תיק פרויקט גמר
תיקייה מסונכרנת
במספר מחשבים

מוטי פרנסיס 201463114
תומר מתתיהו 305705618

תוכן עניינים

מבוא

בפרויקט זה בנינו את תוכנת SyncIt. זוהי תוכנה המאפשרת סנכרון קבצים ותיקיות מלא בין מספר עמדות המחוברות לרשת (מקומית או גלובאלית).

בתוכנה מתבצע גיבוי נתונים על העמדה המקומית לצורך יכולת עבודה על הקבצים גם במצב בו העמדה לא מחוברת לרשת (עבודה במצב לא מקוון).

עם הפעלת התוכנה בפעם הראשונה יצטרך המשתמש לבחור את הנתיב של התיקייה המסונכרנת ולעדכן בקובץ ההגדרות את כתובת ה-IP של השרת והאם הוא יתפקד בתור שרת או לקוח (עמדה אחת בלבד יכול לתפקד כשרת).

סמל התוכנה יופיע בסרגל היישומים ודרכו יהיה ניתן לבצע פעולות שונות בתוכנה (פתיחת קובץ Log, בחירת נתיב חדש לתיקייה המסונכרנת, פתיחת התיקייה המסונכרנת ויציאה מהתוכנה).

בזמן שהתוכנה פועלת על עמדה מסוימת כל השינויים שיבוצעו על הקבצים בעמדה זו יסונכרנו לשרת ודרכו לשאר העמדות המחוברות לרשת.

השינויים הנתמכים ע"י התוכנה:

יצירת קובץ חדש או תיקיה חדשה, שינוי שם של קובץ או תיקיה, העברת קובץ או תיקיה למקום חדש, עריכת קובץ קיים ומחיקת קובץ או תיקיה.

התוכנה רצה ברקע מבלי להפריע לעבודת המשתמש על המחשב, כך שהמשתמש מעדכן את הנתונים בתיקיה שנבחרה וכל תהליך הסנכרון מול העמדות האחרות מתבצע כל הזמן ללא התערבות או צורך במתן פקודות של המשתמש.

ניתן לראות מעקב על כל השינויים שהתרחשו בקובץ Log ייעודי דרך האפשרויות בסמל התוכנה בסרגל היישומים ועל ידי התראות קופצות הנותנות מידע על תהליכים בתיקייה המסונכרנת.

דרישות מערכת

המערכת מותאמת לעבודה בסביבת Linux.

בכדי שיתבצע סנכרון באופן שוטף יש לחבר את העמדה לרשת (מקומית / אינטרנט).

במקרה של חיבור עמדה ללא גישה לרשת יהיה ניתן לעבוד על הקבצים הקיימים באופן לא מקוון וברגע ההתחברות לרשת יתבצע הסנכרון עם שאר העמדות.

הוראות שימוש

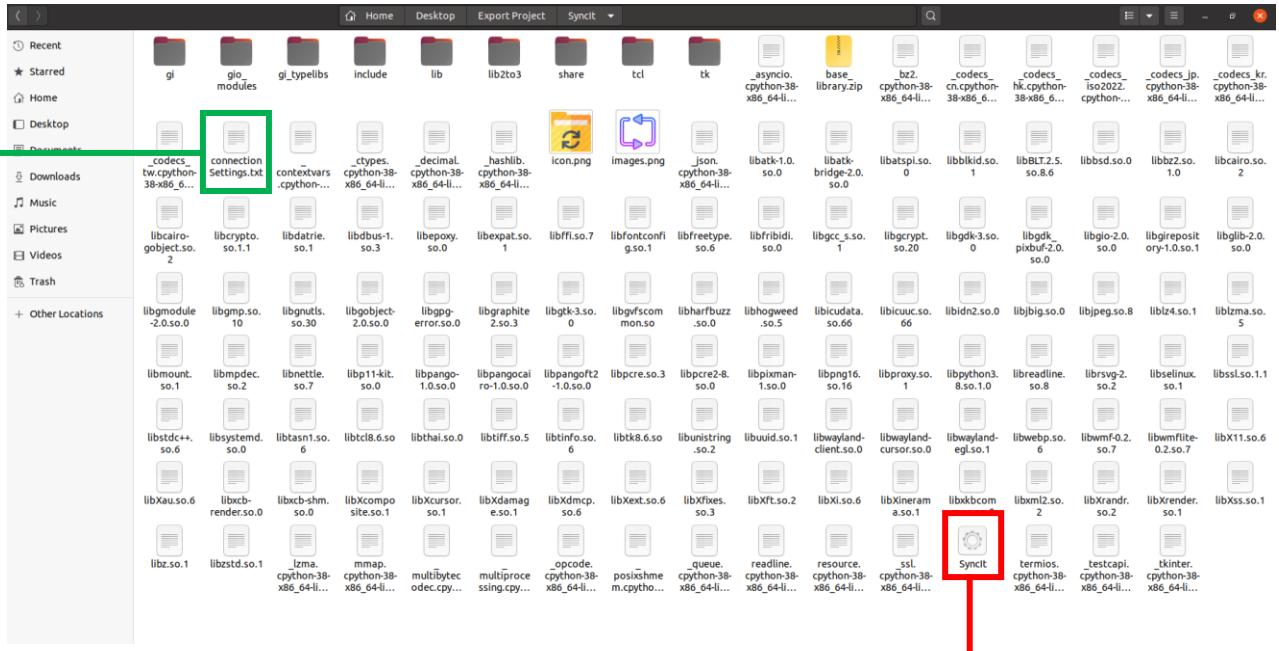
הפעלת Server / Client

התוכנה הומרה לקובץ הרצה ע"י PyInstaller שיוצר קובץ הרצה בספריה אחת המאגדת בתוכה את כל הספריות הדרושות ואת כל הקבצים הדרושים להפעלת התוכנה.

הרצת התוכנה מפעילה BootLoader עם סביבת עבודה Python 3.8 כך שאין צורך בגרסת ה – Python שאיתה יצרנו את התוכנה והיא מגיעה ביחד עם קובץ ההרצה.

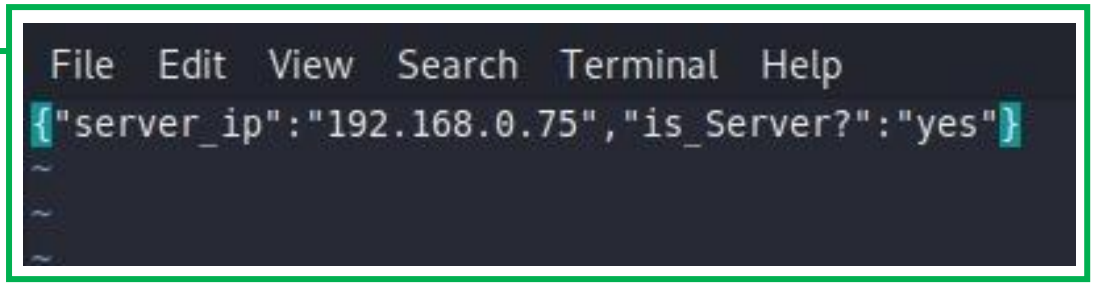
לכן הפעלת Server / Client היא פעולה פשוטה של פתיחת קובץ ההרצה.

בנוסף, טרם ההרצה הראשונית יש לעדכן את קובץ הטקסט connectionSettings.txt שנמצא באותה תיקייה עם קובץ ההרצה ובו לכתוב האם מדובר בשרת ומהי כתובת ה –IP של השרת.



קובץ ההרצה

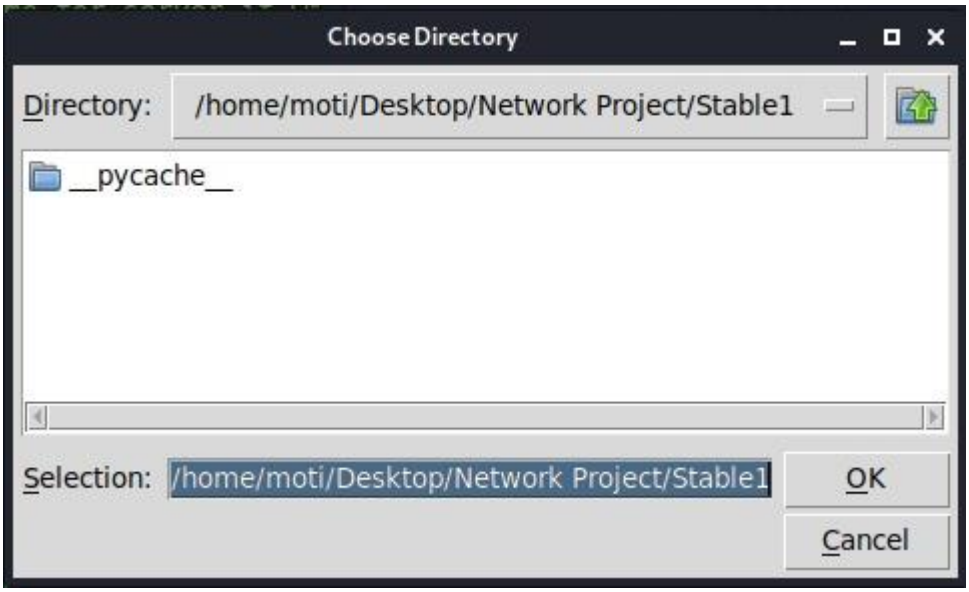
קובץ ההגדרות connectionSettings.txt



הגדרת תיקייה מסונכרנת

בפתיחת התוכנית ייפתח חלון ובו יש לבחור את התיקייה שנרצה שתהיה התיקייה המסונכרנת.

אם לא ייבחר נתיב, התוכנית תסתיים.



שינוי נתיב תיקייה מסונכרנת

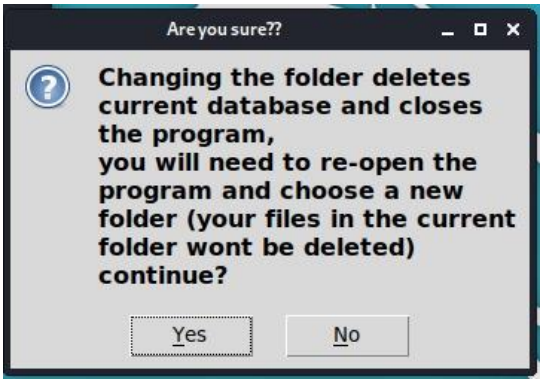
לצורך שינוי נתיב התיקייה המסונכרנת יש ללחוץ מקש ימני על אייקון התוכנה בסרגל היישומים

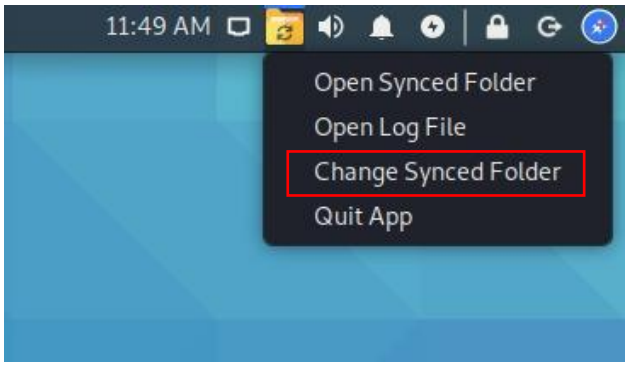
ולבחור באפשרות "Change Synced Folder".

תוקפץ הודעה המתריעה על תהליך השינוי שכולל בתוכו מחיקת קובץ הנתונים השמור (לא מתבצעת מחיקת קבצים בתיקייה הנוכחית).

אם המשתמש בחר ב"כן" התוכנית תסתיים והוא יצטרך לפתוח אותה מחדש ולבחור את הנתיב החדש לתיקייה המסונכרנת ולהמתין לסנכרון הנתונים מול השרת.

ניתן לבחור תיקייה שבה קיימים כבר קבצים.



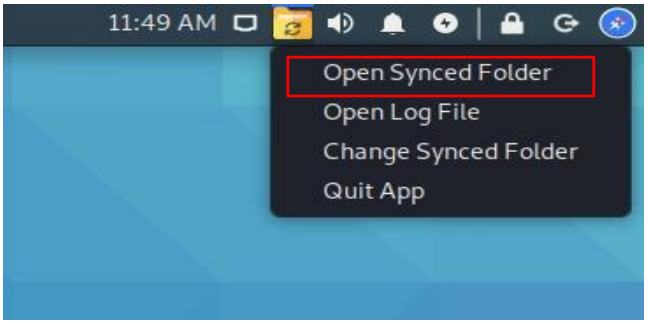


פתיחת התיקייה

המסונכרנת

יש ללחוץ מקש ימני על אייקון התוכנה בסרגל היישומים ולבחור באפשרות

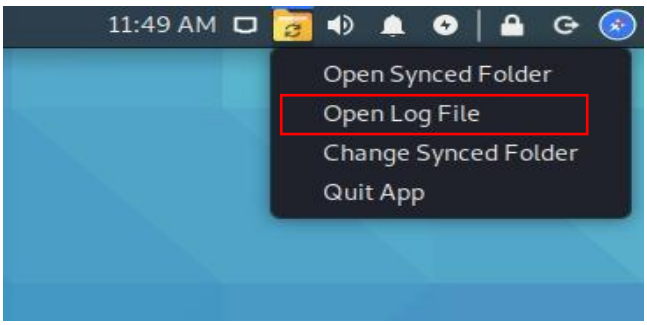
"Open Synced Folder".



קבלת קובץ Log

יש ללחוץ מקש ימני על אייקון התוכנה בסרגל היישומים ולבחור באפשרות "Open Log File ".

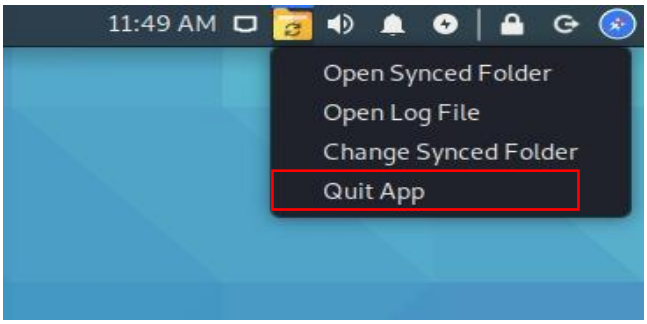
ייפתח קובץ טקסט (.txt) ובו פירוט כל הפעולות שנעשו בתוכנה (התחברות, התנתקות, שינויים בקבצים וכו').



יציאה מהתכנית

כדי לצאת מהתוכנית יש ללחוץ מקש ימני על אייקון התוכנה בסרגל היישומים ולבחור באפשרות

"Quit App".



ארכיטקטורת תוכנה

הסבר כללי

- התוכנה הומרה לקובץ הרצה ע"י PyInstaller שיוצר קובץ הרצה יחיד ומאגד בתוכו את כל הספריות הדרושות ואת כל הקבצים הדרושים להפעלת התוכנה.
  הרצת התוכנה מפעילה BootLoader עם סביבת עבודה Python 3.8 כך שאין צורך בגרסת ה – Python שאיתה יצרנו את התוכנה והיא מגיעה ביחד עם קובץ ההרצה.
- התוכנה נבנתה בשפת Python בתכנות בשיטת Object Oriented Programming.
- העברת הודעות והקבצים מתבצעת בפרוטוקול TCP לצורך שמירה על אמינות העברת המידע המועבר.
- העברת כל המידע משרת אחד לשאר הלקוחות – תצורת כוכב.
- Multi-Threading - עבודה עם תהליכונים לצורך יכולת עבודה מקבילית בקבלת / שליחת קבצים, סריקת התיקייה וכו'.
- GUI – אייקון בסרגל היישומים, התראות קופצות (PopUp) על שינויים בתיקייה המסונכרנת ותפריט עם פונקציות.

מבנה ההודעות שנשלחות

כל הודעה שנשלחת מהשרת ללקוח או להפך היא אובייקט מסוג Command המכיל את הפקודה הרלוונטית לשליחה. נבנה HEADER בגודל 64 בתים המכיל את גודל הפקודה הנשלחת.

ראשית נשלח ה – HEADER כדי שהצד המקבל יידע את גודל המידע שהוא אמור לקבל ואחריו נשלחת ההודעה עצמה המכילה את הפקודה.

האובייקט נשלח ע"י פירוק המבנה שלו לצורה הניתנת לשליחה (באמצעות pickle) ובקבלתה בצד השני היא נבנית מחדש (גם כן ע"י pickle) ומבוצעת בהתאם לפקודה.

האובייקט Command מקבל את הפרמטרים הבאים:

Command - שם הפקודה.

aboutFile - משמש ליכולת הגדרה לצד המקבל לאיזה קובץ הפקודה נוגעת.

Path - נתיב הקובץ.

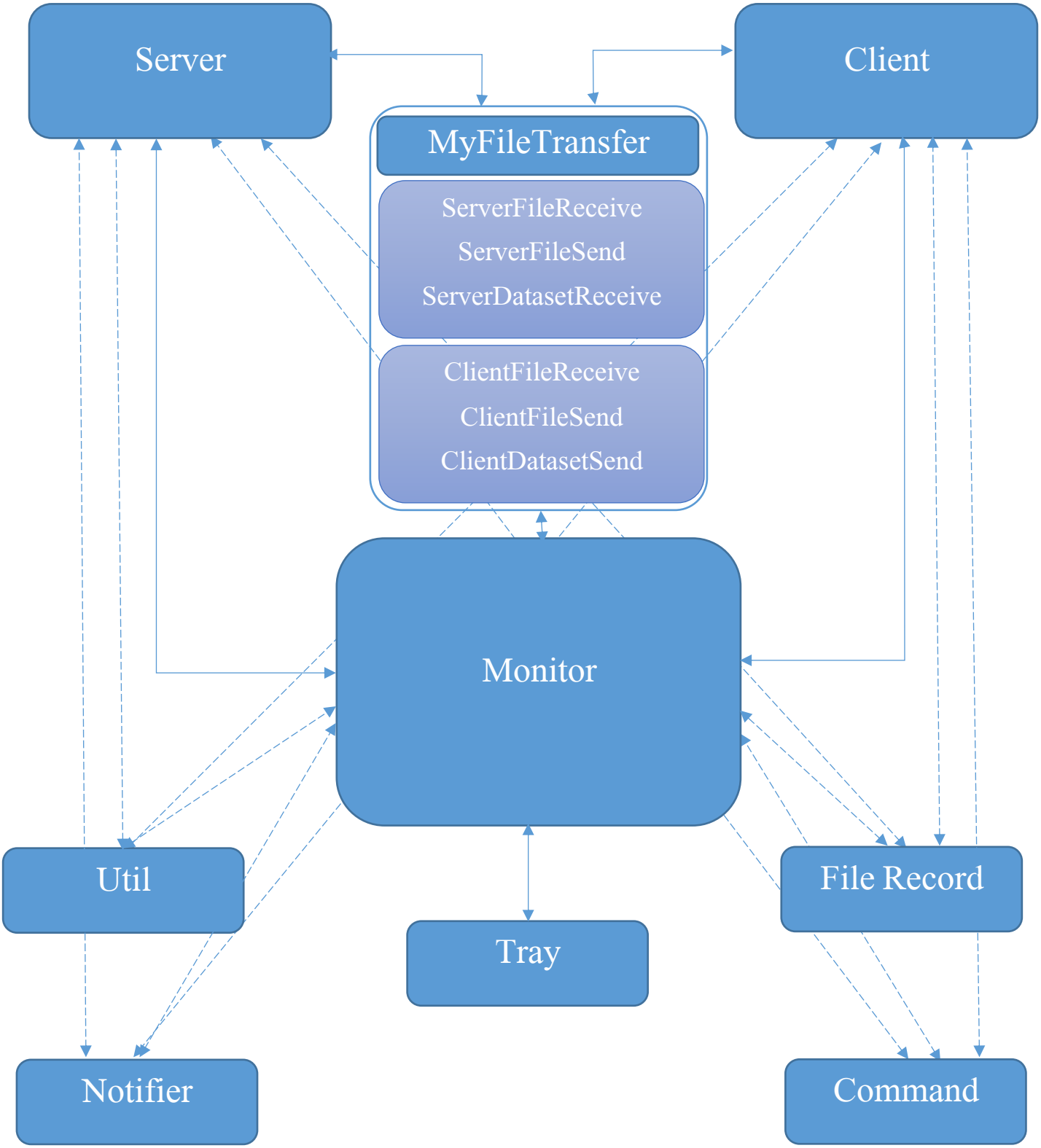
recordID - מזהה ייחודי של הקובץ (Hash על זמן יצירת הקובץ).

Port – הפורט בו תתבצע התקשורת.

moveTo – הנתיב החדש של הקובץ לאחר השינוי.

renameTo – השם החדש של הקובץ לאחר השינוי.

מבנה המחלקות



תהליך ניטור השינויים בתיקייה

התיקייה נסרקת ע"י Thread ייעודי הפועל באופן מחזורי כל 5 שניות.

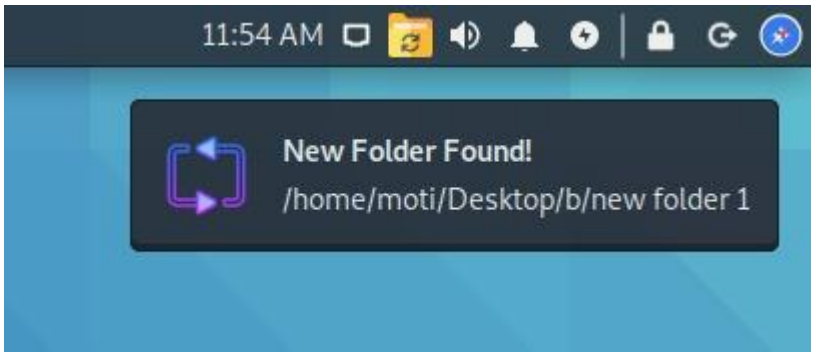
מתבצעת סריקה של תת התיקיות באופן רקורסיבי ע"י פונקציית הסריקה.

כל קובץ משווה למסד הנתונים השמור במערכת כדי לדעת אם בוצע שינוי כלשהו בקובץ.

לאחר הסריקה מתבצעת סריקה הפוכה שבודקת את מסד הנתונים מול הקבצים הקיימים בפועל (כדי לזהות קבצים שנמחקו).

- **<u>יצירת קובץ חדש:</u>**

אם מדובר ביצירת תיקייה חדשה נשלחת פקודת MKDIR אשר מכילה את נתיב התיקייה שנוצרה ואת השם שלה.
התוכנה יוצרת בנתיב שהתקבל תיקייה חדשה עם השם שהתקבל.
אם מדובר ביצירת קובץ חדש בשרת אז הוא נשלח לשאר הלקוחות בכדי לשמור על הסנכרון.
אם מדובר ביצירת קובץ חדש אצל אחד הלקוחות אז הוא נשלח לשרת ומשם לשאר הלקוחות (חוץ מהלקוח שיצר את הקובץ).
לאחר יצירת הקובץ / התיקייה מתבצע עדכון במסד הנתונים, מוצגת התראה למסך שנוצר קובץ חדש או תיקייה חדשה והיצירה נרשמת בקובץ ה – Log.



- **<u>שינוי שם קובץ:</u>**

במקרה של שינוי שם קובץ נשלחת פקודת RENAME אשר מכילה את נתיב הקובץ והשם החדש לאחר השינוי.
התוכנה מזהה את הקובץ ומעדכנת את השם שלו ואת הנתיב החדש עם השם החדש.
לאחר השינוי מתבצע עדכון השינוי גם במסד הנתונים, מוצגת התראה למסך ששם הקובץ שונה והשינוי נרשם בקובץ ה – Log.
בנוסף, פקודת RENAME נשלחת מהשרת לכל הלקוחות כדי לשמור על הסנכרון. אם השינוי בוצע אצל אחד הלקוחות הוא לא יקבל חזרה פקודת שינוי אלא רק שאר הלקוחות.

- **שינוי מיקום קובץ:**

במקרה של שינוי מיקום קובץ נשלחת פקודת MOVE אשר מכילה את נתיב הקובץ הישן והנתיב החדש לאחר השינוי.
התוכנה מזהה את הקובץ ומעדכנת את הנתיב שלו החדש עם הנתיב החדש.
לאחר השינוי מתבצע עדכון השינוי גם במסד הנתונים, מוצגת התראה למסך שמיקום הקובץ שונה והשינוי נרשם בקובץ ה – Log.
בנוסף, פקודת MOVE נשלחת מהשרת לכל הלקוחות כדי לשמור על הסנכרון. אם השינוי בוצע אצל אחד הלקוחות הוא לא יקבל חזרה פקודת שינוי אלא רק שאר הלקוחות.

- **עדכון קובץ:**

עדכון קובץ מזוהה בפונקציית הסריקה לאיתור שינויים ע״י מציאת שינוי בזמן עריכת הקובץ.
התוכנה מזהה את הקובץ שעודכן ומבצעת שתי פעולות – ראשית מחיקת הקובץ הלא מעודכן ואז שליחת הקובץ המעודכן.
לאחר מכן מתבצע עדכון השינוי גם במסד הנתונים, מוצגת התראה למסך שהקובץ עודכן והשינוי נרשם בקובץ ה – Log.
בנוסף, הפעולות הנ״ל מבוצעות ע״י השרת עבור כל הלקוחות כדי לשמור על הסנכרון. אם השינוי בוצע אצל אחד הלקוחות הוא לא יקבל חזרה את פעולות השינוי אלא רק שאר הלקוחות.

- **מחיקת קובץ:**

במקרה של מחיקת קובץ נשלחת פקודת DELETE אשר מכילה את נתיב הקובץ ובמקרה של מחיקת תיקייה נשלחת פקודת DELETE_DIR אשר מכילה את נתיב התיקייה.
אם מדובר בקובץ, התוכנה מזהה אותו ומוחקת אותו.
אם מדובר בתיקייה, התוכנה מזהה אותה ומוחקת אותה ואת כל הקבצים שהיא מכילה.
לאחר המחיקה התוכנה מעדכנת את המחיקה במסד הנתונים, מציגה התראה למסך שהקובץ או התיקייה נמחקו והמחיקה נרשמת בקובץ ה – Log.
בנוסף, פקודת DELETE / DELETE_DIR נשלחת בהתאם מהשרת לכל הלקוחות כדי לשמור על הסנכרון. אם המחיקה בוצעה אצל אחד הלקוחות הוא לא יקבל חזרה פקודת מחיקה אלא רק שאר הלקוחות.

שליחת וקבלת קבצים

שליחת וקבלת קבצים מתבצעות רק במקרה של יצירת קובץ חדש או עדכון קובץ קיים.

כלומר במקרה של שינוי שם קובץ, שינוי מיקום קובץ ומחיקת קובץ נעדכן את פרטי הקובץ ולא נשלח אותו כל פעם מחדש, בכך נייעל את מהירות עבודת התוכנה.

- **צד שרת:**
  - שליחת קבצים:

שליחת קובץ מהשרת ללקוח מתבצעת ע"י יצירת socket חדש לצורך שליחת הקובץ (עבור כל לקוח נפתח socket נפרד).
ה – socket נוצר עם timeout של 60 שניות וממתין להתחברות הלקוח אליו. במידה ולא מתבצעת בזמן זה התחברות ה – socket נסגר.
השרת בוחר באופן אקראי פורט בטווח 30000-40000 ופותח תהליכון חדש עבור התקשורת עם הלקוח.
לאחר התחברות הלקוח, השרת שולח את הקובץ בחתיכות (chunks) של 2048Byte.

  - קבלת קבצים:

קבלת קובץ בשרת מהלקוח מתבצעת בשני חלקים – ראשית השרת מקבל פקודת העברת קובץ מהלקוח. השרת יוצר socket חדש לטובת קבלת הקובץ ושולח פקודה נוספת ללקוח שמכילה את פרטי ה – socket ומסמנת לו שניתן להתחיל את ההעברה.
ה – socket נוצר עם timeout של 60 שניות וממתין להתחברות הלקוח אליו. במידה ולא מתבצעת בזמן זה התחברות ה – socket נסגר.
השרת בוחר באופן אקראי פורט בטווח 30000-40000 ופותח תהליכון חדש עבור התקשורת עם הלקוח.
לאחר התחברות הלקוח, השרת מקבל את הקובץ בחתיכות (chunks) של 2048Byte.

- **צד לקוח:**
  - שליחת קבצים:

שליחת קובץ מהלקוח לשרת מתבצעת ע"י שליחת בקשה של הלקוח לשרת לצורך פתיחת socket עבור שליחת הקובץ וקבלת פרטי ההתחברות.
לאחר שהשרת פותח socket עבור הלקוח הוא שולח לו את הנתונים הבאים – כתובת IP ופורט לצורך החיבור ואת הנתיב של הקובץ אותו צריכים לקבל.
לאחר מכן, הלקוח מתחבר לשרת על פי הפרטים שהתקבלו ולאחר החיבור הלקוח שולח לשרת את הקובץ בחתיכות (chunks) של 2048Byte.

  - קבלת קבצים:

קבלת קובץ בלקוח מהשרת מתבצעת ע"י קבלת נתונים מהשרת – כתובת IP ופורט לצורך החיבור ואת הנתיב של הקובץ אותו צריכים לקבל.
לאחר מכן, הלקוח מתחבר לשרת על פי הפרטים שהתקבלו ולאחר החיבור הלקוח מקבל את הקובץ בחתיכות (chunks) של 2048Byte.

עבודה במצב לא מקוון

אחת מיכולות התוכנה היא עבודה על מחשב לקוח באופן לא מקוון.

מטרתה היא מתן יכולת לעבוד על הקבצים שבתיקייה המסונכרנת באופן מקומי ללא גישה לרשת (למשל בנסיעות או במקום ללא גישה לרשת) וסנכרון מחדש ברגע שמתחברים מחדש לרשת.

דבר זה מתבצע ע"י שמירת נתוני התקשורת עם השרת (כתובת ה – IP) בקובץ ייעודי אצל הלקוח.

ברגע של ניתוק מהרשת התוכנה מנסה להתחבר מחדש לשרת בכל 5 שניות.

ברגע של חיבור מחדש לרשת, הלקוח שולח לשרת פקודת RECONNECT אשר מגדירה לשרת שמדובר בחיבור חוזר ולא חדש, ולכן אין צורך בשליחת כל הקבצים של התיקייה המסונכרנת אלא רק בדיקה שלהם ועדכון הקבצים הרלוונטיים.

דבר זה מתבצע על ידי פונקציה ייעודית שמטרתה לסרוק את המידע של הקבצים של הלקוח מול המידע שבמסד הנתונים וחלוקת הקבצים והתיקיות למספר מערכים – מערך עבור שליחה, מערך עבור מחיקה, מערך עבור עריכה, מערך עבור שינוי שם ומערך עבור שינוי מקום.

אם מצאנו קובץ תואם בין התיקייה של הלקוח למסד הנתונים נבדוק אם הוא נערך ע"י בדיקת מזהה ה-Hash שלו. אם כן, נכניס אותו למערך עבור עריכה.

אם מצאנו קובץ תואם בעל נתיב שונה אז או ששונה השם שלו או ששונה המקום שלו.

אם השם זהה – שונה לו המקום ולכן נכניס אותו למערך עבור שינוי מקום, אחרת נכניס אותו למערך עבור שינוי שם.

אם לא מצאנו קובץ תואם אז הוא נמחק ויש להכניסו למערך עבור מחיקה.

כל שאר הקבצים שנמצאים במסד הנתונים אבל לא נמצאים בתיקייה הם קבצים חדשים שיש ליצור ולכן נכניס אותם למערך עבור יצירה.

לבסוף נעבור על כל המערכים ועבור כל קובץ נשלח לשרת את הפקודה המתאימה לו.

תיאור המחלקות והפונקציות

המחלקה Client:

מחלקה שאחראית על פעולות התקשורת של מחשבי הקליינט עם השרת. היא שולחת ומקבלת פקודות ומטפלת בהן.

שם הפונקציה	תיאור הפונקציה
init	בנאי של המחלקה. מעדכן את כתובת ה-IP שלו (אם קיים קובץ הגדרות מוציא ממנו ואם לא יוצר קובץ חדש עם כתובת ברירת המחדל שלו), מעדכן בקובץ הגדרות שמדובר בקליינט ומגדיר פורט לחיבור.
connect	פונקציית החיבור לשרת. משמשת לפתיחת חיבור מול השרת – חיבור ראשוני או חיבור מחדש לאחר התנתקות מהרשת.
startClient	פונקציה לפתיחת תהליכונים עבור הודעות תקשורת עם השרת.
sendCommand	פונקציה המשמשת לשליחת אובייקט פקודה (המכיל את כל פרטי הפקודה).
getMassage	פונקציה זו מפעילה תהליכון נפרד כדי לקבל הודעות מהשרת ולשלוח אותן לפונקציית הטיפול בפקודות.
handelCommand	פונקציה לטיפול בפקודות שמתקבלות. ברגע שמתקבלת אחת מהפקודות שהוסברו קודם היא מבצעת את הטיפול הנדרש (יצירת קובץ, שינוי שם, שינוי מיקום קובץ וכו').

המחלקה FileRecord:

מחלקה זו מגדירה אובייקט מסוג FileRecord שנוצר עבור כל קובץ שקיים בתיקיה המסונכרנת ומכיל בתוכו את כל התכונות של הקובץ. אובייקטים אלה נשמרים בבסיס הנתונים הראשי לצורך השוואת תכונות מול קבצים אחרים בתיקיה.

שם הפונקציה	תיאור הפונקציה
init	בנאי של המחלקה. מגדיר את כל התכונות של האובייקט - שם, נתיב, מזהה קובץ ייחודי, גודל, מס' קובץ, האם תיקיה או קובץ וזמני יצירת / שינוי הקובץ.
printData	הדפסת תכונות הקובץ.

המחלקה Server :

מחלקה שאחראית על פעולות התקשורת של השרת עם הלקוחות. מול כל לקוח היא שולחת ומקבלת פקודות ומטפלת בהן.

השרת מחזיק רשימה של כל חיבורי הלקוחות הפעילים מולו ומעדכן את כולם בכל שינוי שמתבצע בו או בכל לקוח אחר.

שם הפונקציה	תיאור הפונקציה
init	בנאי של המחלקה. מעדכן את כותבת ה-IP שלו (אם קיים קובץ הגדרות מוציא ממנו ואם לא יוצר קובץ חדש עם כתובת ברירת המחדל שלו), מעדכן בקובץ הגדרות שמדובר בשרת ומגדיר פורט לחיבור.
sendCommand	פונקציה המשמשת לשליחת אובייקט פקודה (המכיל את כל פרטי הפקודה). השליחה מתבצעת לכל החיבורים הפעילים.
sendCommandToIndClient	פונקציה זהה ל – sendCommand אך בניגוד אליה היא שולחת את הפקודה ללקוח אחד בלבד.
newConnection	פונקציה זו מפעילה תהליכון נפרד עבור כל לקוח שמתחבר לשרת. הפונקציה מקבלת הודעות מהלקוח ושולחת אותן לפונקציית הטיפול בפקודות.
handelCommand	פונקציה לטיפול בפקודות שמתקבלות. ברגע שמתקבלת מלקוח מסוים אחת מהפקודות שהוסברו קודם היא מבצעת את הטיפול הנדרש (יצירת קובץ, שינוי שם, שינוי מיקום קובץ וכו') ושולחת את הפקודה לשאר הלקוחות לביצוע כדי לשמור על הסנכרון.
startServer	פונקציה המפעילה תהליכון נפרד לצורך האזנה ברקע לבקשת חיבור חדש מלקוח נוסף.

המחלקה Command :

מחלקה זו מגדירה אובייקט מסוג Command שמייצג פקודה ובה כל הפרטים הדרושים לביצוע הפקודה ושמירה על הקבצים מסונכרנים.

שם הפונקציה	תיאור הפונקציה
init	בנאי של המחלקה. מגדיר את הפקודה ואת כל הפרטים שלה – נתיב, פורט, נתיב חדש, שם חדש, מזהה ייחודי של הקובץ ומידע נוסף על הקובץ.
printData	הדפסת פרטי הפקודה.

המחלקה myFileTransfer :

מגדירה מחלקות של השרת והלקוח לצורכי שליחת וקבלת קבצים.

בשליחת קבצים מהשרת – עבור כל קובץ ועבור כל לקוח נפתח socket נפרד בפורט אקראי שדרכו הקובץ נשלח. כששליחת הקובץ מסתיימת ה – socket נסגר.

בשליחת קבצים מלקוח – הלקוח מבקש מהשרת שיפתח לו socket לשליחת הקובץ, השרת פותח ושולח ללקוח את הפרטים של ה – socket שנפתח ודרכו שולח את הקובץ.

המחלקה ServerFileReceive :

שם הפונקציה	תיאור הפונקציה
init	בנאי של המחלקה. יוצר socket חדש למשך 60 שניות לצורך חיבור של הלקוח אליו, בוחר פורט אקראי בטווח 30000-40000 ופותח תהליכון חדש עבור התקשורת.
startServer	פונקציה לפתיחת תהליכון עבור האזנה לבקשות חיבור של לקוחות.
newConnection	פונקציה לקבלת קובץ מהלקוח. הקובץ מתקבל בחתיכות (chunks) של 2048Byte.

המחלקה ServerFileSend :

שם הפונקציה	תיאור הפונקציה
init	בנאי של המחלקה. יוצר socket חדש למשך 60 שניות לצורך שליחת הקובץ, בוחר פורט אקראי בטווח 30000-40000 ופותח תהליכון חדש עבור התקשורת.
startServer	פונקציה לפתיחת תהליכון עבור האזנה לבקשות חיבור של לקוחות.
newConnection	פונקציה לשליחת קובץ ללקוח. הקובץ נשלח בחתיכות (chunks) של 2048Byte.

המחלקה ServerDatasetReceive :

שם הפונקציה	תיאור הפונקציה
init	בנאי של המחלקה. מיועד עבור קבלת מסד הנתונים מהלקוח לצורך השוואה מול הקבצים. יוצר socket חדש למשך 60 שניות לצורך קבלת מסד הנתונים, בוחר פורט אקראי בטווח 30000-40000 ופותח תהליכון חדש עבור התקשורת.
startServer	פונקציה לפתיחת תהליכון עבור האזנה לבקשות חיבור של לקוחות.
newConnection	פונקציה לקבלת מסד הנתונים מהלקוח. מתקבל בחתיכות (chunks) של 2048Byte.

המחלקה ClientFileReceive:

שם הפונקציה	תיאור הפונקציה
init	בנאי של המחלקה. מקבל כתובת ip, פורט ונתיב של הקובץ, מתחבר לשרת לפי הפרטים שקיבל ומקבל את הקובץ בחתיכות (chunks) של 2048Byte.

המחלקה ClientFileSend:

שם הפונקציה	תיאור הפונקציה
init	בנאי של המחלקה. מקבל כתובת ip, פורט ונתיב של הקובץ, מתחבר לשרת לפי הפרטים שקיבל ושולח את הקובץ בחתיכות (chunks) של 2048Byte.

המחלקה ClientDatasetSend:

שם הפונקציה	תיאור הפונקציה
init	בנאי של המחלקה. מיועד עבור שליחת מסד הנתונים לשרת. מקבל כתובת ip ופורט, מתחבר לשרת לפי הפרטים שקיבל ושולח את מסד הנתונים בחתיכות (chunks) של 2048Byte.

המחלקה Monitor:

מחלקה שמגדירה אובייקט מסוג MonitorCom שמטרתו יכולת תקשורת בין מחלקות ופונקציות אחרות לבין פונקציית ניטור השינויים. אובייקט זה נשלח גם לשרת וגם ללקוח ובעצם האחראי העיקרי על כל תהליך סנכרון התיקייה.

שם הפונקציה	תיאור הפונקציה
sendAllRequested	פונקציה שמטרתה שליחת כל הקבצים הקיימים בתיקייה המסונכרנת ללקוח חדש שמתחבר לשרת. ראשית נוצרות כל התיקיות והתת תיקיות בתיקייה המסונכרנת ולאחר מכן נשלחים הקבצים.
sendUpdateRequested	פונקציה שמטרתה סנכרון מחדש של התיקייה המסונכרנת לאחר ניתוק התקשורת בין הלקוח לשרת והתחברות מחדש. הפונקציה מחלקת את הקבצים והתיקיות למערכים שונים – עבור שליחה, עבור מחיקה, עבור עריכה, עבור שינוי שם ועבור שינוי מקום. לכל קובץ במערך אלה מתבצעת הפקודה המתאימה לו.
insertToDataSet	פונקציה להכנסת פרטי קובץ חדש למסד הנתונים שלנו.
existsInDataSet	פונקציה זו מחזירה True אם נתיב נתון קיים במסד הנתונים שלנו
updateFileInDataSet	פונקציה זו נקראת כאשר אנו מעדכנים קובץ קיים (כמו שינוי שם קובץ או שינוי מיקום קובץ). הפונקציה משתמשת במנעול על מסד הנתונים כדי שבזמן העריכה תהליכון אחר לא יוכל לבצע פעולות על הקובץ.
deleteFromDataSet	פונקציה זו נקראת כאשר אנו מוחקים קובץ קיים. הפונקציה משתמשת במנעול על מסד הנתונים כדי שבזמן המחיקה תהליכון אחר לא יוכל לבצע פעולות על הקובץ.
saveDataSet	פונקציה לשמירת מסד הנתונים בתוך קובץ.
openDataSet	פונקציה לפתיחת קובץ מסד הנתונים. במקרה שהקובץ לא קיים הפונקציה תחזיר False.
md5checksum	עוברת על כל הבתים של הקובץ ויוצרת Hash עליהם. נשתמש ב – Hash זה כדי לגלות שינויים בקובץ בעבודה עם מחשבים שונים (לא נוכל לבדוק שינויים בדרך אחרת למשל בדיקת זמן כי אין הבטחה שהשעונים יהיו מסונכרנים במחשבים שונים).
addFilesInDirs	פונקציה הנקראת בהפעלה הראשונית של התוכנה. היא סורקת את התיקייה המסונכרנת ומכניסה למסד הנתונים את פרטי כל הקבצים. פונקציה זו יוצרת את מסד הנתונים הבסיסי אותו אנחנו מתחזקים בעזרת הפונקציה scanFolderForChanges.
recallScan	פונקציה זו משמשת לקריאה מחדש לפונקציית סריקת התיקיות בכל 5 שניות. הפונקציה קוראת ראשית ל - scanFolderForChanges ולאחר מכן קוראת ל - scanForDeleted לצורך השלמת סריקת התיקייה המסונכרנת.

שם הפונקציה	תיאור הפונקציה
scanFolderForChanges	פונקציה זו היא החלק הראשון של סריקת התיקייה לצורך איתור שינויים. היא מחפשת הבדלים בין קבצים בתיקייה לבין הנתונים במסד הנתונים.
scanForDeleted	פונקציה זו היא החלק השני של סריקת התיקייה לצורך איתור שינויים. היא מחפשת קבצים שקיימים במסד הנתונים אבל כבר לא נמצאים בתיקייה ומחוקת אותם מהתיקייה. שילוב של שתי הפונקציות האלה נותן לנו סריקה מלאה של שינויים ושמירה על סנכרון.
main	הפונקציה הראשית של התוכנה. בחירת נתיב התיקייה המסונכרנת (ושמירתו בקובץ config), יצירת קובץ מסד נתונים, יצירת אובייקט מסוג MonitorCom ליצירת קשר בין המוניטור לבין השרת/הלקוח, יצירת קובץ connectionSettings המכיל נתוני תקשורת (כתובת IP והאם מדובר בשרת או לקוח), יצירת אובייקט שרת / לקוח בהתאם לקובץ הגדרות התקשורת, בקשת קבצים מהשרת (חיבור ראשוני / התחברות מחדש), ולבסוף הקפצת התראת ברוכים הבאים ואודות.

המחלקה Tray :

מחלקה שאחראית על ה-GUI בפרויקט. היא יוצרת אייקון בסרגל היישומים ומגדירה את הפונקציות שניתן לבצע דרך האייקון בתוכנה.

שם הפונקציה	תיאור הפונקציה
init	בנאי של המחלקה. מקבל קישור ל-monitor ומגדיר את נתיב התיקייה המסונכרנת, קובע את אייקון התוכנה בסרגל היישומים ומצמיד לו תפריט עבור תפעול המערכת.
menu	פונקציה שבונה תפריט לתפעול המערכת דרך האייקון בסרגל היישומים.
openLog	פונקציה ליצירת ופתיחת קובץ Log אליו יגיעו כל הלוגים של המערכת.
quit	פונקציה שדרכה יוצאים מהתוכנה.

המחלקה Util:

מחלקה שאחראית על הכנסת מידע לקובץ ה – Log של התוכנה.

שם הפונקציה	תיאור הפונקציה
printToLog	פונקציה שמקבלת מחרוזת (מידע על תהליך מסוים בתוכנה), יוצרת לו פורמט שכולל תאריך ושעה ומכניסה אותו לקובץ Log.

המחלקה Notifier:

מחלקה שאחראית על יצירת אובייקט התראות מערכת להצגה עבור המשתמש.

הרעיון הוא להציג הודעות חשובות למשתמש כמו עדכון על קבצים שעודכנו, שונו, התקבלו ונשלחו בחלונית קטנה כהתראה.

שם הפונקציה	תיאור הפונקציה
init	בנאי של המחלקה ליצירת האובייקט.
notify	פונקציה זו מציגה התראה על המסך עם תמונת ברירת המחדל שבחרנו, הגדרת כותרת ההודעה, ההודעה עצמה וזמן תצוגת ההודעה.
notifyChangeIcon	פונקציה להחלפת התמונה בחלונית ההתראה.