

פרויקט באפליקציות רשת אסינכרוניות: StockWizard

תיכון דרכא רמון על שם אילן ואסף רמון | גדרה

מנחה: אלכסנדר גרשברג

מחבר: יהונתן וישנה;ת"ז: 215310269

תשפ"ג

# תוכן עניינים

תוכן

[תוכן עניינים 1](#_Toc127697751)

[מבוא 2](#_Toc127697752)

[הרקע לפרויקט 2](#_Toc127697753)

[מטרות המערכת 2](#_Toc127697754)

[מטרות על: 2](#_Toc127697755)

[מטרות נלוות: 2](#_Toc127697756)

[תיאור המערכת: 2](#_Toc127697757)

[גבולות המערכת 3](#_Toc127697758)

[עבור תת משתמש 3](#_Toc127697759)

[עבור משתמש עליון: 3](#_Toc127697760)

[עבור מנהל: 3](#_Toc127697761)

[סביבת פיתוח 3](#_Toc127697762)

[שפות תכנות: 3](#_Toc127697763)

[שכבות: 3](#_Toc127697764)

[פלטפורמות של הלקוחות: 4](#_Toc127697765)

[אתגרים מרכזיים: 4](#_Toc127697766)

[חידושים התאמות ועיצוב: 4](#_Toc127697767)

# מבוא

## הרקע לפרויקט

שם הפרויקט: StockWizard

תיאור קצר של הפרויקט: הפרויקט נותן לארגונים גדולים לנהל בקלות מערכות מלאי גדולות ומערכות הוצאה, החזרת ציוד (ניהול השאלות מלאי). יש באפליקציה שני סוגי משתמשים: משתמש תחתון, ומשתמש עליון (מנהל המלאי). משתמש עליון הוא מנהל המלאי של הארגון: הוא יכול להוסיף ולהוריד מלאי, להוציא ציוד (ליצור השאלות), וליצור משתמשים תחתונים השייכים אליו. משתמש תחתון שייך למשתמש עליון מסויים, הוא יכול לראות חלק מרשימת המלאי של המשתמש העליון אליו הוא שייך (רק שמות הפריטים), ולשלול הזמנות ציוד למשתמש העליון. המשתמש העליון מקבל את ההזמנות ששלחו המשתמשים התחתונים, והוא יכול לדחות אותם או לקבל אותם.

קהל היעד: ארגונים גדולים שצריכים לנהל מלאים גדולים והשאלות רבות.

הסיבות לבחירת הנושא: בתור ראש מחסן בשבט הצופים שלי([שחר גדרה](http://צופים.com)), אני מבין את האתגר שבניהול מלאי גדול וכמות גדולה של השאלות ציוד במקביל, לכן רציתי ליצור אפליקציה שתסייע בבעיה זו.

## מטרות המערכת

### מטרות על:

* ניהול יעיל של מלאים גדולים
* ניהול ורישום מהיר וקל של השאלות ציוד
* הזמנה קלה של ציוד ממנהל המלאי

### מטרות נלוות:

* עיצוב קל ומובן- הסברים בתוך האפליקציה, עיצוב עקבי, שימוש בכמה שפחות טקסט וכמה שיותר אייקונים.
* צמצום כמות העמודים באפליקציה- שלא יווצר מבוך של מסכים

## תיאור המערכת:

המערכת מכילה מספר רכיבים: 4 שרותי רשת, אפליקציית צד לקוח, ומסד נתונים המכיל 5 טבלאות מידע.

שירותי רשת:

* שירות שאחראי על רשימות המלאי
* שירות שאחראי על ניהול המשתמשים העליונים
* שירות שאחראי על ניהול השאלות הציוד
* שירות שאחראי על ניהול המשתמשים התחתונים וההזמנות שלהם

טבלאות:

* טבלת הזמנות
* טבלת השאלות
* טבלת משתמשים עליונים
* טבלת משתמשים תחתונים
* טבלת מלאי

עבור משתמש מסוג מנהל מלאי:

בצד הלקוח, כאשר המשתמש מוסיף פריטים לרשימת המלאי שלו, הפריט נשלח לשירות הרשת שאחראי על מלאי, שמוסיף אותו למסד הנתונים. שירות רשת זה מקבל גם פקודות מחיקה, ועדכון של פריטים מצד הלקוח.

מנהל המלאי יכול להוסיף, לערוך, או למחוק מצד הלקוח השאלות של פריטים, בקשות אלו נשלחות לשירות השרת שאחראי על ההשאלות שבתורו, מבצע את שינויים אלו במסד הנתונים.

עבור משתמש מסוג תחתון:

בצד הלקוח, המשתמש יכול לראות את הפריטים שבמלאי של מנהל המלאי שלו ויכול להזמין פריטים. צד הלקוח שולח את משימות אלו אל שירות השרת שאחראי על המשתמשים התחתונים שבתורו, מבצע שינויים אלו במסד הנתונים.

## גבולות המערכת

### עבור תת משתמש

יכול לקבל חלק מהמידע בטבלת המלאי, ולשלוח הזמנות, זמינים לו סך הכול 2 מסכים:

1. שליחת הזמנות מלאי למשתמש העליון
2. צפייה בהזמנות שהוא כבר שלח.

### עבור משתמש עליון:

משתמש עליון יכול להוסיף, לשנות, ולמחוק רשומות ברשימת המלאי שלו, הוא יכול להוסיף משתמשים תחתונים שהשייכים אליו ולשנות את פרטייהם, הוא יכול הוסיף השאלות ידנית או לקבל הזמנות מהמשתמשים התחתונים שלו ולהוסיף אותם להשאלות. סך הכול זמינים לו 7 עמודים:

1- התחברות

2- הרשמה

3- שינוי פרטי

4- הוספת משתמשים

5- ניהול מלאי

6- ניהול השאלות

7- ניהול הזמנות

### עבור מנהל:

למשתמש מנהל (system admin) יש גישה למסך בו הוא יכול לנהל את כל המשתמשים הגבוהים(לשנות פרטים, להוסיף, למחוק).

## סביבת פיתוח

כסביבת פיתוח, השתמשתי בvisual studio 2017 community addition של Microsoft לפיתוח אפליקציית הלקוח ושרותי הרשת. לצורך ניהול מסדי הנתונים (sequential relational databases) השתמשתי בSQL server express 2014 של Microsoft.

## שפות תכנות:

לצורך פיתוח האפליקציה השתמשתי בשפות הבאות:

* בC#: לצורך כתיבת הקוד המנהל אצ האירועים בצד הלקוח ובצד השרת.
* בXAML: לצורך עיצוב עמודי האפליקציה (צד לקוח)- זוהי שפת תגיות לעיצוב ממשקים ויזואלים (דומה לHTML).
* בHTML: השתמשתי בHTML לצורך עיצוב עמוד האודות על האפליקציה.
* בSQL: לצורך ניהול הנתונים והמידע של המשתמשים בשרותי השרת.

## שכבות:

בפרוייקט שלי יש 3 שכבות עיקריות: שכבת לקוח (האפליקציה הויזואלית שאיתה מתממשק המשתמש), שכבת שרת – שם מבוצעות פעולות שינוי המידע לפי בקשות המתקבלות מצד הלקוח בין אם בשירותי רשת פנימיים (ששיכים לאותה חברה שמפתחת את צד הלקוח) או בחיצוניים (שירותי רשת של חברות אחרות), שכבת נתונים- זהו מסד הנתונים שמכיל את כל המידע של המשתמשים ושל המערכת, ומאחסן אותם.

## פלטפורמות של הלקוחות:

הלקוחות משתמשים בפלטפורמת Windows, זאת כיוון שהאפליקציה פותחה עבור פלטפורמת UWP (universal windows platform) שהיא הפלטפורמה שבה הכי קל לכתוב אלפיקציות native למחשבי windows- שהם סוג המחשבים הכי נפוץ.

כלומר- הפלטפורמה UWP נבחרה עקב מספר המשתמשים הרב שיכול להשתמש בפלטפורמה זו.

## אתגרים מרכזיים:

כמו שכבר כתבתי, הצורך בפרויקט כזה בא לענות על צורך בו נתקלתי בתור ראש מחסן בצופים. התחלתי את תהליך הפיתוח עם אפליקציה רזה שנועדה לניהול מלאי בלבד, ולאט לאט הוספתי פיצ'רים נוספים למשתמש העליון- הוספת השאלות, ייבוא המלאי מקובץ csv, חיפוש פריטים חדשים לקנייה בגוגל וכו'...

האתגר המרכזי בו נתקלתי בפיתוח הפרוייקט היה שינוי המערכת כך שהיא תתאים לשני סוגי משתמשים (עליון ותחתון), לא תכננתי את שינוי זה בהתחלה ולקח זמן לארגן מחדש את צד השרת והלקוח כך שיתאים לשינוי זה.

אתגר נוסף בו נתקלתי הוא הצגת טבלאות אינטואיטיבות שהמשתמש יכול להשתמש בהן (כיוון שמטבע הפרוייקט יש צורך בטבלאות ורשימות רבות). היה מידע מוגבל בנושא באינטרנט ולאחר מחקר רב למדתי איך עובדים עם פקד ListView ואיך משתמשים בקשירה דו צדדית של מידע לטבלה (two way data binding) כדי שהמשתמש יוכל לשנות ישירות את המידע בעצמו.

## חידושים התאמות ועיצוב:

לאחר התקדמות רבה בפרויקט, למדתי רבות על שיטות עיצוב ופריסת התוכן באפליקציה בuwp. לאחר העמודים הראשונים, למדתי איך לפרוס נתונים בצורה שתתאים לכל מסך בעזרת עמודות ושורות בפקד Grid, איך ליצור רצף של תוכן אנכי ואופקי בעזרת פקד StackLayout ואיך להוריד את כמות המסכים בפרוייקט בעזרת עיצוב מתקדם של ContentDialogs וכו'...

# ניתוח המערכת

## מצב קיים:

כרגע, במערכת יש שני סוגי משתמשים: עליון ותחתון.

משתמש עליון יכול: להירשם ולהיכנס למשתמש שלו, ליצור ולערוך רשימת מלאי, להוסיף השאלות של פריטים מהמלאי, להוסיף משתמשים תחתונים השייכים אליו, ולקבל ולאשר או לדחות הזמנות המשתמשים התחתונים השייכים אליו.

משתמש תחתון: משתמש תחתון יכול לראות רשימה של הפריטים הזמינים במלאי של המשתמש העליון אליו הוא שייך, לבצע הזמנות, ולראות את מצב ההזמנות ששלח.

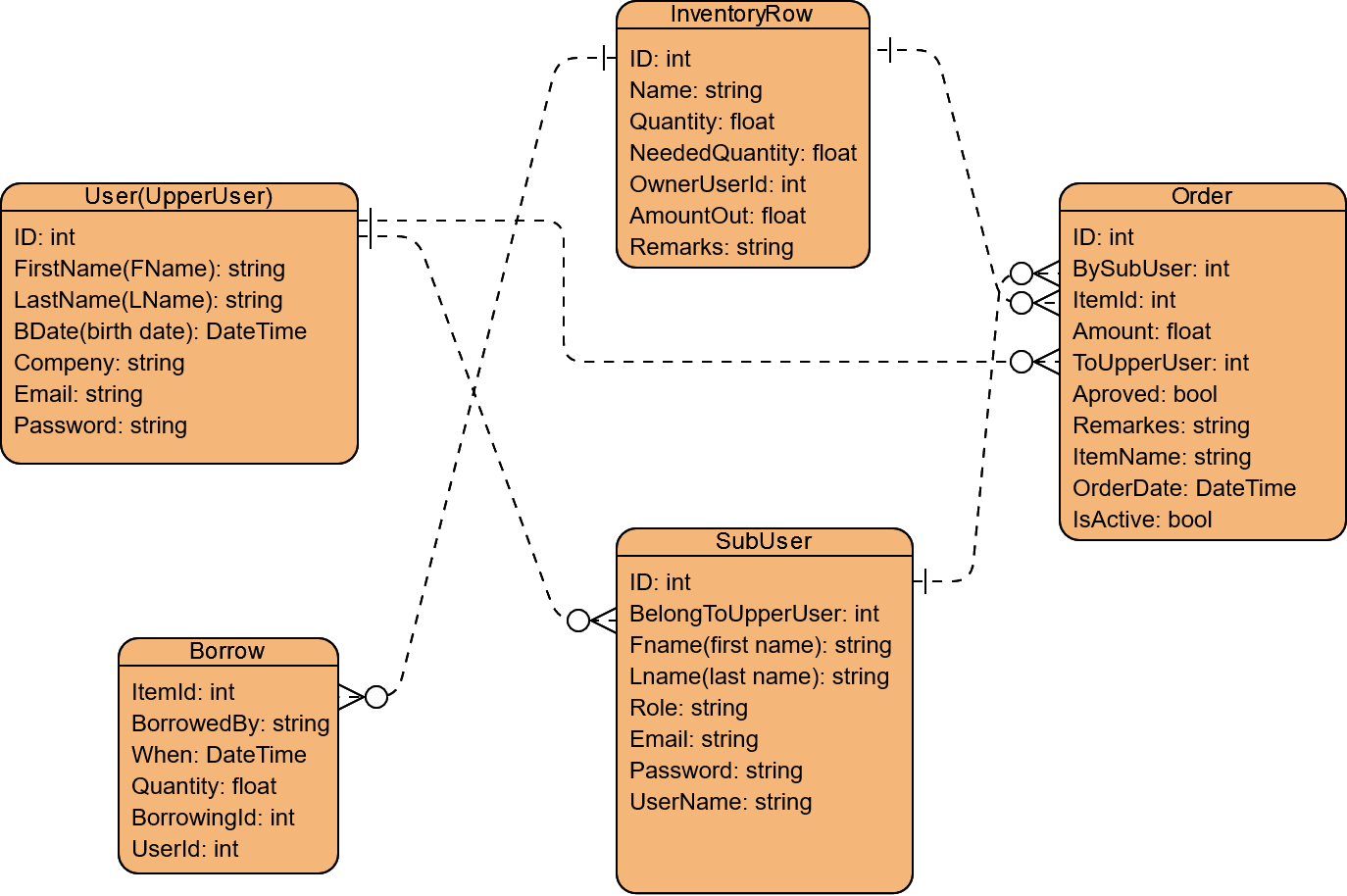
## מטרות מערכת עתידיות

לעתיד, הייתי שם למטרה להוסיף עוד אופציות מעקב של המשתמש התחתון אחרי ההזמנות שלו. כרגע, לאחר שהזמנה מאושרת והופכת להשאלת ציוד, היא רק מופיעה למשתמש כמאושרת ואין לו אופציה לעקוב אחרי השאלות הציוד הפעילות שלו.

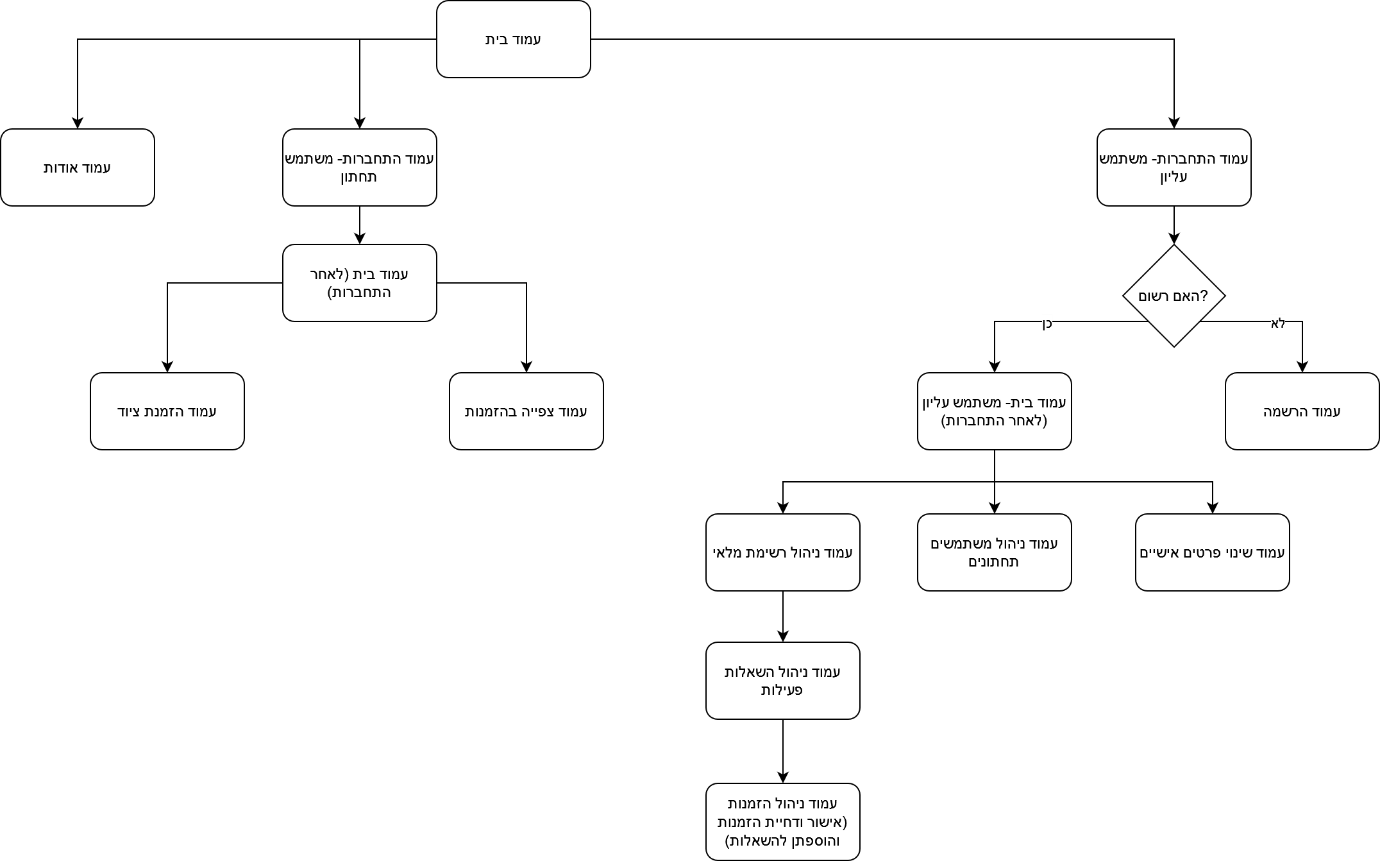
## מצב מערכת עתידית

במערכת עתידית אידיאלית, המשתמש התחתון יוכל לעקוב אחרי ההשאלות הפעילות שלו ומצבן (כאשר היום הוא יכול לעקוב רק אחרי ההזמנות).

## תרשים ERD



## עץ תהליכים



# בסיס הנתונים (database)

## תרשים DSD

