

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](#_Toc127966034)

[מבוא 2](#_Toc127966035)

[הרקע לפרויקט 2](#_Toc127966036)

[מטרות המערכת 2](#_Toc127966037)

[מטרות על: 2](#_Toc127966038)

[מטרות נלוות: 2](#_Toc127966039)

[תיאור המערכת: 2](#_Toc127966040)

[גבולות המערכת 3](#_Toc127966041)

[עבור תת משתמש 3](#_Toc127966042)

[עבור משתמש עליון: 3](#_Toc127966043)

[עבור מנהל: 3](#_Toc127966044)

[סביבת פיתוח 3](#_Toc127966045)

[שפות תכנות: 3](#_Toc127966046)

[שכבות: 3](#_Toc127966047)

[פלטפורמות של הלקוחות: 4](#_Toc127966048)

[אתגרים מרכזיים: 4](#_Toc127966049)

[חידושים התאמות ועיצוב: 4](#_Toc127966050)

[ניתוח המערכת 5](#_Toc127966051)

[מצב קיים: 5](#_Toc127966052)

[מטרות מערכת עתידיות 5](#_Toc127966053)

[מצב מערכת עתידית 5](#_Toc127966054)

[תרשים ERD 5](#_Toc127966055)

[עץ תהליכים 6](#_Toc127966056)

[בסיס הנתונים (database) 7](#_Toc127966057)

[תרשים DSD 7](#_Toc127966058)

[טבלאות ראשיות 7](#_Toc127966059)

[טבלת Users: 7](#_Toc127966060)

[טבלת Inventory 7](#_Toc127966061)

[טבלת BorrowedItems 8](#_Toc127966062)

[טבלת SubUsers 8](#_Toc127966063)

[טבלת Orders 8](#_Toc127966064)

[צד שרת 10](#_Toc127966065)

# 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.

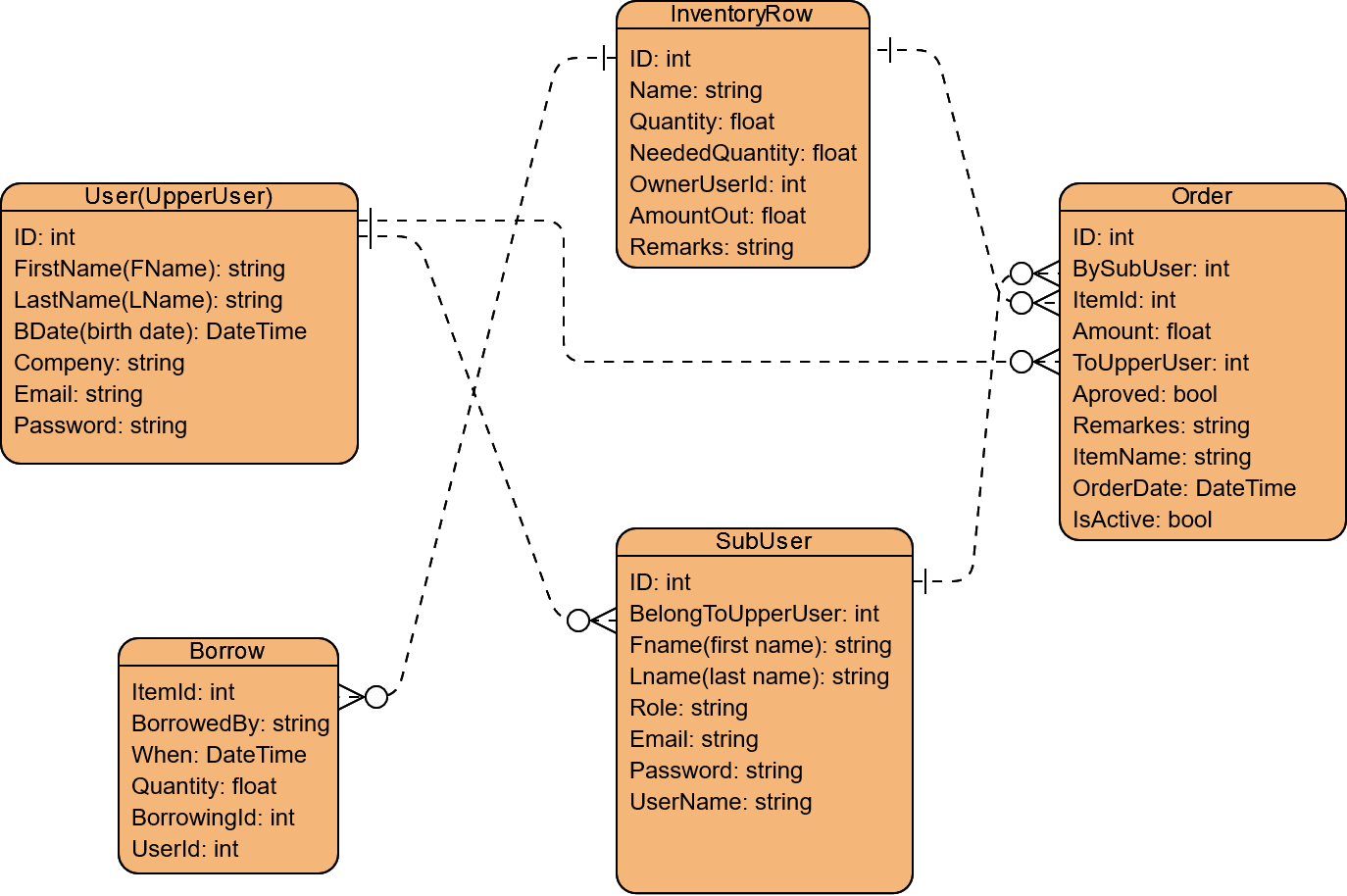
## מטרות מערכת עתידיות

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.

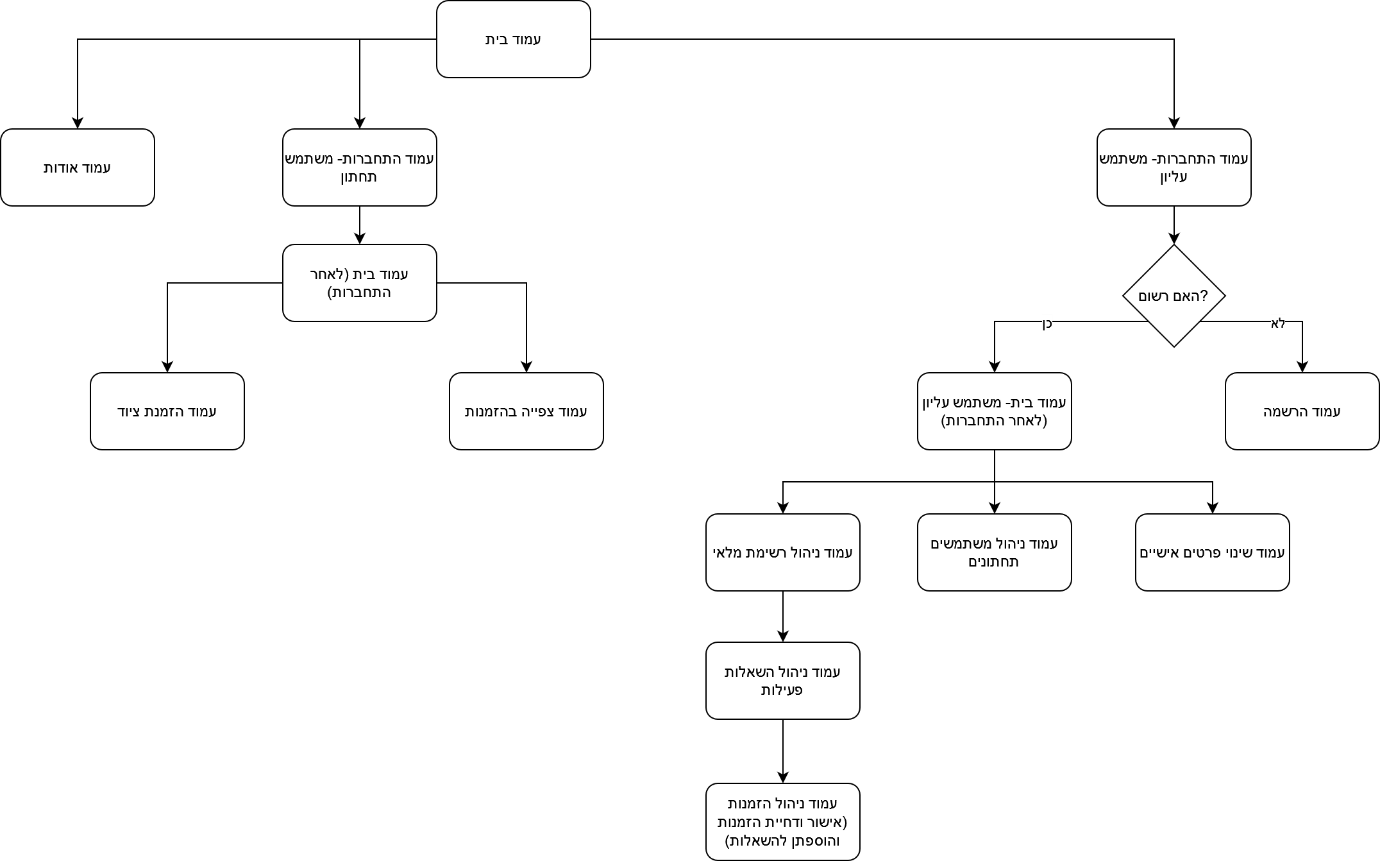
## מצב מערכת עתידית

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

## תרשים ERD

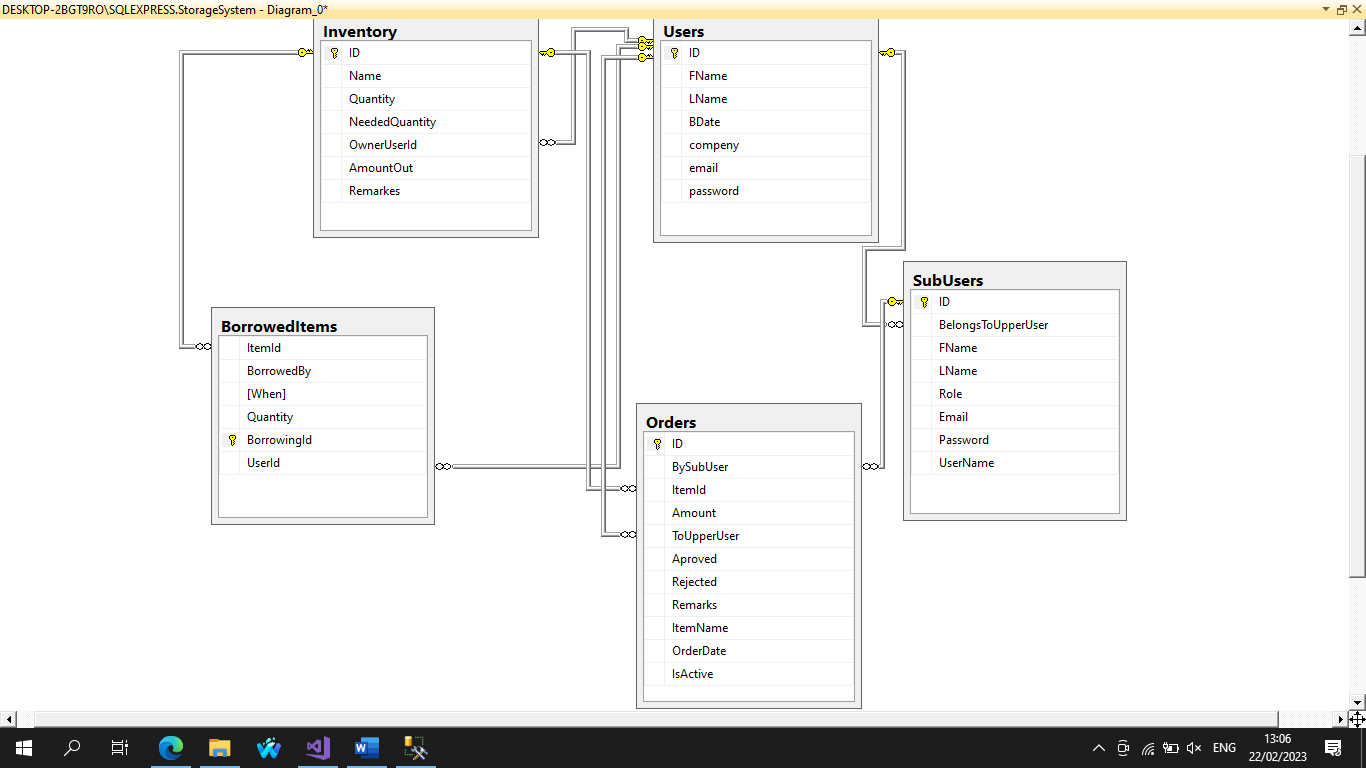


## עץ תהליכים



# בסיס הנתונים (database)

## תרשים DSD



## טבלאות ראשיות

### טבלת Users:

כל רשומה בטבלה מכילה את הפרטים של משתמש עליון במערכת.

הטבלה מכילה מספר שדות:

1. ID- המפתח הראשי והמיוחד של כל משתמש, שדה זה הוא שדה חובה שנקבע אוטומטית לכל משתמש לצורך זיהויו במערכות.
2. fName- שדה טקסט עם השם הפרטי של כל משתמש עליון
3. Lname- שדה טקסט המכיל את השם משפחה של כל משתמש עליון
4. Bdate- שדה מסוג datetime המכיל את תאריך הלידה של המשתמש
5. Compeny- שדה טקסט שמכיל את שם החברה שמייצג המשתמש (במחסן של מי הוא עובד?)
6. Email- שדה טקסט המכיל את כתובת הדוא"ל של המשתמש.
7. Password- שדה טקסט המכיל את קוד הגישה של המשתמש העליון לשירות.

### טבלת Inventory

כל רשומה בטבלה זו מכילה את הפרטים על פריט אחד ברשימת מלאי של אחד ממשתמשי השירות.

הטבלה מכילה את השדות הבאים:

1. ID- שדה ייחודי לכל רשומה, משומש לזיהוי הרשומה והבחתה מאחרות. מיוצר בצורה אוטומטית.
2. Name- שדה טקסט המכיל את השם של הפריט שמיוצג ברשומה.
3. Quantity- שדה מספרי (מסוג עשרוני) שמכיל את הכמות הקיימת במלאי מכל מוצר
4. NeededQuantity- שדה מסוג מספרי (עשרוני) המכיל את הכמות שיש בה צורך במלאי (כמה צריך שתמיד יהיה מפריט זה).
5. OwnerUserId- מייצג את המשתמש שאליו משוייך פריט זה . רשומה זו מקושרת בקשר יחיד אל רבים אל ID בטבלה Users.
6. AmountOut- שדה מסוג מספרי (עשרוני). מייצג את הכמות מפריט זה שכרגע נמצאת מחוץ למחסן (הושאלה החוצה).
7. Remarkes- שדה טקסט המייצג את ההערות של המשתמש על פריט זה (האם יש משהו שהוא רוצה לכתוב על הפריט).

### טבלת BorrowedItems

כל רשומה בטבלה זו מייצגת השאלה של פריט (הוצאתו אל מחוץ למחסן אל אדם אחר).

השדות בטבלה זו:

1. ItemId- שדה זה מייצג את המספר המזהה של הפריט שהושאל. שדה זה מקושר בקשר רבים אל יחיד אל ID בטבלת Inventory.
2. BorrowedBy- זהו שדה טקסט שבו השם של האדם ששאל את הפריט מהמלאי.
3. [When]- זהו שדה datetime שמכיל את התאריך שבו בוצעה ההשאלה.
4. Quantity- זהו שדה מספרי (עשרוני) המכיל את הכמות מאותו הפריט שהושאלה.
5. BorrowingId- זהו שדה מספרי, הוא השדה המזהה של כל רשומה, והוא מייוצר בצורה אוטומטית לכל רשומה.
6. UserId- זהו השדה שאומר אל איזה משתמש עליון שייכת ההשאלה (מהמלאי של מי יצא הפריט). שדה זה מקושר בקשר רבים אל יחיד אל השדה ID בטבלת Users.

### טבלת SubUsers

כל רשומה בטבלה זו מייצגת משתמש תחתון (משתמש שמשוייך למשתמש עליון).

השדות בטבלה הם:

1. ID- שדה זה הוא המזהה הייחודי של כל משתמש תחתון, הוא מיוצר אוטומטית עבור כל משתמש תחתון.
2. BelongsToUpperUser- שדה זה הוא שדה מספרי המייצג את המשתמש העליון אליו משתייך המשתמש התחתון. שדה זה מקושר בקשר רבים אל יחיד אל שדה ID בטבלת Users.
3. FName- זהו שדה טקסט המכיל את השם הפרטי של המשתמש.
4. LName- זהו שדה טקסט המכיל את שם המשפחה של המשתמש.
5. Role- זהו שדה טקסט המכיל את תפקיד המשתמש בארגון.
6. Email- זהו שדה טקסט המכיל את הדוא"ל של המשתמש.
7. Password- זהו שדה טקסט המכיל את קוד הגישה של המשתמש.
8. UserName- זהו שדה טקסט ייחודי לכל משתמש (אין שתי משתמשים עם אותו שם משתמש). הוא מייצג את שם המשתמש של המשתמש.

### טבלת Orders

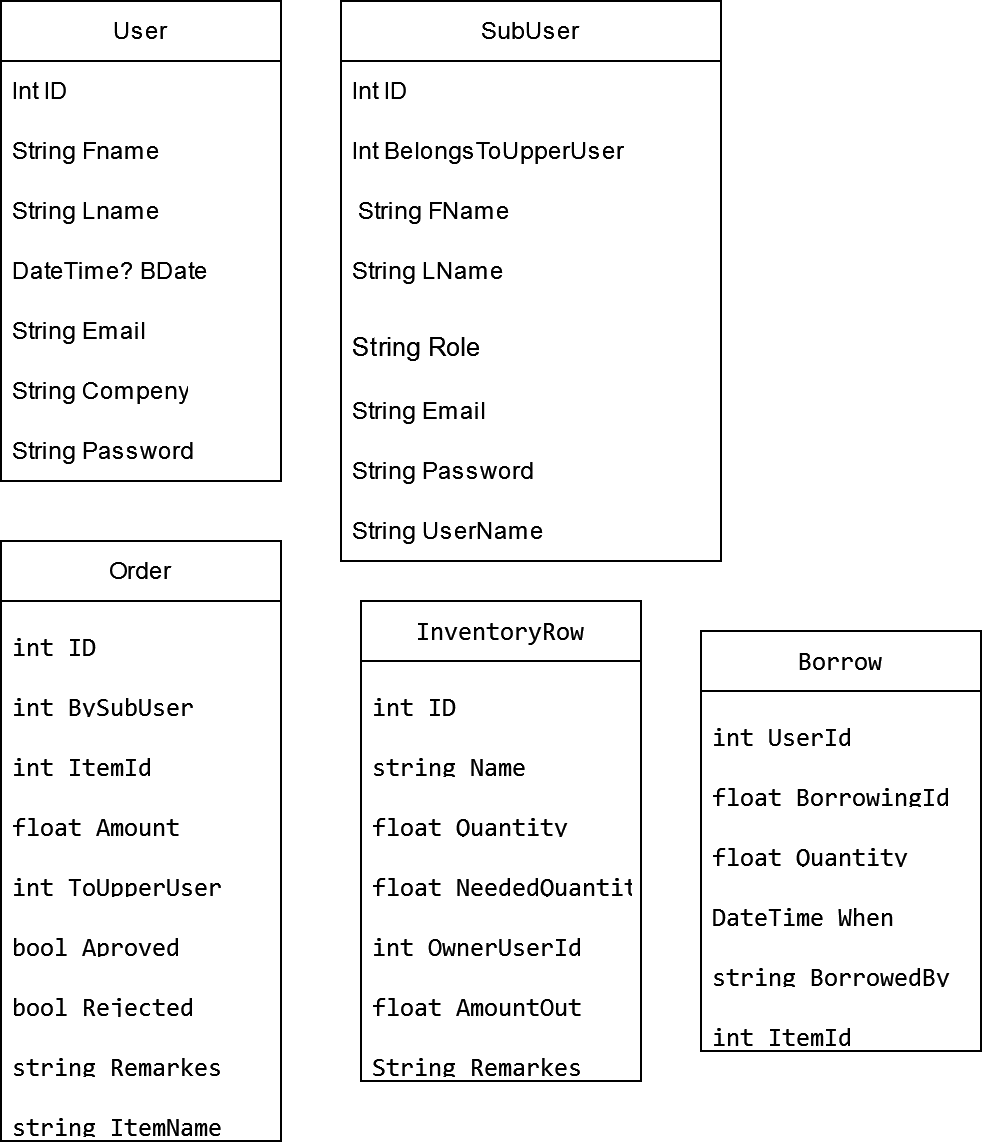
כל רשומה בטבלה זו מייצגת הזמנת ציוד של משתמש תחתון אל משתמש עליון (מחבר בארגון אל מנהל המלאי).

השדות בטבלה הם:

1. ID- שדה מספרי ייחודי לכל רשומה, מיוצר אוטומטית לכל רשומה חדשה.
2. BySubUser- שדה מספרי המייצג ממי הגיע ההזמנה, שדה זה מקושר בקשר רבים אל יחיד אל ID בטבלת SubUsers.
3. ItemId- שדה מספרי זה מייצג את הפריט שהמשתמש התחתון רוצה להזמין. שדה זה מקושר בקשר רבים אל יחיד אל ID בטבלת Inventory.
4. Amount- שדה מספרי(עשרוני), שדה זה מייצג את הכמות מאותו הפריט שהמשתמש רוצה לשאול.
5. Aproved- שדה בוליאני(bit), שדה זה מייצג את התשובה לשאלה: האם מנהל המלאי אישר את ההזמנה?
6. Rejected- שדה בוליאני(bit), שדה זה מייצג את התשובה לשאלה: האם מנהל המלאי דחה את ההזמנה?
7. Remarks- שדה טקסט, שדה זה מכיל את ההערות שיש למשתמש התחתון אל העליון בנוגע להזמנתו.
8. ItemName- שדה טקסט, לצורך פישוט הגישה לשמות הפריטים, טבלה זו מכיל עותק של השם של הפריט שבהזמנה.
9. OrderDate- שדה תאריך (datetime), שדה זה מכיל את התאריך בו הוזמנה ההזמנה.
10. IsActive- שדה בוליאני (bit), עונה על השאלה: האם ההשאלה עוד פעילה ? (האם היא עוד רלוונטית או שניתן להתעלם ממנה).

# צד שרת

## תרשים UML



## קוד של מחלקות נבחרות:

### הקוד של מחלקת User

using System;

namespace WebServ1

{

    public class User

    {

        //תכונות בסיסיות של המשתמש

        //defining basic properties

        private int id;

        private string fname;

        private string lname;

        private DateTime? bdate;

        private string compeny;

        private string email;

        private string password;

        //בנאי ריק

        // empty constractor

        public User()

        {

        }

        //גטרים וסטרים לכל התכונות

        //getters + setters for properties

        public int ID

        {

            get

            {

                return this.id;

            }

            set

            {

                this.id = value;

            }

        }

        public string Fname

        {

            get { return this.fname; }

            set { this.fname = value; }

        }

        public string Lname

        {

            get { return this.lname; }

            set { this.lname = value; }

        }

        public DateTime? BDate

        {

            get { return this.bdate; }

            set { this.bdate = value; }

        }

        public string Compeny

        {

            get { return this.compeny; }

            set { this.compeny = value; }

        }

        public string Email

        {

            get { return this.email; }

            set { this.email = value; }

        }

        public string Password

        {

            get { return this.password; }

            set { this.password = value; }

        }

        //פעולה שיוצרת עצם חדש עם אותם תכונות כמו עצם זה

        // a fution that creatates another User obj identical to this one

        public User copy()

        {

            User a = new User();

            a.ID = this.id;

            a.Fname = this.Fname;

            a.Lname = this.Lname;

            a.BDate = this.BDate;

            a.compeny = this.compeny;

            a.email = this.email;

            a.password = this.password;

            return a;

        }

        //תכונה הבודקת האם משתמש אחר זהה בכל פרטיו למשתמש זה

        // a function that returns wether another User object is identical to this one

        public bool IsSame(User user)

        {

            try

            {

                bool idS = this.ID == user.ID;

                bool FnameS = this.Fname.Equals(user.Fname);

                bool LnameS = this.Lname.Equals(user.Lname);

                bool DateS = this.BDate.Equals(user.BDate);

                bool CompS = this.compeny.Equals(user.Compeny);

                bool EmailS = this.email.Equals(user.Email);

                bool PassS = this.password.Equals(user.Password);

                return idS && FnameS && LnameS && DateS && CompS && EmailS && PassS;

            }

            catch

            {

                return false;

            }

        }

        //made by yehonatan vishna

    }

}

### הקוד למחלקת Order

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace WebServ1

{

    public class Order

    {

        //תכונות של ההזמנה

        private int iD;

        private int bySubUser;

        private int itemId;

        private float amount;

        private int toUpperUser;

        private bool aproved;

        private bool rejected;

        private string remarkes= "";

        private string itemName;

        private DateTime orderDate;

        private bool isActive;

        public Order()

        {

        }

        //גטרים וסטטרים של התכונות

        public int ID { get => iD; set => iD = value; }

        public int BySubUser { get => bySubUser; set => bySubUser = value; }

        public int ItemId { get => itemId; set => itemId = value; }

        public float Amount { get => amount; set => amount = value; }

        public int ToUpperUser { get => toUpperUser; set => toUpperUser = value; }

        public bool Aproved { get => aproved; set => aproved = value; }

        public bool Rejected { get => rejected; set => rejected = value; }

        public string Remarkes { get => remarkes; set => remarkes = value; }

        public string ItemName { get => itemName; set => itemName = value; }

        public DateTime OrderDate { get => orderDate;

            set

            {

                orderDate = value;

            }

            }

        private string status;

        //תכונה שמטרתה לתת סיכום בטקסט למשתמש של מצב הזמנתו

        public string Status

        {

            get { return status; }

            set { status = value; }

        }

        //האם ההזמנה פעילה

        public bool IsActive { get => isActive; set => isActive = value; }

    }

}

### הקוד למחלקת Borrow

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WpfApp1;

namespace InventoryServ

{

    public class Borrow

    {

        private int itemId;

        private string borrowedBy;

        private DateTime when;

        private float quantity;

        private float borrowingId;

        private int userId;

        public Borrow() { }

        // because of the desine and the relation of the tables, a qury to the inventory table must be made in order to get the item's name.

        public string ItemName {

            get

            {

                var con = new Connection(BorowwingsDB.constr);

                con.openCon();

                var ds = con.GetDataSet("item name", "select Name from Inventory where ID = " + this.itemId + ";");

                return ds.Tables["item name"].Rows[0]["Name"].ToString();

            }

        }

        public int ItemId

        {

            get { return itemId; }

            set { itemId = value; }

        }

        public string BorrowedBy { get => borrowedBy; set => borrowedBy = value; }

        public DateTime When { get => when; set => when = value; }

        public float Quantity { get => quantity; set => quantity = value; }

        public float BorrowingId { get => borrowingId; set => borrowingId = value; }

        public int UserId { get => userId; set => userId = value; }

    }

}

### הקוד למחלקת InventoryRow

using System;

using System.Collections.Generic;

using System.Linq;

namespace InventoryServ

{

    public class InventoryRow

    {

        //defines some properies

        private int id;

        private string name;

        private float quantity;

        private float neededQuantity;

        private int ownerUserId;

        private float amountOut;

        private string remarks;

        public InventoryRow() { }

        public InventoryRow(int id, string name, int q, int nq, int oui)

        {

            this.id = id;

            this.name = name;

            this.quantity = q;

            this.neededQuantity = nq;

            this.ownerUserId = oui;

        }

        //getters and setters for all the properties listed above

        public int ID

        {

            get { return this.id; }

            set { this.id = value; }

        }

        public string Name

        {

            get { return this.name; }

            set { this.name = value; }

        }

        public float Quantity

        {

            get { return this.quantity; }

            set { this.quantity = value; }

        }

        public float NeededQuantity

        {

            get { return this.neededQuantity; }

            set { this.neededQuantity = value; }

        }

        public int OwnerUserId

        {

            get { return this.ownerUserId; }

            set { this.ownerUserId = value; }

        }

        public float AmountOut

        {

            get { return this.amountOut; }

            set { this.amountOut = value; }

        }

        public String Remarkes

        {

            get { return this.remarks; }

            set { this.remarks = value; }

        }

        // a fution that creatates another InventoryRow obj identical to this one

        public InventoryRow copy()

        {

            InventoryRow a = new InventoryRow();

            a.ID = this.id;

            a.Quantity = this.quantity;

            a.OwnerUserId = this.ownerUserId;

            a.NeededQuantity = this.neededQuantity;

            a.Name = this.name;

            a.Remarkes = this.remarks;

            return a;

        }

        // a function that returns wether another InventoryRow object is identical to this one

        public bool Equal(InventoryRow row)

        {

            bool IdSame = this.id == row.ID;

            bool QSame = this.quantity == row.Quantity;

            bool OUIdSame = this.ownerUserId == row.OwnerUserId;

            bool NQSame = this.neededQuantity == row.NeededQuantity;

            bool NameSame;

            if ((row.Name == null && this.name != null) || (row.Name != null && this.name == null))

            {

                NameSame = false;

            }

            else

            {

                if ((row.Name == null && this.name == null))

                {

                    NameSame = true;

                }

                else

                {

                    NameSame = this.name.Equals(row.Name.ToString());

                }

            }

            bool RSame;

            if ((row.Remarkes == null && this.remarks != null) || (row.Remarkes != null && this.remarks == null))

            {

                RSame = false;

            }

            else

            {

                if ((row.Remarkes == null && this.remarks == null))

                {

                    RSame = true;

                }

                else

                {

                    RSame = this.remarks.Equals(row.Remarkes.ToString());

                }

            }

            return IdSame && QSame && OUIdSame && NQSame && NameSame && RSame;

        }

    }

}

### הקוד למחלקת SubUser

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace WebServ1

{

    public class SubUser

    {

        //מגדיר תכונות בסיסיות

        //defining reqired properties

        private int id;

        private int belongsToUpperUser;

        private string fName;

        private string lName;

        private string role;

        private string email;

        private string password;

        private string userName;

        public SubUser()

        {

        }

        public int Id { get => id; set => id = value; }

        public int BelongsToUpperUser { get => belongsToUpperUser; set => belongsToUpperUser = value; }

        public string FName { get => fName; set => fName = value; }

        public string LName { get => lName; set => lName = value; }

        public string Role { get => role; set => role = value; }

        public string Email { get => email; set => email = value; }

        public string Password { get => password; set => password = value; }

        public string UserName { get => userName; set => userName = value; }

        public SubUser Copy()

        {

            return new SubUser() { Id = this.id, BelongsToUpperUser = this.BelongsToUpperUser, FName = this.FName, LName = this.LName, Role = this.Role, UserName = this.userName };

        }

        public bool IsSame(SubUser user)

        {

            try

            {

                bool idS = this.Id == user.Id;

                bool FnameS = this.FName.Equals(user.FName);

                bool LnameS = this.LName.Equals(user.LName);

                bool EmailS = this.email.Equals(user.Email);

                bool PassS = this.password.Equals(user.Password);

                bool belS = this.BelongsToUpperUser.Equals(user.BelongsToUpperUser);

                bool roleS = this.Role.Equals(user.Role);

                bool Usr = this.UserName.Equals(user.UserName);

                return idS && FnameS && LnameS && EmailS && PassS && belS && roleS && Usr;

            }

            catch

            {

                return false;

            }

        }

    }

}

## המחלקות המטפלות בנתונים

### תרשים UML

למחלקת Connection שלי אין תכונות (יש רק פעולות), ולכן אי אפשר ליצור לה תרשים UML.