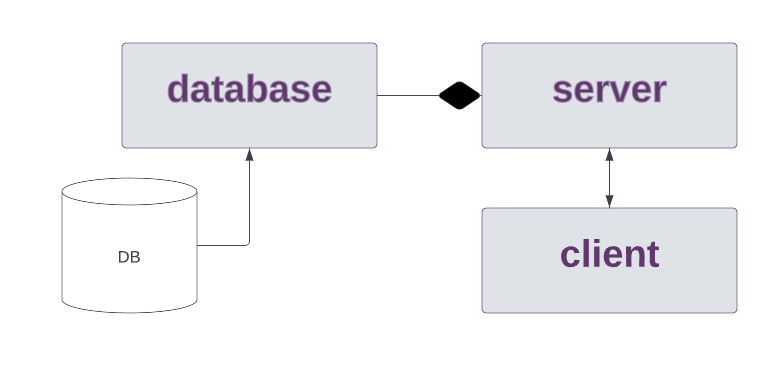
פתרון בעיה שניה:

הפתרון שבחרתי להשתמש בו הוא שימוש בthreading. בחרתי בפתרון זה הן בגלל הנוחות והפשטות שלו ביחד לשאר הפתרונות שמצאתי, והן בגלל ההכרות וההתעסקות הקודמת שלי עם הספרייה במסגרות שונות. אמנם השיטה הזו היא יותר "בזבזנית", אך היא יותר נוחה לתחזוקה ולעבודה איתה.



פתרון בעיה שלישית:

בשביל לעדכן ולהוסיף נתונים לdatabase בחרתי בפתרון השלישי. ניסיתי לבנות את הפרויקט שלי לפי עקרונות של clean code ולכן בחרתי בפתרון זה. Clean code מתאר את הדרך שלנו לכתוב קוד נקי שקל לעבוד איתו. ישנם הרבה עקרונות כדי לעמוד בסטנדרטים של clean code, ואחד מהם הוא לנסות שלכל אלמנט בקוד תהיה מטרה אחת. למשל, כל פונקציה עושה משהו אחד בלבד. לכן החלטתי לחלק את העבודה של השרת למחלקה שמטפלת בdatabase ומחלקה שמטפלת בתקשורת, לכל אחד תפקיד מוגדר וברור.



מבנה המערכת:

שרטוט של המרכיבים העיקריים במערכת – ברמת שרת לקוח ותקשורת

מבנה Classes:

דיאגרמת class , איזה classes עיקריים במערכת ואיך הם מקושרים

פרוטוקול תקשורת:

דיאגרמת פרוטוקול כפי שלמדנו, אל מול כל תרחיש שהוצג במסמך האפיון

**פיתוח הפתרון בשכלול הקוד עם שפת התכנות**

החלק האחרון של הספר והארוך ביותר, הסבר מפורט של הפתרון ואיך הוא נראה בקוד, כאן מראים class, ומסבירים את הרציונל שלו – לא כל שורה **(את זה עושים בקוד עצמו שחייב להיות מתועד)**

בנוסף מראים את הפונקציות העיקריות והחשובות, כולל הסבר על האלגוריתם שבהן הן עובדות ואיך הוא משוקף למעשה

לדוגמא:

**ניתוב וסידור המידע בין כל המשתמשים:**

למען סידור המשתמשים, אני משתמש בClass של משתמש:

# ----- IMPORTS -----

import threading

from socket import \*

# ----- CONSTANTS -----

PORT\_DEST = 5050

# ----- CLASS -----

class SessionWithClient(threading.Thread):

# ----- DATA -----

#clientSock = None

def \_\_init\_\_(self, Server, clientSock, addr ):

print 5.1

threading.Thread.\_\_init\_\_(self)

print 5.2

self.Server = Server

print 5.3

self.clientSock = clientSock

print 5.4

self.addr = addr

print 5.5

# self.Server = self.proxyServer.connectToServer(addr[0],addr[1])

def fromClient(self):

data = self.clientSock.recv(4096)

return data

def run(self):

request = self.fromClient()

print "From Client :----------------------------"

print request

def SendData(self,data): #step 4

self.clientSock.send(data)

ופעולה המשתמשת בClass כדי לחבר בין המשתמשים:

def WorkWithClients(self):

listenerSock = socket(AF\_INET, SOCK\_STREAM)

listenerSock.bind(LISTENER\_ADDR)

listenerSock.listen(5)

while 1:

clientSock, address = listenerSock.accept()

addr = address[0]

sessionWithClient = SessionWithClient(self, clientSock, addr )

self.open\_clients.append(sessionWithClient)

print self.open\_clients

sessionWithClient.start()

client\_exists = False

for item in self.cdatabase.DB:

if addr == item[0]:

client\_exists = True

if not client\_exists:

self.tree\_data.append((addr, "None"))

self.tree.insert('', 'end', values=(addr, "None"))

self.Send()

כל משתמש שאינו קיים כבר במסד נתונים מתווסף למסד נתונים.

**סריקה של הקבצים ובדיקתם כקבצים מותרים או אסורים:**

ראשית יש רשימה של מMagic Numbers:

MAGICNUM = [("Adobe Illustrator", ".ai", "25 50 44 46"),

("Bitmap", ".bmp", "42 4D"),

("Class", ".class", "CA FE BA BE"),

("JPEG", ".jpg", "FF D8"),

("JPEG 2000", ".jp2", "00 00 00 0C 6A 50 20 20 0D 0A"),

("GIF", ".gif", "47 49 46 38"),

("TIF", ".tif", "49 49"),

("PNG", ".png", "89 50 4E 47"),

("Photoshop", ".psd", "38 42 50 53"),

("Windows Meta File", ".wmf", "D7 CD C6 9A"),

("MIDI", ".mid", "4D 54 68 64"),

("Icon", ".ico", "00 00 01 00"),

("MP3", ".mp3", "49 44 33"),

("AVI", ".avi", "52 49 46 46"),

("Shockwave", ".swf", "46 57 53"),

("Flash", ".flv", "46 4C 56"),

("Mpeg 4", ".mp4", "00 00 00 18 66 74 79 70 6D 70 34 32"),

("MOV", ".mov", "6D 6F 6F 76"),

("Windows Video file", ".wmv", "6D 6F 6F 76"),

("Windows Audio file", ".wma", "30 26 B2 75 8E 66 CF"),

("Zip", ".zip", "50 4B 05 06"),

("GZip", ".gz", "1F 8B 08"),

("Tar", ".tar", "75 73 74 61 72"),

("Microsoft Installer", ".msi", "D0 CF 11 E0 A1 B1 1A E1"),

("Object Code File", ".obj", "4C 01"),

("DLL", ".dll", "4D 5A"),

("CAB Installer", ".cab", "4D 53 43 46"),

("RAR", ".rar", "52 61 72 21 1A 07 00"),

("Executable", ".exe", "4D 5A"),

("Help File", ".hlp", "3F 5F 03 00"),

("PDF", ".pdf", "25 50 44 46"),

("Office Document", ".doc, .xls, .ppt", "D0 CF 11 E0 A1 B1 1A E1"),

("Office Document(2010)", ".docx, .xlsx, .pptx", "50 4B 03 04"),

("Microsoft Database", ".mdb", "53 74 61 6E 64 61 72 64 20 4A 65 74"),

("Outlook Message File", ".msg", "D0 CF 11 E0 A1 B1 1A E1"),

("Jar", ".jar", "50 4B 03 04 14 00 08 00 08 00")]

ובנוסף 2 פעולות אשר סורקות ובודקות את הקבצים:

def match(mnum):

for i in MAGICNUM:

if mnum.startswith(i[2]):

return i

return ("Unknown", "Unknown", "Unknown")

def MagicNumScan(path, find):

pathdir = os.listdir(path)

filelist = []

filelist.append(path)

for i in pathdir:

if os.path.isfile(path +"/" + i):

try:

str1 = open(path +"/" + i,"rb").read()

hexStr=""

if len(str1) > 0:

if len(str1) > 20:

uselen = 20

else:

uselen = len(str1)

for x in range(0, uselen):

if(len(hex(ord(str1[x]))) == 3):

hexStr=hexStr+"0"+hex(ord(str1[x]))+" "

else:

hexStr=hexStr+hex(ord(str1[x]))+" "

newhexStr = hexStr.replace("0x", "")

newhexStr = newhexStr.upper()

filetype = match(newhexStr)

if(filetype[1] == find):

os.remove(path +"/" + i)

filelist.append(i + " is a " + filetype[0] + " and will be deleted")

else:

filelist.append(i + " is a " + filetype[0])

else:

filelist.append(i + " is empty and suspicious")

except:

filelist.append(i + ''' doesn't have "read" permissions''')

else:

filelist.append(MagicNumScan((path +"/" + i), find))

return filelist

**הקשבה למערכות המחשב לייצור קבצים חדשים ובדיקתם:**

ראשית יש סריקה דומה לזאת בפתרון הקודם:

def SpecificMagicNumScan(path, find):

str1 = open(path,"rb").read()

hexStr=""

if len(str1) > 0:

if len(str1) > 20:

uselen = 20

else:

uselen = len(str1)

for x in range(0, uselen):

if(len(hex(ord(str1[x]))) == 3):

hexStr=hexStr+"0"+hex(ord(str1[x]))+" "

else:

hexStr=hexStr+hex(ord(str1[x]))+" "

newhexStr = hexStr.replace("0x", "")

newhexStr = newhexStr.upper()

filetype = match(newhexStr)

if(filetype[1] == find):

os.remove(path)

ולאחר מכן, הClass אשר משתמש בHook שמשתמש בסריקה

class HookChange():

def \_\_init\_\_(self, path):

self.actions = {

1 : "Created",

2 : "Deleted",

3 : "Updated",

4 : "Renamed from something",

5 : "Renamed to something"

}

self.FILE\_LIST\_DIRECTORY = 0x0001

s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

s.connect(("gmail.com",80))

self.Own\_IP = (s.getsockname()[0])

s.close()

self.path\_to\_watch = path

self.files\_to\_watch = []

self.hDir = win32file.CreateFile (

self.path\_to\_watch,

self.FILE\_LIST\_DIRECTORY,

win32con.FILE\_SHARE\_READ | win32con.FILE\_SHARE\_WRITE | win32con.FILE\_SHARE\_DELETE,

None,

win32con.OPEN\_EXISTING,

win32con.FILE\_FLAG\_BACKUP\_SEMANTICS,

None

)

def hook(self, args, q):

exec args

files = []

for i in self.DB[0][1]:

files.append(i)

print self.DB

for i in self.DB:

if i[0] == self.Own\_IP:

for j in i[1]:

files.append(j)

for i in files:

self.files\_to\_watch.append(getEnd(i[0]))

while 1:

if not q.empty():

print "bye"

queuedata = q.get()

q.task\_done()

return

results = win32file.ReadDirectoryChangesW (

self.hDir,

1024,

True,

win32con.FILE\_NOTIFY\_CHANGE\_FILE\_NAME |

win32con.FILE\_NOTIFY\_CHANGE\_DIR\_NAME |

win32con.FILE\_NOTIFY\_CHANGE\_ATTRIBUTES |

win32con.FILE\_NOTIFY\_CHANGE\_SIZE |

win32con.FILE\_NOTIFY\_CHANGE\_LAST\_WRITE |

win32con.FILE\_NOTIFY\_CHANGE\_SECURITY,

None,

None

)

for action, file1 in results:

full\_filename = os.path.join (self.path\_to\_watch, file1)

for ftype in self.files\_to\_watch:

try:

SpecificMagicNumScan(full\_filename, ftype)

except:

print "error"

**תיאור המודולים של מערכת התוכנה**

win32con.FILE\_FLAG\_BACKUP\_SEMANTICS,

None

)

def hook(self, args, q):

exec args

files = []

for i in self.DB[0][1]:

files.append(i)

print self.DB

for i in self.DB:

if i[0] == self.Own\_IP:

for j in i[1]:

files.append(j)

for i in files:

self.files\_to\_watch.append(getEnd(i[0]))

while 1:

if not q.empty():

print "bye"

queuedata = q.get()

q.task\_done()

return

results = win32file.ReadDirectoryChangesW (

self.hDir,

1024,

True,

win32con.FILE\_NOTIFY\_CHANGE\_FILE\_NAME |

win32con.FILE\_NOTIFY\_CHANGE\_DIR\_NAME |

win32con.FILE\_NOTIFY\_CHANGE\_ATTRIBUTES |

win32con.FILE\_NOTIFY\_CHANGE\_SIZE |

win32con.FILE\_NOTIFY\_CHANGE\_LAST\_WRITE |

win32con.FILE\_NOTIFY\_CHANGE\_SECURITY,

None,

None

)

for action, file1 in results:

full\_filename = os.path.join (self.path\_to\_watch, file1)

for ftype in self.files\_to\_watch:

try:

SpecificMagicNumScan(full\_filename, ftype)

except:

print "error"

בשביל להריץ את התוכנה ואת האלמנטים השונים בה, המערכת משתמשת בכמה מודלים. להלן פירוט של מודלי המערכת:

* server.py

מחלקת השרת אשר ימצא על המחשב של מנהל המערכת (system administrator). מתקשר עם לקוחות המערכת.

* client.py

מחלקה המייצגת את שאר המחשבים במערכת - הלקוחות אשר מתקשרים עם השרת.

* database.py

מחלקה אשר אחראית על אחראית על עדכון הנתונים בdatabase.

* snmpServer.py

מודל אשר מאפשר לכל לקוח ברשת לקבל מידע שונה עליו, כגון הperformance שלו, מידע על הusers השונים על המחשב, מידע על הprocesses השונים הרצים על המחשב, ועוד.

* up\_to\_dater.py

מודל אשר מאפשר ללקוח לעדכן תוכנות שונות שעל המחשב שלו.

* WOL.py

מודל אשר מאפשר שימוש בWOL – wake on lan, הדלקה מרחוק של מחשבים.

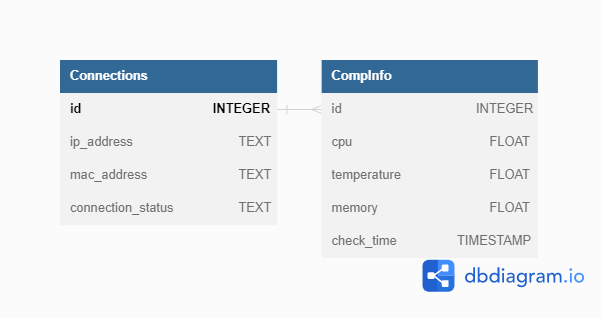
* Server\_GUI.py

מחלקה היוצרת את ממשק המשתמש של השרת.

**תיעוד הקוד**

**פירוט מבניי הנתונים:**

|  |  |
| --- | --- |
| **שם המבנה** | **תפקיד המבנה** |
| Ipconnections.db | מכיל מידע שונה אודות הלקוחות ברשת. מסד הנתונים הזה מכיל שתי טבלאות. אחת מהן – Connections, מכילה id, את כתובות הip של הלקוחות, כתובות הmac שלהם ואת סטטוס החיבור שלהם (“off”/“on”). השניה - CompInfo,מכילה id שמקשר אותה לטבלה הראשונה, מכילה מידע אודות הביצועים של המחשב – cpu, temperature וmemory. בנוסף היא מכילה TIMESTAMP של הזמן בו המדידה התבצעה. |



**פירוט הפעולות ותפקידן:**

**server.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def \_\_init\_\_ (self) | מקבלת את מחלקת השרת | - | בנאי, קורא לפעולות |
| def CreateBody(self) | מקבלת את מחלקת הממשק | - | יוצרת את גוף הממשק |
| def CreateButtons(self) | מקבלת את מחלקת הממשק | - | יוצרת את הכפתורים של הממשק |
| def WorkWithClients(self) | מקבלת את מחלקת הממשק | - | מוסיפה משתמשים לטיפול אצל השרת |
| def Send(self) | מקבלת את מחלקת הממשק | - | שולחת את הנתונים למשתמשים |
| def CreateRule(self) | מקבלת את מחלקת הממשק | - | פותחת את הממשק של מתן ההרשאות ונותנת לו את הIP של המשתמש הנבחר |

**RuleGUI.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def CreateHeader(self) | מקבלת את מחלקת הממשק | - | יוצרת את ראש הממשק |
| def CreateBody(self) | מקבלת את מחלקת הממשק | - | יוצרת את גוף הממשק |
| def CreateButtons(self) | מקבלת את מחלקת הממשק | - | יוצרת את הכפתורים של הממשק |
| def Add(self) | מקבלת את מחלקת הממשק | - | מוסיפה מרשימת הקבצים האופציונאליים אל רשימת הקבצים הנבחרים |
| def Remove(self) | מקבלת את מחלקת הממשק | - | מסירה מרשימת הקבצים הנבחרים ומחזירה לרשימת הקבצים האופציונאליים |
| def Ok(self) | מקבלת את מחלקת הממשק | - | סוגרת את החלון ומעדכנת את מבנה הנתונים DB |

**database.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def match(mnum) | מקבלת מספר זיהוי MagicNumber | מחזירה את סוג הקובץ | מגלה את סוג הקובץ לפי הMagicNumber שלו |
| def getEnd(name) | מקבלת שם של סוג של קובץ | מחזיר את הסיומת של הקובץ | מביאה את הסיומת של סוג הקובץ לפי שם סוג הקובץ |
| def SpecificMagicNumScan(path, find) | מקבלת מיקום של קובץ, וסוג קובץ | - | בודקת אם הקובץ מותר ומוחקת אותו אם לא |
| def MagicNumScan(path, find) | מקבלת תיקיה וסוג קובץ | - | סורקת את התיקייה ובודקת כל קובץ אם הוא מותר ומוחקת אותו אם לא |
| def printTreeDir(li, count) | מקבלת רשימת log על הסריקה | - | מדפיסה את log הסריקה בצורת עץ |

**ClientDatabase.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def updatedatabase(self, new\_list) | מקבלת את מחלקת הממשק ורשימה חדשה של מבנה נתונים | - | מחליפה את מבנה הנתונים הקיים במבנה חדש |

**SessionWithClient.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def fromClient(self) | מקבלת את המחלקה | מחזירה את הנתונים שקיבלה מהמשתמש | מקבלת מידע מן המשתמש |
| def run(self) | מקבלת את המחלקה | - | מתחברת למשתמש |
| def SendData(self,data) | מקבלת את המחלקה ונתונים לשליחה | - | שולחת את המידע המבוקש למשתמש |

**Scrambler.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def Decode(self) | מקבלת את המחלקה | מחזירה את מבנה הנתונים לאחר פענוח | מפענחת את המבנה המוצפן |
| def Encode(self, new\_data) | מקבלת את המחלקה, ומבנה חדש | - | מצפינה את המבנה החדש ושומרת אותו |

**threadclass.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def run(self) | מקבלת את המחלקה | - | מפעילה את התהליכון של המחלקה |

**Client.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def scan(filetypes) | מקבלת רשימה של הקבצים אשר יש לסרוק | - | קוראת לפעולת הסריקה מ-database.py |
| def listen(args) | מקבלת את הערכים הנתונים על ידי התהליכון | - | קוראת לפונקציה scan עם הערכים ומחזירה לשרת תשובה על סיום התהליך |
| def (args) | מקבלת את הערכים הנתונים על ידי התהליכון | - | קוראת לפעולת הhook מ-HookChange.py |

**HookChange.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def hook(self, args, q) | מקבלת את המחלקה, את הערכים של הקבצים, ותור(לבדיקת יציאה) | - | עושה את הפניה API על התיקייה הנדרשת, לשינויים וקוראת לפונקציית הסריקה כדי לבדוק את הקבצים אשר השתנו |

**השוואת העבודה עם פתרונות ויישומים קיימים**

השוואה לפתרונות קיימים מה ההבדל בסופו של דבר...

**הערכת הפתרון לעומת התכנון והמלצות לשיפורו**

מה הצלחתם בסוף, האם אתם מרוצים ומה עוד כדאי לשפר בפרוייקט בעתיד

**תיאור של הממשק למשתמש – הוראות הפעלה**

**הסבר – איך עובדים עם התוכנה , זה ממש manuel של הפרוייקט שלכם**

**לדוגמא:**

\* הוראות ההפעלה נכתבו בלשון זכר בשל טעמי נוחות בלבד.

שלום מנכ"ל נכבד,

באמצעות התוכנה הזו תוכל לצמצם את הסחות הדעת של העובדים שלך בכך  
שתבחר אלו קבצים תיתן להם לשמור על המחשב ואלו לא.

תרצה להתקין את תוכנת השרת על השרת כך שיישאר דלוק בין אם המחשבים האחרים דלוקים ובין אם לא. בנוסף לכך תרצה להתקין ולהפעיל את השרת לפני שתתקין ותפעיל את המשתמשים.

כאשר השרת מותקן ומופעל, יפתח חלון בעל רשימה של המחשבים המחוברים וככל שמתחברים עוד מחשבים, הרשימה גדלה.  
תוכל לעשות 2 הגדרות:

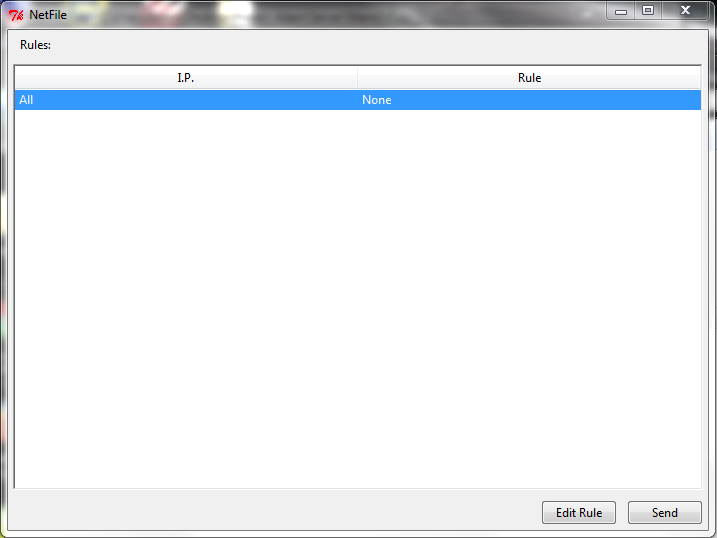
* כדי להגדיר הרשאות על כל המחשבים, תבחר באפשרות ALL ותלחץ על הכפתור Edit Rule.
* כדי להגדיר הרשאות על מחשב ספציפי, תבחר את הIP שלו ותלחץ על הכפתור Edit Rule.

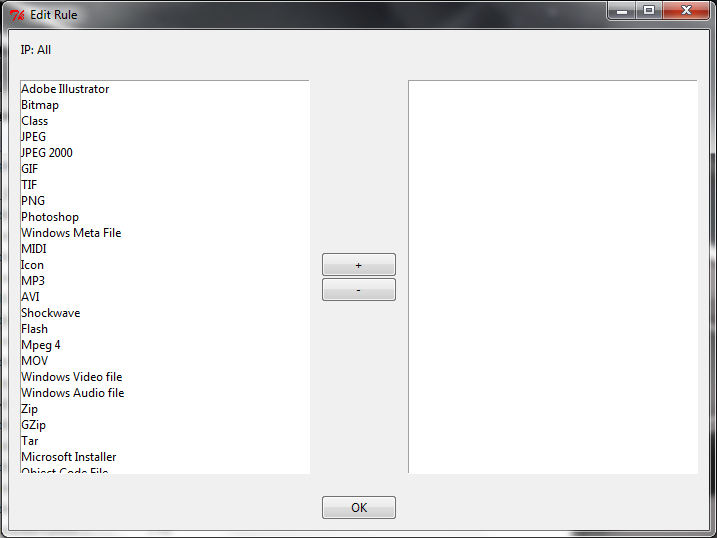
כאשר תלחץ על Edit Rule, יפתח חלון עם אפשרויות איסור. תלחץ על אפשרות ועל כפתור ה'+' כדי להעביר את האפשרות לרשימת הקבצים הנבחרים. תלחץ על כפתור ה'-' כדי להסירו מרשימת הקבצים הנבחרים ולהחזירו לרשימת האפשרויות. כאשר סיימת לבחור הרשאות, לחץ על כפתור ה'Ok'.

כאשר סיימת לבחור הרשאות לכל המחשבים כרצונך, לחץ על כפתור ‘Send’ כדי לשלוח את הנתונים למשתמשים, והתוכנה תעשה את השאר בשבילך.

בהצלחה.

ממשק השרת:





**מבט אישי על העבודה ותהליך הפיתוח**

איך ראיתם באופן אישי את הפיתוח מה לקחתם איתכם ומה הייתם מוותרים

**ביבליוגרפיה**

חומר חיצוני שהיה בשימוש, ספרים, אתרים, מאמרים, כתבות בעיתון ועוד.

**קוד התוכנית:**

**Copy paste of the code**