**Cloud server**

מסמך עיצוב

דניאל וולקוביץ

גרסה 2.0

8.6.2016

**היסטוריית גרסאות המסמך**

|  |  |  |
| --- | --- | --- |
| **תאריך** | גרסה | **תקציר השינויים** |
| 2.3.2016 | 1.0 | הכנת המסמך לקראת התחלת הפרויקט |
| 22.4.2016 | 1.1 | שינויים מינוריים |
| 8.6.2016 | 2.0 | עדכון המסמך לפני הגשה |



**1. הקדמה**

1.1 מטרה

מטרת המסמך היא להציג את המערכת Cloud Server בצורה פונקציונלית, את הדרישות עם ההיבט התפעולי שלה ואת ניתוח הדרישות. מסמך זה משמש כבסיס לכתיבת קוד התוכנה. בנוסף המסמך ישמש לבדיקת המערכת על ידי יצירת תסריטי בדיקה.

1.2 המוצר

**מערכת Cloud Server, מכאן ואילך יקרא הפרויקט בשם: "ענן".**

**ה"ענן" מאפשר לכל משתמש רשום, ז.א. שנרשם בעזרת התוכנה ונמצא במאגרי השרת, לסנכרן את קבציו הנמצאים על מקום אחסון פרטי כגון קונן קשיח, על גבי ה"ענן".**

**המשתמש יכול להסיר דאגה מליבו בידיעה שהמידע שלו נמצא מגובה ב"ענן". אם תקרה למחשב שלו תקלה, הקבצים עדיין יהיו קיימים ונגישים ב"ענן".**

**המשתמש מתקין על המחשב שלו אפליקצית לקוח ודרך ממשק משתמש הוא בוחר תיקייה (או מספר תיקיות) שיסונכרנו אל ה"ענן". כמוכן יוכל המשתמש לשחזר ולהוריד קבצים שהעלה בעבר ל"ענן".**

**המטרה היא לספק למשתמש ממשק פשוט וטבעי שמתחבר עם צורת השמירה הקיימת, יכולת לסנכרן את קבציו ותיקיותיו ל"ענן". תמיד ניתן לאחזר, במקרה הצורך, את הקבצים מה"ענן".**

1.3 מסמכים קודמים

מסמך אפיון

1.4 הגדרות

*ענן: בפרויקט זה משמעות המילה "ענן" הינו שרת המסנכרן קבצים ממספר בלתי מוגבל של לקוחות ולאו דווקא מאפשר שיתוף של הקבצים הנ"ל.*

1. ארכיטקטורת המערכת

2.1 מבט על

המערכת מורכבת מ-2 קומפוננטות עקריות: קומפוננטת שרת וקומפוננטת לקוח.



Cloud\_Server component הינה קומפוננטה המייצגת את התוכנה שרצה בשרת. המיקום הפיזי של השרת אינו משנה, ובכל מקרה התקשורת אליו מתבצעת לפי התרשים למעלה.

המבנה הפנימי של השרת בנוי בהתאם לתרשים מטה:



Cloud\_Client component הינה קומפוננטה המייצגת את התוכנה שרצה בלקוח:



2.2 פירוט רכיבי המערכת

**2.2.1 המבנה הסטטי של התוכנה:**



**הערה**: בסעיפים הבאים יפורטו הפקג'ים/מודולים שכוללים מחלקות.

פקג'ים שאינם כלולים מחלקות כבר פורטו עם הפונקציות שלהם בתרשים הקודם.

2.2.2 Folder\_manager



2.2.3 Folder\_monitor



2.2.4 User\_system



2.2.5 cloud\_server



2.2.6 InitSyncSequence

תרשים זה מפרט את המימוש של use case ההרשמה למערכת ואת המימוש של use case הכניסה לחשבון המשתמש שצוינו במסמך האפיון.



2.2.7 Synchronize from Server to Client

תרשים זה מפרט את ההמשך של מימוש של use case ההרשמה למערכת ואת המימוש של use case הכניסה לחשבון המשתמש. בעצם המשך ה-sequence הקודם.



2.2.8 Watch for changes sequence

תרשים זה הינו המשך ה-sequence הקודם.

תרשים זה מפרט את המימוש של use case המחיקה, היצירה, שינוי השם והעדכון.



2.2.9 יציאה מהמערכת

היות והמערכת תומכת בכל התנתקות פתאומית מן השרת, לא קיימת דרישה מן הלקוח לבצע יציאה מסודרת בהתאם ל-use case יציאה מחשבון משתמש.

2.3 דיון בנושא העיצוב הנבחר

בחרתי לכתוב את הפרויקט בשפת פיתון מכיוון שזו השפה שאני הכי בקיא בה, והיא השפה שלי הכי נוח לכתוב בה מכיוון שהכתיבה בה מהירה ופשוטה ולפעמים אף מגיעה לכדי אנגלית אמתית (למשל: if folder is in list).

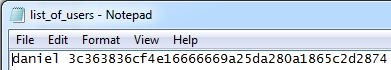
בחרתי לכתוב את הפרויקט ביותר משני מודולים וזאת מכיוון ששאיפתי הייתה להקטין ככל האפשר את התלותיות בין קטעי הקוד. לכן גם הפונקציות הן יחסית קצרות. כמוכן השאיפה הייתה לכתוב מודולים מסודרים ופשוטים. כך למשל: מודול user\_system מטפל בכל הקשור להרשמה, כניסה ויציאה מן המערכת. כמוכן אפשר להבחין כיצד פרוטוקול התקשורת תואם בצורה כמעט מוחלטת חלק מן הפונקציות בשרת ובלקוח.

1. עיצוב נתונים ופרוטוקולים

השרת משתמש בשני מאגרי מידע:

* users\_list
* folders

בעת הרצת השרת במקרה שמאגרי מידע אלו לא קיימים, השרת יצור אותם בעצמו. לאחר מכן, השרת טוען את המאגרים הללו לתוך משתנים. כל שינוי שיתבצע על המשתנה יתבצע על מאגר המידע המתאים לו. השרת זקוק למאגרים אלו כי אם הוא ישמור אותם בתור משתנים הם יימחקו בכל הפעלה מחדש.

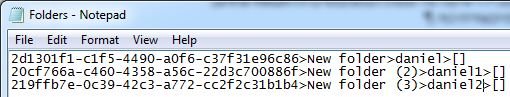
users\_list הוא קובץ שמכיל רשימה של שמות המשתמשים וה-HASH שחושב לסיסמאותיהם, בצמדים ובצורה כזו:

על מנת להבין את קובץ folders יש צורך בהסבר קצר על מבנה שמירת הקבצים בשרת:

לכל לקוח קיימת "תיקיית סנכרון ראשית" (ראה: תיאור הממשק למשתמש; תחת לתיקייה זו מותר לשים רק תיקיות(. זאת מכיוון שהמערכת נבנתה כאופטימלית לאופציית הרחבה לשיתוף קבצים בין משתמשים. לאחר הרחבה זו כל תיקייה שנמצאת מתחת לתיקיית הסנכרון, תהיה בעלת הגדרות ומאפיינים אחרים שעל פיהם יקבע עם אילו משתמשים תשותף כל תיקייה. מתחת לתיקיות הללו ניתן לאחסן כל קובץ\תיקייה כלשהם והם ישותפו עם משתמשים אחרים עפ"י הגדרות התיקייה שנמצאת מתחת לתיקיית הסנכרון.

כל תיקיית סנכרון ראשית לא נשמרת אצל השרת בשם שהלקוח נתן לה, זאת מכיוון שאז ייווצרו התנגשויות בין תיקיות של משתמשים שונים. על כן לכל תיקייה מחושב GUID ומכיוון ש GUID אינו יכול לחזור על עצמו פעמיים אין סיכוי שתיקרה התנגשות בין תיקיות שונות. אך על מנת להחזיר את התיקייה אל הלקוח יש צורך לשמור את שם התיקייה שניתן על ידי הלקוח.

לאחר הבנת עקרונות אלו ניתן להבין את מבנה קובץfolders . הקובץ שומר את: שם המשתמש שלו שייכת התיקייה, רשימה של שמות המשתמשים המשותפים עם התיקייה (כרגע הרשימה תמיד ריקה כי אין אפשרות של שיתוף קבצים בין משתמשים),שם ה GUID של התיקייה כפי שנשמרה בשרת ולבסוף את שם התיקייה האמתי. במקום רווח בין כל אחד מן המשתנים קיים הסימן "<" זאת מכיוון ששימוש ברווח יוצר באגים ומורכבויות מיותרות.

בתמונה הבאה ניתן לראות כיצד נשמרים הנתונים לגבי כל תיקיית סנכרון ראשית:

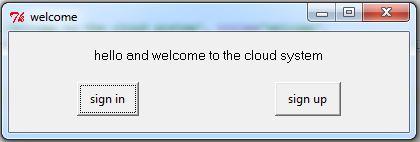
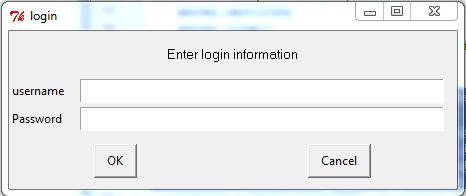
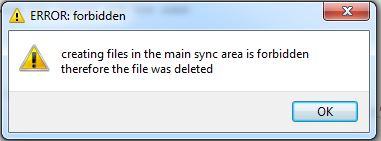
4. ממשק משתמש

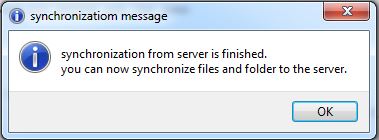
## מידע בסיסי:

1. בתוכנת הענן הלקוחות (המשתמשים) יסנכרנו את הקבצים והתיקיות שלהם אל השרת הייעודי.
2. על מנת לסנכרן כל לקוח חייב לבצע כניסה לחשבון צ'אט שנמצא בשרת. הכניסה תתבצע על ידי שימוש בשם משתמש וסיסמא. משתמש שאין לו חשבון חייב להירשם בפעם ראשונה.
3. השרת מסונכרן עם משתמשים רבים במקביל ובנפרד לחלוטין.
4. השרת תומך בהתנתקות פתאומית של הלקוח.
5. קליטת המסרים מן המשתמש על ידי תוכנית הלקוח (client) מתבצעת בחלון המיועד לכך.
6. המשתמש יגיב להודעות הקופצות בעזרת הקלדה או בעזרת לחיצה על כפתורים שיופיעו על גבי ההודעות הקופצות
7. ללקוח אסור לבצע פעולת DRAG אל תוך תיקיית הסנכרון (אין מגבלה על פעולת העתק הדבק).
8. ללקוח אסור לבצע שינויים בתיקיית הסנכרון כל עוד תוכנת הלקוח לא רצה ברקע.
9. ללקוח אסור לבצע שינויים בתיקיית הסנכרון כל עוד לא קפצה הודעה שהשרת סיים לסנכרן אל המשתמש.
10. ללקוח מותר להניח תיקיות בלבד מתחת לתיקיית synchronization folder (כמובן שבתיקיות שמתחת לתיקייה זו מותר להניח קבצים כרגיל) ובמקרה שינסה לבצע פעולה כזו הקובץ ימחק ותופיע הודעה בהתאם.
11. התווים: / \ @ > < ? \* : !" '

אינם מורשים להימצא בסיסמא או בשם המשתמש.

## דוגמאות להודעות חזותיות בצד הלקוח:

* דוגמא להודעה קופצת בעלת כפתורים מרובים:
* דוגמא להודעה קופצת בעלת שדה קבלת קלט מהמקלדת:
* דוגמא להודעת שגיאה:
* דוגמא להודעה רגילה בעלת כפתור OK בלבד:



* תיקיית הסנכרון שממנה יסונכרנו קבצי המשתמש לשרת:



## הסנכרון

לאחר תהליך הרישום מתבצע הסנכרון. לאחר שהלקוח קיבל הודעה כי השרת סיים לסנכרן אל הלקוח את כל הקבצים והתיקיות, המשתמש יכול לצפות בקבציו בתיקייה synchronization folder הנמצאת בתיקייה שבה רץ קובץ הלקוח. כמוכן הלקוח יכול להניח שם קבצים חדשים והם יסונכרנו אל השרת.

## 

## סיום התוכנית

התוכנית מסתיימת כאשר המשתמש סוגר את התוכנית או כאשר הוא לוחץ על X בהודעות החזותיות לפני ביצוע כניסה לחשבון המשתמש.

## 

## אמצעי הקלט אצל הלקוח

הלחצן השמאלי של העכבר – בחלונות הקופצים העכבר ישמש ללחיצה על כפתורים ( פעמים רבות מקש האנטר יכול להחליף את לחיצת העכבר)

כל שאר המקשים יתפקדו כמו בכל תוכנה רגילה.

5. נספחים

גם הלקוח וגם השרת יוצרים תיקיות זמניות (temp zip files client, temp zip files client). בתיקיות אלו מאוחסנים קבצי ה zip העוברים ברשת ולאחר הרחבתם וגמר השימוש בהם התיקיות הללו נמחקות.