בס"ד

**יומן לבעלת עסק-ספרית**

**המגישה: תמר הולצר**

**ת. ז.: 325984011**

**סמינר דרכי חנה אלעד**

**המנחה: המורה אסתר המניק**

**חלופה: אפליקציות אסינכרוניות**

**שנה: חורף תשפ"ג**

**תוכן עניינים**

[מבוא 2](#_Toc105890107)

[בסיס נתונים 4](#_Toc105890108)

[קשרי גומלין 4](#_Toc105890109)

[תיאור טבלאות 4](#_Toc105890110)

[צד שרת 7](#_Toc105890111)

[שכבת הModel 7](#_Toc105890112)

[שכבת הViewModel 10](#_Toc105890113)

[10](#_Toc105890114)

[שכבת שרותי רשת 18](#_Toc105890115)

[צד לקוח 24](#_Toc105890116)

[ממשק מנהל 38](#_Toc105890117)

[רפלקציה 43](#_Toc105890118)

[.](#_Toc105890119)

# **מבוא**

הרקע לפרויקט

* יומן לבעלת עסק – ספרית.
* תיאור פרויקט- פרויקט זה הינו ניהול יומן לספרית בו היא מעדכנת הזמנות לפגישות, פרטי הזמנה, פרטי לקוח ועוד, כאשר הפרויקט בנוי על לוח שנה שבו מסוכמים כל הפגישות.
* הפרויקט זה מיועד לספריות המעוניינות בניהול מסודר ונגיש לעסק שלהן.
* בחרתי בנושא זה מכמה סיבות, אני מתחברת מאוד לאפליקציות המקלות עלינו בחיי היום- יום בפעולות פשוטות ושיטתיות שחוסכות מאתנו שימת לב לפרטים זוטרים. בעל עסק פרטי תפקידו לנהל את העסק שלו ביעילות ובצורה מסודרת ועקבית מאוד מפני שאין מעליו גורם אחראי, כל האחראיות מוטלת עליו. רצונו לעשות זאת בקלות. אני הענקתי לו בדיוק את היכולת שהוא מבקש, לעשות זאת במהירות ובפשטות ובכלל אני מתחברת מאד לנושא הספרות והאסתטיקה.

מטרות המערכת

מטרת המערכת היא לנהל את יומנה של בעלת העסק ספרית, לקבוע פגישות למלא פרטים לשמור ולהציג זאת בלוח השנה.

בפרויקט זה הספרית פותחת את יומנה על ידי קוד קבוע ונכנסת ללוח השנה שלה בו היא יכולה לדפדף ולבדוק את מספר הפגישות בכל יום ללחוץ על יום ובמידה שיש פגישות הן יוצגו ובמידה שלא ישנה אפשרות לפתוח דף למילוי פרטי הזמנה חדשה.

גבולות המערכת של הלקוח

האפליקציה מאפשרת לספרית לקבוע פגישות ושומרת לה אותן באופן שתוכל לפתוח ביום שנבחר ולהציג את הפגישות של אותו היום, וכמובן לקבוע פגישות חדשות למחוק ולעדכן פגישות קימות. האפליקציה איננה מאפשרת חישוב עצמאי לשעות פנויות לפגישה חדשה, כלומר הצגת השעות הפנויות באותו יום לפגישה חדשה(במידה ויש פגישות באותו היום.) אלא על הספרית לחשב את שעת הפגישה על פי הפגישות הקימות ומשך הזמן שלהן (שזה מוצג בצורה מסודרת ברשימת הפגישות). בחרתי לעשות זאת באופן זה על מנת להעניק לספרית תחושת שליטה מלאה כאלו מדובר ביומנה האישי והפשוט שלה.

סביבת הפיתוח

Visual Studio 2019

מסד נתונים Access 2019

שפות תכנות

צד לקוח : C#, טכנולוגית Windows Forms

צד שרת : C# טכנולוגית WCF

אתגרים מרכזיים

האתגר המרכזי איתו התמודדתי בפרויקט היה קישור רשימת התסרוקות לשאר פרטי ההזמנה והוא היווה עבורי אתגר לוגי משמעותי.

וכן בעיות איתן התמודדתי במהלך פיתוח הפרויקט- בעיות טכניות לרוב שעם הזמן הצלחתי לפתור בכוחות עצמי.

בנוסף לכך, העובדה שהצטרפתי לקבוצת התכנות כ"מוחרגת" (יחד עם בת נוספת אחת בלבד) – שמשלימות בכוחות עצמן את החומר הלימודי בנוסף למטלות הלימודיות הרבות בלימודי התכנות (הכוללים את תוכנית הלימוד של אולטרה-קוד) היוותה אתגר מורכב מאד עבורי.

על איזה צורך הפרויקט עונה-

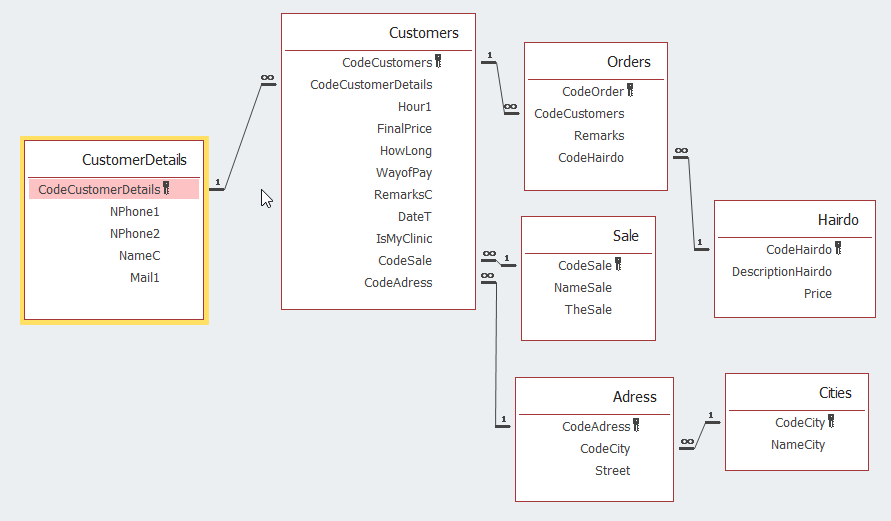
צורך הספריות לניהול העסק שלהן, שזה כולל את כל נושא ההזמנות באופן מסודר ונגיש.

# **בסיס נתונים**

## 

# **קשרי גומלין**

צילום קשרי גומלין מאקסס:



# **תיאור טבלאות**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם הטבלה: Orders תסרוקת בהזמנה. | | | | |
|  | שם אנגלי | שם עברי | סוג | בדיקות תקינות |
| מפתח ראשי | CodeOrder | קוד הזמנה | מספר |  |
|  | CodeCustomers | קוד לקוח | מספר |  |
| Remarks | הערות | טקסט |  |
| CodeHairdo | קוד תסרוקת | מספר |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם הטבלה: Customers פרטי לקוח והזמנה. | | | | |
|  | שם אנגלי | שם עברי | סוג | בדיקות תקינות |
| מפתח ראשי | CodeCustomers | קוד לקוח | מספר |  |
|  | CodeCustomerDetails | קוד פרטי לקוח | מספר |  |
| Hour1 | שעה | טקסט קצר | טקסט בפורמט של שעה |
| FinalPrice | מחיר סופי | מספר |  |
| HowLong | משך זמן | מספר |  |
| WayofPay | אופן תשלום | טקסט קצר | טקסט-מזומן/צקים |
| RemarksC | הערות | טקסט קצר |  |
| DateT | תאריך | טקסט קצר | טקסט בפורמט של תאריך |
| IsMyClinic | האם הטיפול בקליניקה | כן/לא |  |
| CodeSale | קוד הנחה | מספר |  |
| CodeAdress | קוד כתובת הטיפול |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם הטבלה: CustomerDetailsפרטי התקשרות לקוח | | | | |
|  | שם אנגלי | שם עברי | סוג | בדיקות תקינות |
| מפתח ראשי | CodeCustomerDetails | קוד פרטי לקוח | מספר |  |
|  | NPhone1 | מספר פלאפון 1 | טקסט קצר | מספר |
| NPhone2 | מספר פלאפון 2 | טקסט קצר | מספר |
| NameC | שם לקוח | טקסט קצר |  |
| Mail1 | כתובת מייל | טקסט קצר | צריך להכיל אות באנגלית מספר ו-@gmail |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם הטבלה: Adress כתובת | | | | |
|  | שם אנגלי | שם עברי | סוג | בדיקות תקינות |
| מפתח ראשי | CodeAdress | קוד כתובת | מספר |  |
|  | CodeCity | קוד עיר | מספר |  |
| Street | רחוב | טקסט קצר |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם הטבלה: Hairdo סוגי תסרוקות | | | | |
|  | שם אנגלי | שם עברי | סוג | בדיקות תקינות |
| מפתח ראשי | CodeHairdo | קוד תסרוקת | מספר |  |
|  | DescriptionHairdo | תיאור תסרוקת | טקסט קצר |  |
| Price | מחיר | מספר |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם הטבלה: Cities רשימת ערים | | | | |
|  | שם אנגלי | שם עברי | סוג | בדיקות תקינות |
| מפתח ראשי | CodeCity | קוד עיר | מספר |  |
|  | NameCity | שם העיר | טקסט קצר |  |

# **צד** **שרת**

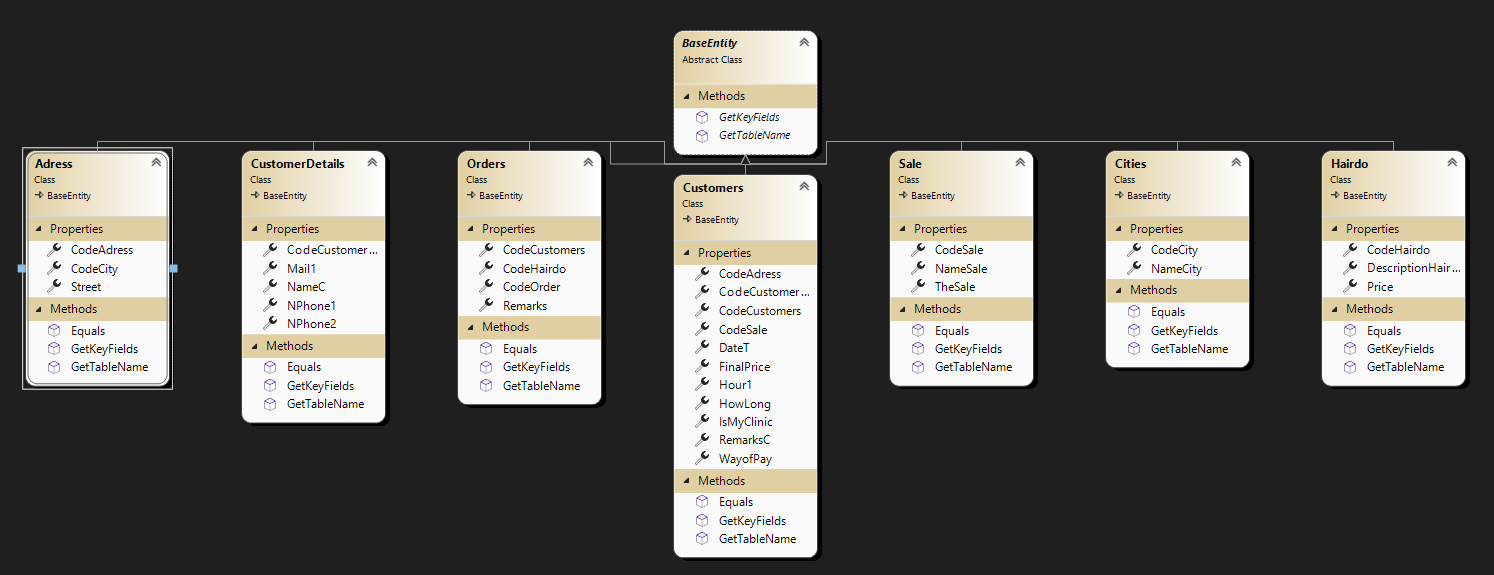
השרת מורכב מ – 4 פרויקטים:

## **שכבת הModel**

שכבה זו מכילה את פירוט העצמים הבסיסיים.

תרשים המחלקות

UML (Class Diagram מVisual Studio)



**מחלקת BaseEntity**

מחלקה המייצגת פריט בודד (שורה) מהאקסס

היא מחלקת הבסיס לכל מחלקות הmodel.

public abstract class BaseEntity

{

public abstract string GetTableName();

public abstract string[] GetKeyFields();

}

**מחלקת פרטי הזמנת הלקוחה**

מחלקה המייצגת את פרטי הלקוחה מיקום הטיפול, שעת הטיפול ואורכו.

public class Customers:BaseEntity

{

public int CodeCustomers { get; set; }

public CustomerD CustomerD { get; set; }

public DateTime DateT { get; set; }

public string Hour { get; set; }

public int FinalPrice { get; set; }

public int HowLong { get; set; }

public string WayofPay { get; set; }

public string RemarksC { get; set; }

public bool IsMyClinic { get; set; }

public Hairdo CodeHairdo { get; set; }

public override string[] GetKeyFields()

{

return new string[] { "CodeCustomers" };

}

public override string GetTableName()

{

return "Customers";

}

**מחלקת הזמנות**

מחלקה המקשרת בין פרטי הזמנת הלקוחה לבין ההערות עליה ופרטי התסרוקות בהזמנה- ע"י הקוד של ההזמנה.

(כל שורה מייצגת סוג אחד של תסרוקת מתוך ההזמנה).

public class Orders : BaseEntity

{

public int CodeOrders { get; set; }

public Customers CodeCustomers { get; set; }

public string Remarks { get; set; }

public override string[] GetKeyFields()

{

return new string[] { "CodeOrders" };

}

public override string GetTableName()

{

return "Orders";

}

}

**מחלקת פרטי לקוחה**

מחלקה המייצגת את פרטי הלקוחה- שמה ,הדוא"ל שלה וטלפונים שלה.

public class CustomerDetails : BaseEntity

{

public int CodeCustomerDetails { get; set; }

public string NPhone1 { get; set; }

public string NPhone2 { get; set; }

public string NameC { get; set; }

public string Mail1 { get; set; }

public override string[] GetKeyFields()

{

return new string[] { "CodeCustomerDetails" };

}

public override string GetTableName()

{

return "CustomerDetails";

}

public override bool Equals(object obj)

{

if (obj is CustomerDetails)

return (obj as CustomerDetails).CodeCustomerDetails == this.CodeCustomerDetails;

return false;

}

**מחלקת תסרוקת**

מחלקה המייצגת את התסרוקות האפשריות- קוד תסרוקת, תיאור התסרוקת ומחירה

public class Hairdo : BaseEntity

{

public int CodeHairdo { get; set; }

public string DescriptionHairdo { get; set; }

public int Price { get; set; }

public override string[] GetKeyFields()

{

return new string[] { "CodeHairdo" };

}

public override string GetTableName()

{

return "Hairdo";

}

public override bool Equals(object obj)

{

if (obj is Hairdo)

return (obj as Hairdo).CodeHairdo == this.CodeHairdo;

return false;

}

}

**מחלקת כתובת**

מחלקה המייצגת את מקום הטיפול- קוד העיר כתובת מדויקת

public class Adress : BaseEntity

{

public int CodeAdress { get; set; }

public string Street { get; set; }

public Cities CodeCity { get; set; }

public override string[] GetKeyFields()

{

return new string[] { "CodeAdress" };

}

public override string GetTableName()

{

return "Adress";

}

public override bool Equals(object obj)

{

if (obj is Adress)

return (obj as Adress).CodeAdress == this.CodeAdress;

return false;

}

}

**מחלקת עיר**

מחלקה המייצגת ערים- קוד עיר, שם העיר.

public class Cities : BaseEntity

{

public int CodeCity { get; set; }

public string NameCity { get; set; }

public override string[] GetKeyFields()

{

return new string[] { "CodeCity" };

}

public override string GetTableName()

{

return "Cities";

}

public override bool Equals(object obj)

{

if (obj is Cities)

return (obj as Cities).CodeCity == this.CodeCity;

return false;

}

}

**מחלקת הנחה**

מחלקה המייצגת את ההנחה במחיר להזמנה -לפי כמות תסרוקות- קוד הנחה, תיאור גורם ההנחה ואת אחוז ההנחה

public class Sale : BaseEntity

{

public int CodeSale { get; set; }

public string NameSale { get; set; }

public int TheSale { get; set; }

public override string[] GetKeyFields()

{

return new string[] { "CodeSale" };

}

public override string GetTableName()

{

return "Sale";

}

public override bool Equals(object obj)

{

if (obj is Sale)

return (obj as Sale).CodeSale == this.CodeSale;

return false;

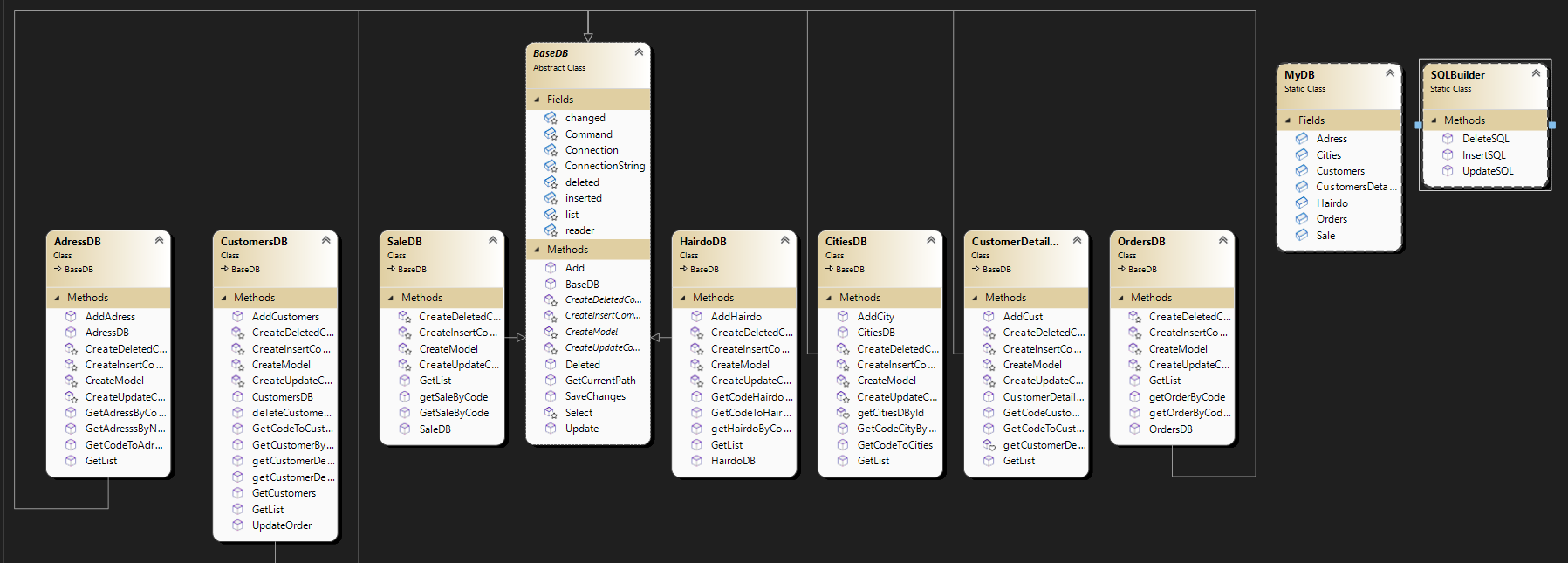
}

}

## **שכבת הViewModel**

שכבה זו מכילה את פונקציונאליות המערכת, והגישה למסד הנתונים.

תרשים המחלקות UML



* מחלקת SQLBuilder:

מחלקה גנרית המשמשת לבנית שאילתות SQL על פי שם וסוג המאפיינים במחלקה.

קוד המחלקה:

public static string InsertSQL(BaseEntity entity)

{

Type type = entity.GetType();

string command = "Insert Into " + entity.GetTableName() + " (";

string values = " Values (";

foreach (var item in type.GetProperties())

{

string n = item.Name;

var value = item.GetValue(entity);

if (value is BaseEntity)

{

string k = ((BaseEntity)value).GetKeyFields()[0];

value = value.GetType().GetProperty(k).GetValue(value);

}

if (value is string || value is DateTime)

{

command += n + " ,";

values += "'" + value + "', ";

}

if (value is int || value is double || value is bool)

{

command += n + " ,";

values += value + " , ";

}

}

command = command.Substring(0, command.Length - 2) + ")";

values = values.Substring(0, values.Length - 2) + ")";

return command + values;

}

public static string UpdateSQL(BaseEntity entity)

{

Type type = entity.GetType();

string command = "Update " + entity.GetTableName() + " set ";

foreach (var item in type.GetProperties())

{

string n = item.Name;

var value = item.GetValue(entity);

if (value is BaseEntity)

{

string k = ((BaseEntity)value).GetKeyFields()[0];

value = value.GetType().GetProperty(k).GetValue(value);

}

if (value is string || value is DateTime)

command += n + " = '" + value + "', ";

if (value is int || value is double || value is bool)

command += n + " = " + value + ", ";

}

string where = "";

foreach (var item in entity.GetKeyFields())

{

if (where != "")

where += " And ";

var value = entity.GetType().GetProperty(item).GetValue(entity);

if (value is BaseEntity)

{

string k = ((BaseEntity)value).GetKeyFields()[0];

value = value.GetType().GetProperty(k).GetValue(value);

}

if (value is string || value is DateTime)

where += item + " = '" + value + "'";

else

where += item + " = " + value;

}

command = command.Substring(0, command.Length - 2) + " Where " + where;

return command;

}

public static string DeleteSQL(BaseEntity entity)

{

Type type = entity.GetType();

string command = "Delete From " + entity.GetTableName() + " Where ";

string where = "";

foreach (var item in entity.GetKeyFields())

{

if (where != "")

where += " And ";

var value = entity.GetType().GetProperty(item).GetValue(entity);

if (value is BaseEntity)

{

string k = ((BaseEntity)value).GetKeyFields()[0];

value = value.GetType().GetProperty(k).GetValue(value);

}

if (value is string || value is DateTime)

where += item + " = '" + value + "'";

else

where += item + " = " + value;

}

command += where;

return command;

}

## }

* מחלקת פרטי לקוח

public CustomerDetailsDB() : base("CustomerDetails")

{

}

///שליפת פרטי כל הלקוחות

public List<CustomerDetails> GetList()

{

if (list.Count() == 0) { Select(); }

var r = list.Cast<CustomerDetails>().ToList();

return r;

}

protected override string CreateDeletedCommand(BaseEntity item)

{

throw new NotImplementedException();

}

protected override string CreateInsertCommand(BaseEntity item)

{

throw new NotImplementedException();

}

protected override BaseEntity CreateModel()

{

CustomerDetails cst = new CustomerDetails();

cst.CodeCustomerDetails = Convert.ToInt32(reader["CodeCustomerDetails"]);

cst.NPhone1 = reader["NPhone1"].ToString();

cst.NPhone2 = reader["NPhone2"].ToString();

cst.NameC = reader["NameC"].ToString();

cst.Mail1 = reader["Mail1"].ToString();

return cst;

}

protected override string CreateUpdateCommand(BaseEntity item)

{

throw new NotImplementedException();

}

//שליפת לקוח על-פי קוד

internal CustomerDetails getCustomerDetailsById(int v)

{

return GetList().FirstOrDefault(item => item.CodeCustomerDetails == v);

}

//שליפת לקוחה לפי שם

public CustomerDetails GetCodeCustomerDetailsByName(string name)

{

return GetList().FirstOrDefault(item => item.NameC == name);

}

//הוספת לקוחה חדשה לאקסס

public int AddCust(CustomerDetails c)

{

Add(c);

var res=SaveChanges();

return res;

}

//שולף את הקוד האחרון ומחזיר אחד יותר - הוספת אוביקט חדש

public int GetCodeToCustomersD()

{

if (GetList().Count == 0) return 1;

return GetList().Max(X => X.CodeCustomerDetails) + 1;

}

## }

* מחלקת הזמנות (customersDB)

public CustomersDB() : base("Customers")

{

}

//שליפת כל ההזמנות

public List<Customers> GetList()

{

if (list.Count() == 0) { Select(); }

return list.Cast<Customers>().ToList();

}

protected override BaseEntity CreateModel()

{

Customers customers = new Customers();

customers.CodeCustomers = (int)reader["CodeCustomers"];

customers.CodeCustomerDetails = MyDB.CustomersDetails.getCustomerDetailsById((int)reader["CodeCustomerDetails"]);

customers.HowLong = reader["HowLong"].ToString();

customers.WayofPay = reader["WayofPay"].ToString();

customers.RemarksC = reader["RemarksC"].ToString();

customers.DateT = Convert.ToDateTime(reader["DateT"]);

customers.Hour1 = reader["Hour1"].ToString();

customers.IsMyClinic = Convert.ToBoolean(reader["IsMyClinic"]);

customers.CodeSale = MyDB.Sale.getSaleByCode((int)reader["CodeSale"]);

customers.CodeAdress = MyDB.Adress.GetAdressByCode((int)reader["CodeSale"]);

customers.FinalPrice = (int)reader["FinalPrice"];

return customers;

}

//שליפת הזמנה לפי תאריך

public List<Customers> GetCustomerByDate(DateTime date)

{

List<Customers> lstDate = new List<Customers>();

foreach (var item in GetList())

{

if (item.DateT == date)

{

lstDate.Add(item);

}

}

return lstDate;

}

public Customers getCustomerDetailsByCode(int code)

{

return GetList().FirstOrDefault(item => item.CodeCustomers == code);

}

public Customers getCustomerDetailsByName(string name)

{

return GetList().FirstOrDefault(item => item.CodeCustomerDetails.NameC == name);

}

protected override string CreateInsertCommand(BaseEntity item)

{

throw new NotImplementedException();

}

protected override string CreateUpdateCommand(BaseEntity item)

{

throw new NotImplementedException();

}

protected override string CreateDeletedCommand(BaseEntity item)

{

throw new NotImplementedException();

}

//מציאת קוד הזמנה חדש- שאינו קיים ברשימה

public int GetCodeToCustomers()

{

if (MyDB.Customers.GetList().Count == 0) return 1;

return MyDB.Customers.GetList().Max(X => X.CodeCustomers) + 1;

}

public Customers GetCustomers(int code)

{

return MyDB.Customers.getCustomerDetailsByCode(code);

}

//הוספת הזמנה

public int AddCustomers(Customers customers,int codeSale1)

{

Add(customers);

int r = SaveChanges();

return r;

}

//מחיקה של הזמנה עם כל הפרטים

public int deleteCustomersWithAllDetails(int code)

{

//חיפוש הלקוחה הזו

//מחיקה של כל השורות בטבלת הזמנות שמקושרות אליה,

//אם יש כתובת

//ורק אז למחוק את הלקוחה

var cust = GetList().FirstOrDefault(u => u.CodeCustomers == 9);

var details = MyDB.Orders.GetList().Where(o => o.CodeCustomers == cust).ToList();

//רק אם זה לא בקליניקה

Adress address = MyDB.Adress.GetList().FirstOrDefault(predicate: u => cust.CodeAdress == u);

MyDB.Adress.Deleted(address);

foreach (var item in details)

{

MyDB.Orders.Deleted(item);

}

var r = MyDB.Orders.SaveChanges();

if (r != 0)

{

Deleted(cust);

var res = SaveChanges();

return res;

}

else

{

return r;

}

}

public int UpdateOrder(int code)

{

//חיפוש הלקוחה הזו

//עידכון השורות בטבלת הזמנות שמקושרות אליה,

var custo = GetList().FirstOrDefault(u => u.CodeCustomers == 9);

var details = MyDB.Orders.GetList().Where(o => o.CodeCustomers == custo).ToList();

Adress address = MyDB.Adress.GetList().FirstOrDefault(predicate: u => custo.CodeAdress == u);

MyDB.Adress.Update(address);

foreach (var item in details)

{

MyDB.Orders.Update(item);

}

var r = MyDB.Orders.SaveChanges();

if (r != 0)

{

Update(custo);

var res = SaveChanges();

return res;

}

else

{

return r;

}

}

* מחלקת צירוף פרטי הזמנה(ordersDB)

public OrdersDB() : base("Orders")

{

}

///שליפת כל השורות

public List<Orders> GetList()

{

if (list.Count() == 0) { Select(); }

var r = list.Cast<Orders>().ToList();

return r;

}

protected override BaseEntity CreateModel()

{

{

Orders orders = new Orders();

orders.CodeOrder = (int)reader["CodeOrder"];

orders.CodeCustomers = MyDB.Customers.getCustomerDetailsByCode((int)reader["CodeCustomers"]);

orders.Remarks = reader["Remarks"].ToString();

orders.CodeHairdo = MyDB.Hairdo.getHairdoByCode((int)reader["CodeHairdo"]);

return orders;

}

}

///שליפת הזמנה לפי קוד

public Orders getOrderByCode(int v)

{

return GetList().FirstOrDefault(item => item.CodeOrder == v);

}

//שליפת הזמנה לפי קוד פרטי הזמנה(קוד של האוביקט

//(המצורף customer

public Orders getOrderByCodeCustomer(int c)

{

return GetList().FirstOrDefault(item => item.CodeCustomers.CodeCustomers == c);

}

protected override string CreateInsertCommand(BaseEntity item)

{

throw new NotImplementedException();

}

protected override string CreateUpdateCommand(BaseEntity item)

{

throw new NotImplementedException();

}

protected override string CreateDeletedCommand(BaseEntity item)

{

throw new NotImplementedException();

}

* מחלקת הנחה(saleDB)

public SaleDB() : base("Sale")

{

}

protected override BaseEntity CreateModel()

{

Sale entity = new Sale();

entity.CodeSale = (int)reader["CodeSale"];

entity.NameSale = reader["NameSale"].ToString();

entity.TheSale = (int)reader["TheSale"];

return entity;

}

public Sale GetSaleByCode(int v)

{

return GetList().FirstOrDefault(item => item.CodeSale == v);

}

//שליפת כל ההנחות

public List<Sale> GetList()

{

if (list.Count() == 0) { Select(); }

return list.Cast<Sale>().ToList();

}

//שליפת הנחה לפי קוד

public Sale getSaleByCode(int v)

{

return GetList().FirstOrDefault(item => item.CodeSale == v);

}

protected override string CreateInsertCommand(BaseEntity item)

{

throw new NotImplementedException();

}

protected override string CreateUpdateCommand(BaseEntity item)

{

throw new NotImplementedException();

}

protected override string CreateDeletedCommand(BaseEntity item)

{

throw new NotImplementedException();

}

* מחלקת סוג תסרוקת(HairdoDB)

public HairdoDB() : base("hairdo")

{

}

protected override BaseEntity CreateModel()

{

{

Hairdo hairdo = new Hairdo();

hairdo.CodeHairdo = (int)reader["CodeHairdo"];

hairdo.DescriptionHairdo = reader["DescriptionHairdo"].ToString();

hairdo.Price = (int)reader["Price"];

return hairdo;

}

}

//שליפת פרטי כל תסרוקות

public List<Hairdo> GetList()

{

if (list.Count() == 0) { Select(); }

return list.Cast<Hairdo>().ToList();

}

//שליפת פרטי תסרוקת על ידי קוד

public Hairdo getHairdoByCode(int v)

{

return GetList().FirstOrDefault(item => item.CodeHairdo == v);

}

//שליפת פרטי תסרוקת על ידי שם/תיאור התסרוקת

public Hairdo GetCodeHairdoByName(string name)

{

return GetList().FirstOrDefault(item => item.DescriptionHairdo == name);

}

//מציאת קוד חדש לתסרוקת שאינו קיים באקסס

public int GetCodeToHairdo()

{

if (MyDB.Hairdo.GetList().Count == 0) return 1;

return MyDB.Hairdo.GetList().Max(X => X.CodeHairdo) + 1;

}

//הוספת תסרוקת

public int AddHairdo(Hairdo hairdo)

{

MyDB.Hairdo.Add(hairdo);

var res = MyDB.Hairdo.SaveChanges();

return res;

}

protected override string CreateInsertCommand(BaseEntity item)

{

throw new NotImplementedException();

}

protected override string CreateUpdateCommand(BaseEntity item)

{

throw new NotImplementedException();

}

protected override string CreateDeletedCommand(BaseEntity item)

{

throw new NotImplementedException();

}

* מחלקת כתובת(AdressDB)

public class AdressDB : BaseDB

{

public AdressDB() : base("Adress") { }

protected override BaseEntity CreateModel()

{

Adress entity = new Adress();

entity.CodeAdress = Convert.ToInt32(reader["CodeAdress"]);

entity.Street = reader["Street"].ToString();

entity.CodeCity = MyDB.Cities.getCitiesDById((int)reader["CodeCity"]);

return entity;

}

//internal Adress GetAdressByCodereader(int v)

//{

// return GetList().FirstOrDefault(item => item.CodeCustomers == v);

//}

public List<Adress> GetList()

{

if (list.Count() == 0) { Select(); }

var l = list.Cast<Adress>().ToList();

return l;

}

public Adress GetAdressByCode(int v)

{

return GetList().FirstOrDefault(item => item.CodeAdress == v);

}

public Adress GetAdresssByName(string name)

{

return GetList().FirstOrDefault(item => item.Street == name);

}

public int GetCodeToAdress()

{

if (MyDB.Adress.GetList().Count == 0) return 1;

return MyDB.Adress.GetList().Max(X => X.CodeAdress) + 1;

}

public int AddAdress(Adress adress)

{

Add(adress);

int r = SaveChanges();

return r;

}

protected override string CreateInsertCommand(BaseEntity item)

{

throw new NotImplementedException();

}

protected override string CreateUpdateCommand(BaseEntity item)

{

throw new NotImplementedException();

}

protected override string CreateDeletedCommand(BaseEntity item)

{

throw new NotImplementedException();

}

}

## 

## **שכבת שרותי רשת**

שכבת שרותי הרשת בפרויקט נבנתה בטכנולוגית WCF. בטכנולוגיה זו השירות כולל Interface + מימוש במחלקה.

(שכבת שירותי הרשת משתמשת באובייקטים של שכבת הModel ובפונקציות שבשכבת הViewModel).

**IService - :**

namespace wcf\_server

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService1" in both code and config file together.

[ServiceContract]

public interface IService1

{

[OperationContract]

Sale getSaleByCode(int value);

[OperationContract]

List<Cities> getAllCities();

[OperationContract]

Adress GetAdressByCode(int v);

[OperationContract]

int GetCodeToHairdo();

[OperationContract]

List<Customers> GetAllCustomers();

[OperationContract]

int AddHairdo(Hairdo hairdo);

[OperationContract]

List<Hairdo> GetAllHairdos();

[OperationContract]

List<CustomerDetails> GetAllCustomerDetails();

[OperationContract]

List<Sale> GetAllSales();

[OperationContract]

Adress GetAdresssByName(string name);

[OperationContract]

Cities GetCodeCityByName(string c);

[OperationContract]

int UpdateOrder(int code);

[OperationContract]

int AddAdress(Adress adress);

[OperationContract]

int GetCodeToAdress();

[OperationContract]

List<Orders> GetAllOrders();

[OperationContract]

CustomerDetails GetCodeCustomerDetailsByName(string name);

[OperationContract]

int GetCodeToCustomers();

[OperationContract]

int AddCustomers(Customers customers, int codeSale1);

[OperationContract]

List<Adress> GetAllAddresses();

[OperationContract]

Orders GetOrderByCodeCustomer(int c);

[OperationContract]

int GetCodeToCities();

[OperationContract]

int AddCity(Cities cities);

[OperationContract]

int deleteCustomersWithAllDetails(int code);

[OperationContract]

Customers getCustomerDetailsByCode(int code);

[OperationContract]

Orders getOrderByCode(int v);

[OperationContract]

int AddCust(CustomerDetails c);

[OperationContract]

int GetCodeToCustomersD();

}

}

**מימוש במחלקה:**

namespace wcf\_server

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "Service1" in both code and config file together.

public class Service1 : IService1

{

//sales

public List<Sale> GetAllSales()

{

return MyDB.Sale.GetList();

}

public Sale getSaleByCode(int value)//שליפת הנחה לפי קוד

{

return MyDB.Sale.GetSaleByCode(value);

}

//cities

public List<Cities> getAllCities()

{

return MyDB.Cities.GetList();

}

public Cities GetCodeCityByName(string c)

{

return MyDB.Cities.GetCodeCityByName(c);

}

public int GetCodeToCities()

{

return MyDB.Cities.GetCodeToCities();

}

public int AddCity(Cities cities)

{

return MyDB.Cities.AddCity(cities);

}

//adress

public List<Adress> GetAllAddresses()

{

return MyDB.Adress.GetList();

}

public int GetCodeToAdress()

{

return MyDB.Adress.GetCodeToAdress();

}

public int AddAdress(Adress adress)

{

return MyDB.Adress.AddAdress(adress);

}

public Adress GetAdresssByName(string name)

{

return MyDB.Adress.GetAdresssByName(name);

}

public Adress GetAdressByCode(int v)

{

return MyDB.Adress.GetAdressByCode(v);

}

//hairdo

public List<Hairdo> GetAllHairdos()

{

return MyDB.Hairdo.GetList();

}

public int GetCodeToHairdo()

{

return MyDB.Hairdo.GetCodeToHairdo();

}

public int AddHairdo(Hairdo hairdo)

{

return MyDB.Hairdo.AddHairdo(hairdo);

}

//customers

public List<Customers> GetAllCustomers()

{

return MyDB.Customers.GetList();

}

public int AddCustomers(Customers customers, int codeSale1)

{

return MyDB.Customers.AddCustomers(customers, codeSale1);

}

public int UpdateOrder(int code)

{

return MyDB.Customers.UpdateOrder(code);

}

public int GetCodeToCustomers()

{

return MyDB.Customers.GetCodeToCustomers();

}

public int deleteCustomersWithAllDetails(int code)

{

return MyDB.Customers.deleteCustomersWithAllDetails(code);

}

public Customers getCustomerDetailsByCode(int code)

{

return MyDB.Customers.getCustomerDetailsByCode((int)code);

}

//customerDetails

public List<CustomerDetails> GetAllCustomerDetails()

{

return MyDB.CustomersDetails.GetList();

}

public CustomerDetails GetCodeCustomerDetailsByName(string name)

{

return MyDB.CustomersDetails.GetCodeCustomerDetailsByName(name);

}

public int AddCust(CustomerDetails c)

{

return MyDB.CustomersDetails.AddCust(c);

}

public int GetCodeToCustomersD()

{

return MyDB.CustomersDetails.GetCodeToCustomersD();

}

//orders

public List<Orders> GetAllOrders()

{

return MyDB.Orders.GetList();

}

public Orders GetOrderByCodeCustomer(int c)

{

return MyDB.Orders.getOrderByCodeCustomer(c);

}

public Orders getOrderByCode(int v)

{

return MyDB.Orders.getOrderByCode(v);

}

}

}

HOST

כדי ששירותי הרשת שבניתי יוחזקו באוויר וניתן יהיה לגשת אליהם מהלקוח, בניתי אפליקציית Host בצד שרת. שתפקידה להפעיל את שירות הרשת ולהשאירו פתוח לגישה.

האפליקציה נכתבה בConsole Application.

קוד התוכנית:

namespace MyHost

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("This is my server");

ServiceHost host = new ServiceHost(typeof(Service1));

host.Open();

Console.Read();

}

}

}

התוכנית נדרשת להשתמש במידע אודות שירותי הרשת

מידע זה נמצא בקובץ App.config בפרויקט הHost

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<appSettings>

<add key="aspnet:UseTaskFriendlySynchronizationContext" value="true"/>

</appSettings>

<system.web>

<compilation debug="true"/>

</system.web>

<!-- When deploying the service library project, the content of the config file must be added to the host's

app.config file. System.Configuration does not support config files for libraries. -->

<system.serviceModel>

<services>

<service name="wcf\_server.Service1">

<host>

<baseAddresses>

<add baseAddress="http://localhost:8733/Design\_Time\_Addresses/wcf\_server/Service1/"/>

</baseAddresses>

</host>

<!-- Service Endpoints -->

<!-- Unless fully qualified, address is relative to base address supplied above -->

<endpoint address="" binding="basicHttpBinding" contract="wcf\_server.IService1">

<!--

Upon deployment, the following identity element should be removed or replaced to reflect the

identity under which the deployed service runs. If removed, WCF will infer an appropriate identity

automatically.

-->

<identity>

<dns value="localhost"/>

</identity>

</endpoint>

<!-- Metadata Endpoints -->

<!-- The Metadata Exchange endpoint is used by the service to describe itself to clients. -->

<!-- This endpoint does not use a secure binding and should be secured or removed before deployment -->

<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior>

<!-- To avoid disclosing metadata information,

set the values below to false before deployment -->

<serviceMetadata httpGetEnabled="True" httpsGetEnabled="True"/>

<!-- To receive exception details in faults for debugging purposes,

set the value below to true. Set to false before deployment

to avoid disclosing exception information -->

<serviceDebug includeExceptionDetailInFaults="False"/>

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2"/>

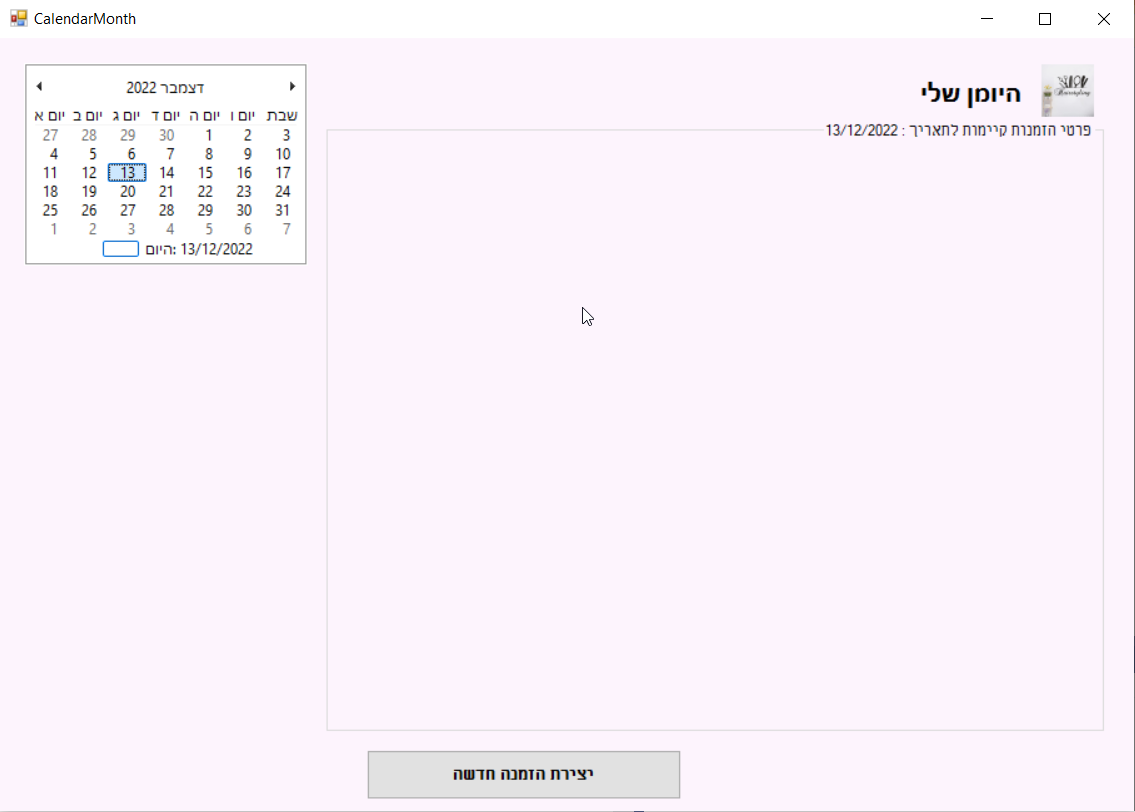
</startup>

</configuration>

## **צד לקוח**

בפרויקט בניתי אפליקציה עבור ספריות עצמאיות.

האפליקציה נכתבה ב- Windows Forms.



מסך זה הוא מסך היומן ובו לוח שנה.

היומן מעודכן ליום הנוכחי.

ישנה אפשרות לשנות את התאריך, ולהציג דו"ח של הפגישות לתאריך המבוקש, בשינוי התאריך בלוח החודשי.

בטופס זה ישנה אופציה לעדכון ולמחיקה.

הקוד של C#:

public partial class CalendarMonth : Form

{

List<Customers> lc = new List<Customers>();

List<Customers> temp;

public CalendarMonth()

{

InitializeComponent();

//ordersInThisDay();

}

public void PageAddOrder()

{

AddOrder ad1 = new AddOrder();

ad1.Show();

this.Hide();

ad1.FormClosed += (s, args) => this.Close();

ad1.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

//עדכון סטטוס בגלובל =הוספה

General.Status = 1;

PageAddOrder();

}

bool flagL;

private void CalendarMonth\_Load(object sender, EventArgs e)

{

General.dateTreat = DateTime.Today.Date;

DateRangeEventArgs dt = new DateRangeEventArgs(DateTime.Today.Date, DateTime.Today.Date);

flagL = false;

ordersInThisDay(sender, dt);

if (flagL == true)

noOrders.Visible = true;

labeDate.Text = DateTime.Today.ToShortDateString().ToString();

labelToday.Text = DateTime.Today.ToShortDateString().ToString();

}

private void button1\_Click(object sender, EventArgs e)

{

General.Status = 1;

AddOrder ad1 = new AddOrder();

ad1.Show();

this.Hide();

ad1.FormClosed += (s, args) => this.Close();

ad1.Show();

}

//הזמנות ליום מבוקש

public void ordersInThisDay(object sender, DateRangeEventArgs e)

{

labeDate.Text = e.End.ToShortDateString().ToString();

General.dateTreat = e.Start.Date;

lc = General.sharat.GetAllCustomers().ToList();

//יש לשים לב אם משווים תאריך שהשעה לא מאופסת - יש להשוות "ידנית" את השנה חודש ויום

var temp = lc.Where(p => p.DateT == General.dateTreat).ToList();

int moneTreat;

moneTreat = lc.Count;

Customers m = lc.First();

if (temp.Count() > 0)

{

flagL = false;

//הצגת ה-panels

List<Panel> lp = new List<Panel>

{

panel1,

panel2,

panel3,

panel4,

panel5,

panel6,

panel7,

panel8,

panel9,

panel10,

panel11,

panel12

};

List<Customers> t2 = new List<Customers>();

t2 = temp;

foreach (var i in temp)

{

lp.First().Visible = true;

lp.RemoveAt(0);

}

//הצגת עיקרי ההזמנה ב-labels

//list of labels to show the name customer

List<Label> listName = new List<Label>

{

l1,

l2,

l3,

l4,

l5,

l6,

l7,

l8,

l9,

l10,

l11,

l12

};

//list of labels to show the price treatment

List<Label> listPrice = new List<Label>

{

labelA,

labelB,

labelC,

labelD,

labelE,

labelF,

labelG,

labelH,

labelI,

labelG,

labelK,

labelL

};

//list of labels to show the price treatment

List<Label> listAdress = new List<Label>

{

lla,

llb,

llc,

lld,

lle,

llf,

llg,

llh,

lli,

llj,

llk,

lll

};

//list of labels to show the price treatment

List<Label> listCode = new List<Label>

{

label1,

label3,

label4,

label5,

label6,

label7,

label8,

label9,

label10,

label11,

label12,

label13

};

foreach (var i in temp)

{

listName.First().Visible = true;

listName.First().Text = i.CodeCustomerDetails.NameC;

listName.RemoveAt(0);

listPrice.First().Visible = true;

listPrice.First().Text = i.FinalPrice.ToString();

listPrice.RemoveAt(0);

listCode.First().Visible = true;

listCode.First().Text = i.CodeCustomers.ToString();

listCode.RemoveAt(0);

if (i.IsMyClinic != true)

{

listAdress.First().Visible = true;

listAdress.First().Text = i.CodeAdress.Street + "," + i.CodeAdress.CodeCity.NameCity;

}

else

{

listAdress.First().Visible = true;

listAdress.First().Text = "הטיפול נערך בקליניקה";

}

listAdress.RemoveAt(0);

}

for (int i = 0; i < lp.Count(); i++)

{

lp.First().Visible = false;

lp.RemoveAt(0);

}

}

else if (e.End.Date == DateTime.Today.Date)

{

flagL = true;

}

else

{

flagL = true;

List<Panel> lpt = new List<Panel>

{

panel1,

panel2,

panel3,

panel4,

panel5,

panel6,

panel7,

panel8,

panel9,

panel10,

panel11,

panel12,

panel13

};

for (int i = 0; i < lpt.Count(); i++)

{

lpt.First().Visible = false;

lpt.RemoveAt(0);

}

MessageBox.Show("אין עדיין הזמנות ליום זה");

}

if (flagL == true)

noOrders.Visible = true;

else

noOrders.Visible = false;

}

private void monthCalendar1\_DateSelected(object sender, DateRangeEventArgs e)

{

ordersInThisDay(sender, e);

}

private void button15\_Click(object sender, EventArgs e)

{

Orders o = new Orders();

o = General.sharat.GetAllOrders().ElementAt(1);

General.UpdateOrder = o;

AddOrder order = new AddOrder();

}

/\*private void button38\_Click(object sender, EventArgs e)

{

ShowDayOrder();

}\*/

//הצגת פרטים נוספים על ההזמנה

public void ShowDayOrder()

{

DayOrders do1 = new DayOrders();

do1.Show();

this.Hide();

do1.FormClosed += (s, args) => this.Close();

do1.Show();

}

private void button40\_Click(object sender, EventArgs e)

{

General.codeOrederToDisplay = int.Parse(label1.Text);

ShowDayOrder();

}

private void button41\_Click(object sender, EventArgs e)

{

General.codeOrederToDisplay = int.Parse(label3.Text);

ShowDayOrder();

}

private void button42\_Click(object sender, EventArgs e)

{

General.codeOrederToDisplay = int.Parse(label4.Text);

ShowDayOrder();

}

private void button39\_Click(object sender, EventArgs e)

{

General.codeOrederToDisplay = int.Parse(label5.Text);

ShowDayOrder();

}

private void button43\_Click(object sender, EventArgs e)

{

General.codeOrederToDisplay = int.Parse(label6.Text);

ShowDayOrder();

}

private void button44\_Click(object sender, EventArgs e)

{

General.codeOrederToDisplay = int.Parse(label7.Text);

ShowDayOrder();

}

private void button45\_Click(object sender, EventArgs e)

{

General.codeOrederToDisplay = int.Parse(label8.Text);

ShowDayOrder();

}

private void button46\_Click(object sender, EventArgs e)

{

General.codeOrederToDisplay = int.Parse(label9.Text);

ShowDayOrder();

}

private void button47\_Click(object sender, EventArgs e)

{

General.codeOrederToDisplay = int.Parse(label10.Text);

ShowDayOrder();

}

private void button48\_Click(object sender, EventArgs e)

{

General.codeOrederToDisplay = int.Parse(label11.Text);

ShowDayOrder();

}

private void button49\_Click(object sender, EventArgs e)

{

General.codeOrederToDisplay = int.Parse(label12.Text);

ShowDayOrder();

}

private void button50\_Click(object sender, EventArgs e)

{

General.codeOrederToDisplay = int.Parse(label13.Text);

ShowDayOrder();

}

//פעולות עריכת הזמנה

private void button6\_Click(object sender, EventArgs e)

{

int n = int.Parse(label3.Text);

Customers c = new Customers();

c = General.sharat.getCustomerDetailsByCode(n);

Orders ooo = new Orders();

ooo = General.sharat.GetOrderByCodeCustomer(c.CodeCustomers);

General.UpdateOrder = ooo;

AddOrder ad5 = new AddOrder();

ad5.Show();

this.Hide();

ad5.FormClosed += (s, args) => this.Close();

ad5.Show();

}

private void button8\_Click(object sender, EventArgs e)

{

int n = int.Parse(label4.Text);

Customers c = new Customers();

c = General.sharat.getCustomerDetailsByCode(n);

Orders ooo = new Orders();

ooo = General.sharat.GetOrderByCodeCustomer(c.CodeCustomers);

General.UpdateOrder = ooo;

AddOrder ad5 = new AddOrder();

}

private void button10\_Click(object sender, EventArgs e)

{

int n = int.Parse(label5.Text);

Customers c = new Customers();

c = General.sharat.getCustomerDetailsByCode(n);

Orders ooo = new Orders();

ooo = General.sharat.GetOrderByCodeCustomer(c.CodeCustomers);

General.UpdateOrder = ooo;

AddOrder ad5 = new AddOrder();

}

private void button12\_Click(object sender, EventArgs e)

{

int n = int.Parse(label6.Text);

Customers c = new Customers();

c = General.sharat.getCustomerDetailsByCode(n);

Orders ooo = new Orders();

ooo = General.sharat.GetOrderByCodeCustomer(c.CodeCustomers);

General.UpdateOrder = ooo;

AddOrder ad5 = new AddOrder();

}

private void button14\_Click(object sender, EventArgs e)

{

int n = int.Parse(label7.Text);

Customers c = new Customers();

c = General.sharat.getCustomerDetailsByCode(n);

Orders ooo = new Orders();

ooo = General.sharat.GetOrderByCodeCustomer(c.CodeCustomers);

General.UpdateOrder = ooo;

AddOrder ad5 = new AddOrder();

}

private void button16\_Click(object sender, EventArgs e)

{

int n = int.Parse(label8.Text);

Customers c = new Customers();

c = General.sharat.getCustomerDetailsByCode(n);

Orders ooo = new Orders();

ooo = General.sharat.GetOrderByCodeCustomer(c.CodeCustomers);

General.UpdateOrder = ooo;

AddOrder ad5 = new AddOrder();

}

private void button18\_Click(object sender, EventArgs e)

{

int n = int.Parse(label9.Text);

Customers c = new Customers();

c = General.sharat.getCustomerDetailsByCode(n);

Orders ooo = new Orders();

ooo = General.sharat.GetOrderByCodeCustomer(c.CodeCustomers);

General.UpdateOrder = ooo;

AddOrder ad5 = new AddOrder();

}

private void button20\_Click(object sender, EventArgs e)

{

int n = int.Parse(label10.Text);

Customers c = new Customers();

c = General.sharat.getCustomerDetailsByCode(n);

Orders ooo = new Orders();

ooo = General.sharat.GetOrderByCodeCustomer(c.CodeCustomers);

General.UpdateOrder = ooo;

AddOrder ad5 = new AddOrder();

}

private void button22\_Click(object sender, EventArgs e)

{

int n = int.Parse(label11.Text);

Customers c = new Customers();

c = General.sharat.getCustomerDetailsByCode(n);

Orders ooo = new Orders();

ooo = General.sharat.GetOrderByCodeCustomer(c.CodeCustomers);

General.UpdateOrder = ooo;

AddOrder ad5 = new AddOrder();

}

private void button24\_Click(object sender, EventArgs e)

{

int n = int.Parse(label12.Text);

Customers c = new Customers();

c = General.sharat.getCustomerDetailsByCode(n);

Orders ooo = new Orders();

ooo = General.sharat.GetOrderByCodeCustomer(c.CodeCustomers);

General.UpdateOrder = ooo;

AddOrder ad5 = new AddOrder();

}

private void button26\_Click(object sender, EventArgs e)

{

int n = int.Parse(label13.Text);

Customers c = new Customers();

c = General.sharat.getCustomerDetailsByCode(n);

Orders ooo = new Orders();

ooo = General.sharat.GetOrderByCodeCustomer(c.CodeCustomers);

General.UpdateOrder = ooo;

AddOrder ad5 = new AddOrder();

}

private void button3\_Click(object sender, EventArgs e)

{

int n = int.Parse(label1.Text);

Customers c = new Customers();

c = General.sharat.getCustomerDetailsByCode(n);

Orders ooo = new Orders();

ooo = General.sharat.getOrderByCode(222);

//ooo = MyDB.Orders.getOrderByCodeCustomer(c.CodeCustomers);

General.UpdateOrder = ooo;

AddOrder ad5 = new AddOrder();

ad5.Show();

this.Hide();

ad5.FormClosed += (s, args) => this.Close();

ad5.Show();

}

//פעולות מחיקת הזמנה

private void deleteOrder(object sender, EventArgs e)

{

//הצגה של הפרטים הנוספים ,

//וכן כפתורים של עריכה ומחיקה

var res = General.sharat.deleteCustomersWithAllDetails(21);

if (res == 1)

{

MessageBox.Show("נמחק");

}

else

{

MessageBox.Show("נסה שוב");

}

}

private void button5\_Click(object sender, EventArgs e)

{

deleteOrder(sender, e);

}

private void button4\_Click(object sender, EventArgs e)

{

deleteOrder(sender, e);

}

private void button9\_Click(object sender, EventArgs e)

{

deleteOrder(sender, e);

}

private void button11\_Click(object sender, EventArgs e)

{

deleteOrder(sender, e);

}

private void button13\_Click(object sender, EventArgs e)

{

deleteOrder(sender, e);

}

private void button15\_Click\_1(object sender, EventArgs e)

{

deleteOrder(sender, e);

}

private void button17\_Click(object sender, EventArgs e)

{

deleteOrder(sender, e);

}

private void button19\_Click(object sender, EventArgs e)

{

deleteOrder(sender, e);

} private void button21\_Click(object sender, EventArgs e)

{

deleteOrder(sender, e);

}

private void button23\_Click(object sender, EventArgs e)

{

deleteOrder(sender, e);

}

private void button25\_Click(object sender, EventArgs e)

{

deleteOrder(sender, e);

}

private void button7\_Click(object sender, EventArgs e)

{

deleteOrder(sender, e);

}

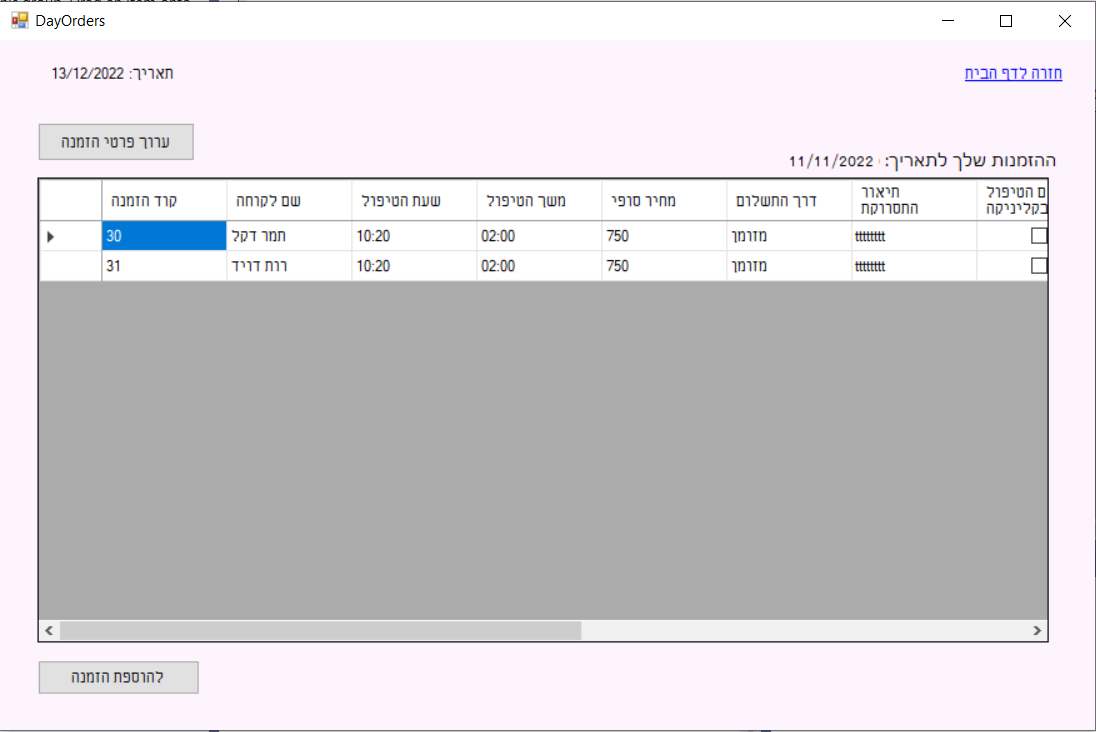
private void button38\_Click(object sender, EventArgs e)

{

}

}

בלחיצה על פרטים נוספים ניתן יהיה לראות פרטים נוספים על לקוחה זו ועל הזמנתה.



public DayOrders()

{

InitializeComponent();

}

List<Customers> c = new List<Customers>();

private void DayOrders\_Load(object sender, EventArgs e)

{

//קבלת תאריך מהזמנות קודמות ולפי זה לשלוף

int monthD = General.dateTreat.Month;

int yearD = General.dateTreat.Year;

int dayD = General.dateTreat.Day;

DateTime d = new DateTime(yearD, monthD, dayD);

List<Customers> cs = new List<Customers>();

cs = MyDB.Customers.GetList();

//יש לשים לב אם משווים תאריך שהשעה לא מאופסת - יש להשוות "ידנית" את השנה חודש ויום

//customers

var customers = cs.Where(p => p.DateT == d).ToList();

var sale = MyDB.Sale.GetList();

var addresses = MyDB.Adress.GetList();

var q = from c in customers

select new

{

c.CodeCustomers,

c.CodeCustomerDetails.NameC,

c.Hour1,

c.HowLong,

c.FinalPrice,

c.WayofPay,

c.RemarksC,

c.IsMyClinic,

c.DateT,

c.CodeSale.NameSale,

mail = c.CodeCustomerDetails.Mail1,

phone1 = c.CodeCustomerDetails.NPhone1,

phone2 = c.CodeCustomerDetails.NPhone2,

address = c.CodeAdress.Street+" "+ c.CodeAdress.CodeCity.NameCity

};

dataGridView1.DataSource = q.ToList();

dataGridView1.Columns[0].HeaderText = "קוד הזמנה";

dataGridView1.Columns[1].HeaderText = "שם לקוחה";

dataGridView1.Columns[2].HeaderText = "שעת הטיפול";

dataGridView1.Columns[3].HeaderText = "משך הטיפול";

dataGridView1.Columns[4].HeaderText = "מחיר סופי";

dataGridView1.Columns[5].HeaderText = "דרך התשלום";

dataGridView1.Columns[6].HeaderText = "תיאור התסרוקת";

dataGridView1.Columns[7].HeaderText = "האם הטיפול בקליניקה";

dataGridView1.Columns[8].HeaderText = "תאריך הטיפול";

dataGridView1.Columns[9].HeaderText = "הנחה";

dataGridView1.Columns[10].HeaderText = "מייל";

dataGridView1.Columns[11].HeaderText = "פלאפון 1";

dataGridView1.Columns[12].HeaderText = "פלאפון 2";

dataGridView1.Columns[13].HeaderText = "כתובת";

label2.Text = d.ToString();

//התאריך של היום

DateTime dtoday1 = DateTime.Today;

label5.Text = dtoday1.ToShortDateString().ToString();

}

private void button2\_Click(object sender, EventArgs e)

{

AddOrder ad = new AddOrder();

ad.Show();

this.Hide();

ad.FormClosed += (s, args) => this.Close();

ad.Show();

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

CalendarMonth cm1 = new CalendarMonth();

cm1.Show();

this.Hide();

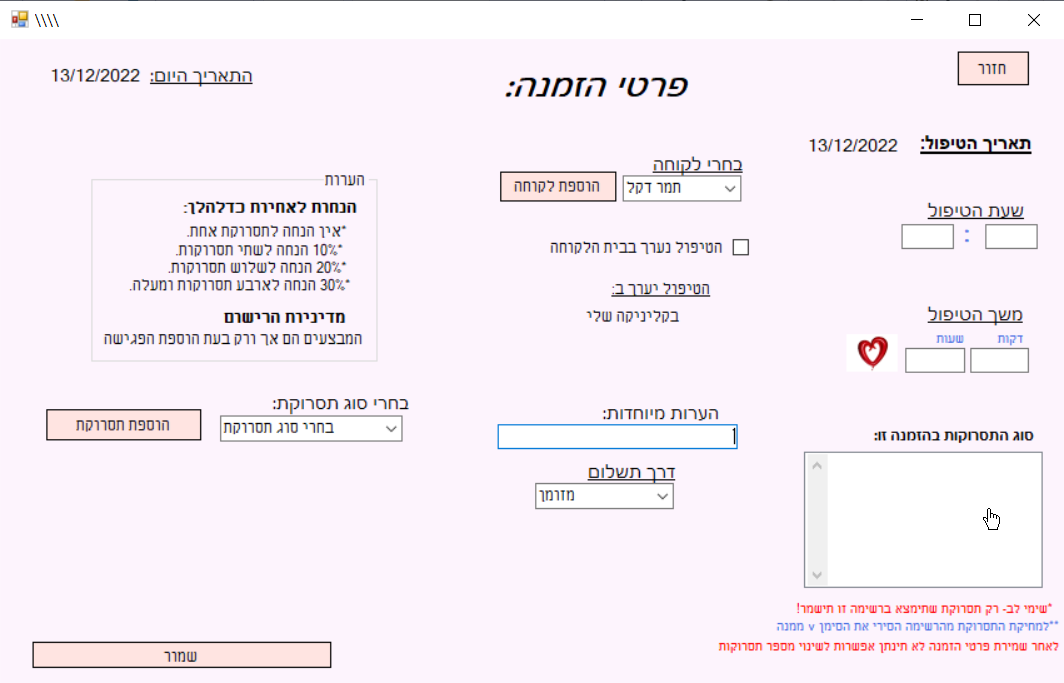
cm1.FormClosed += (s, args) => this.Close();

cm1.Show();

}

בלחיצה על הוספת הזמנה נפתח טופס להוספת הזמנה

פעולה זו ניתן לעשות גם ממסך לוח השנה))



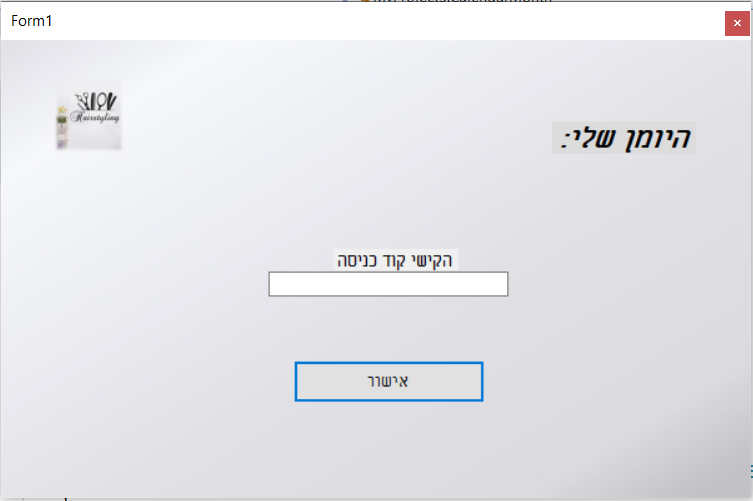
## **מדריך למשתמש**

מנהלת יקרה

אפליקציית GridDiary תוכננה במיוחד כדי לספק לך חווית ניהול עסק בצורה מהנה נגישה ונוחה, באפליקציה זו תוכלי לדפדף ביומנך שבנוי על לוח שנה עדכני, לצפות במספר הפגישות בכל יום להתעדכן מהן הפגישות במידה ויש, לקבוע פגישות חדשות ולמחוק פגישות בלתי רצויות.

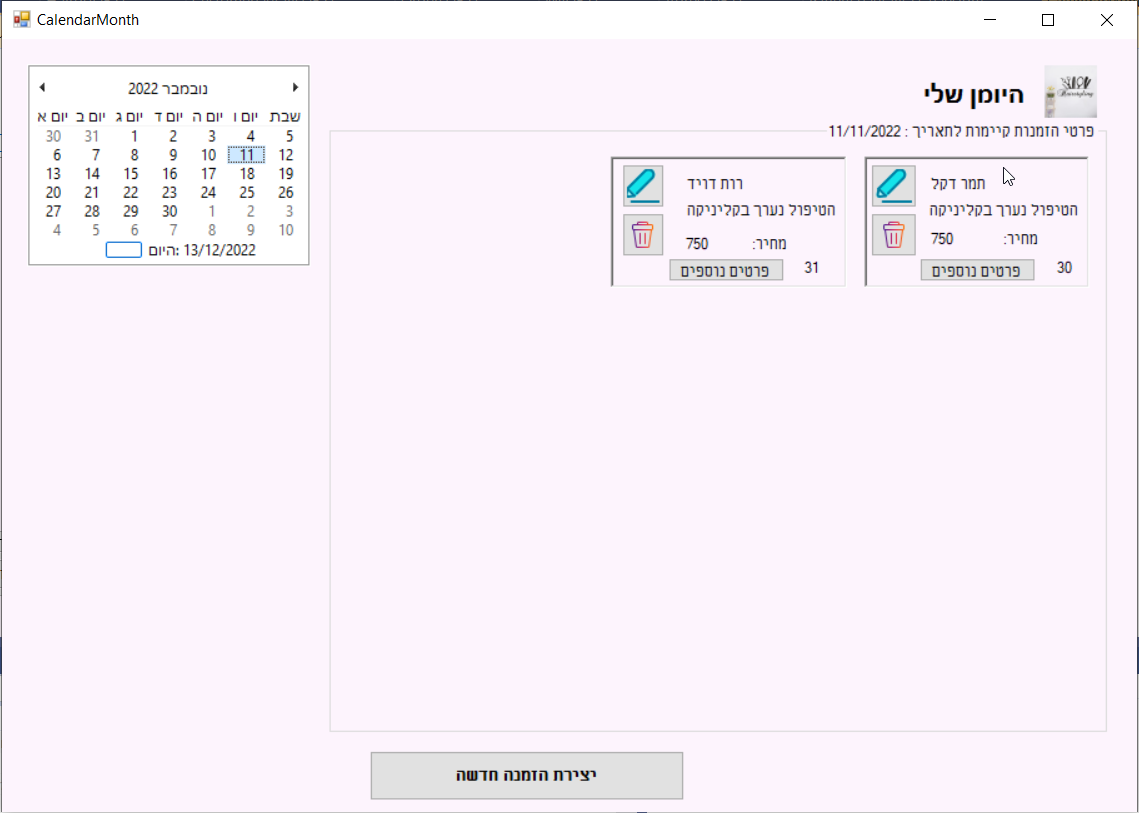
צוות הפיתוח מאחל לך שימוש מהנה ויעיל באפליקציה.

צילומי מסכים של האפליקציה:



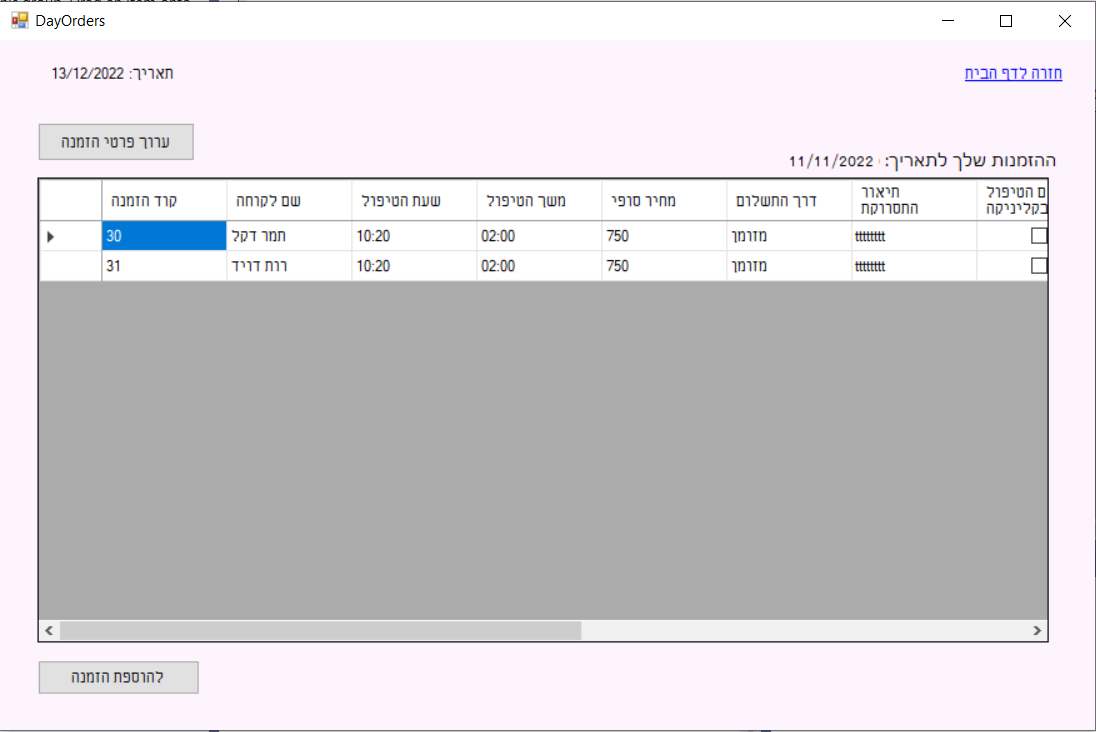
על הספרית להזין קוד ולאשר אותו.

במידה והקוד לא יאומת האפליקציה תשיב "הסיסמה שגויה" ואופציה לתיקון.

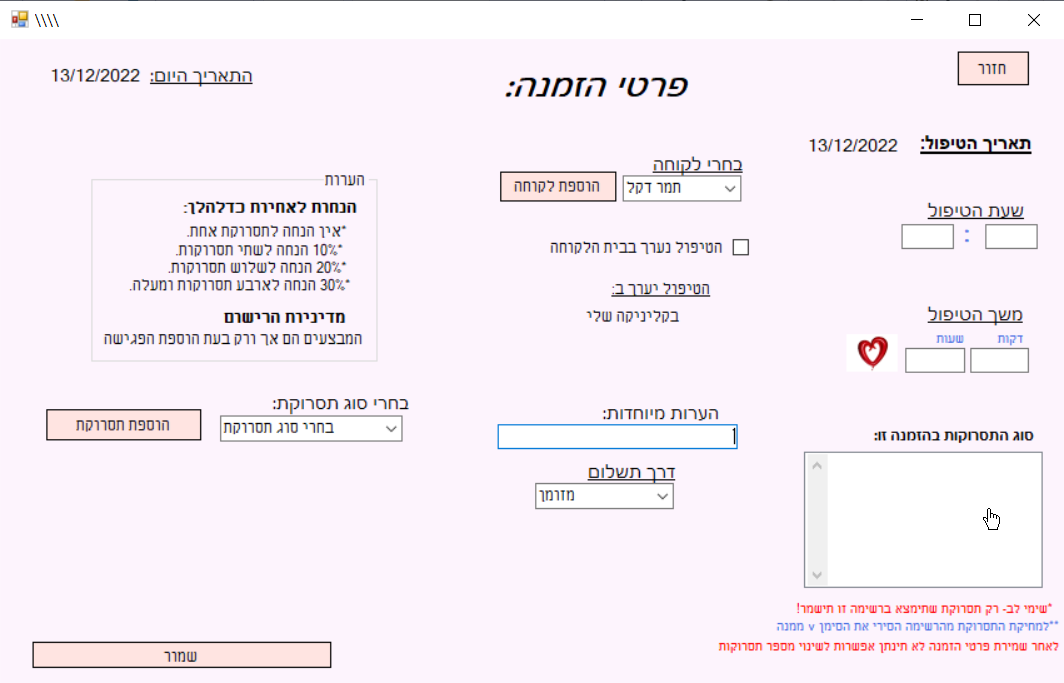


מסך היומן בו מוצגות הזמנות קיימות אם ישנן כאלה.

לכל הזמנה ישנו אייקון פח אשפה -בלחיצה עליו תמחק ההזמנה. וכן אייקון עיפרון- בלחיצה עליו יפתח טופס מלא בפרטי ההזמנה לעדכון הפרטים הרצויים.



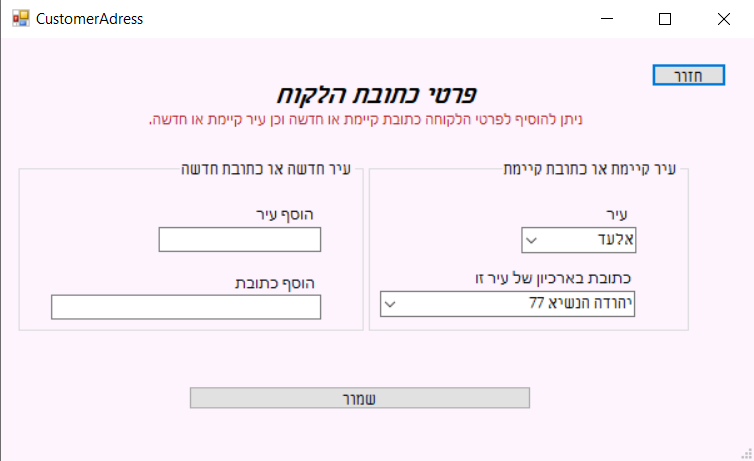
בלחיצה על פרטים נוספים יופיע מסך ובו פרטי ההזמנה. ניתן ללחוץ על יצירת עדכון וכן יצירת הזמנה חדשה.



בלחיצה על הוספה נפתח מסך זה.

בו יש למלא את כל הפרטים ובלחיצה על "שמור" ההזמנה תישמר ונחזור למסך היומן.

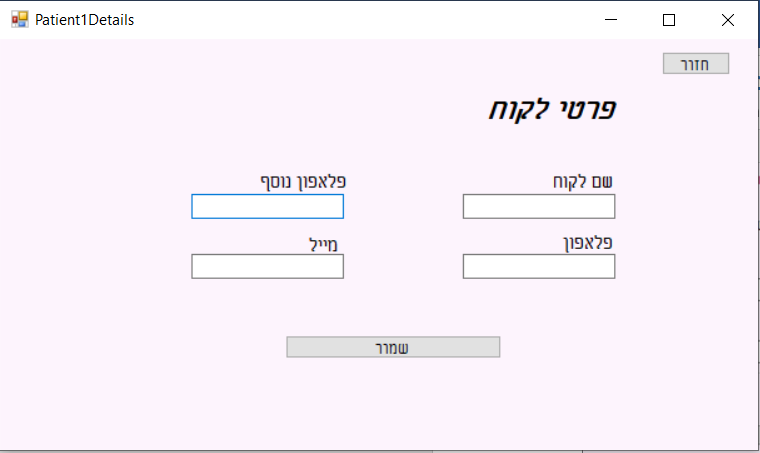
במידה והטיפול בבית הלקוחה יש לסמן את המשפט ב-וי.



אז יוצג טופס ובו יש לבחור מהרשימה את העיר והכתובת הרצויות.

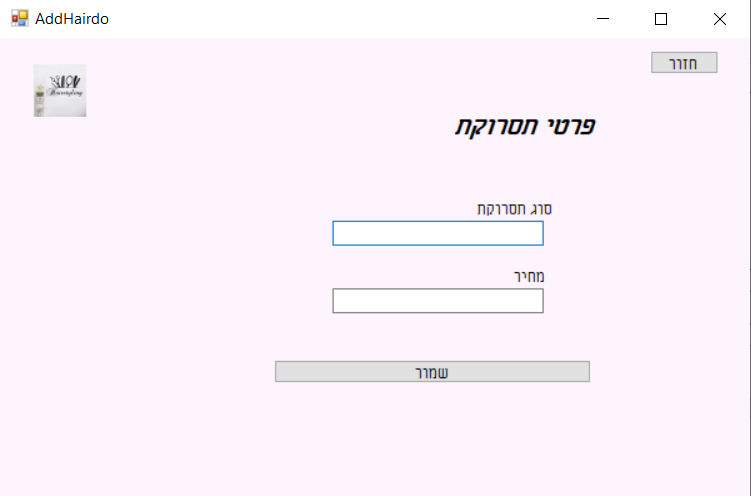
במידה והן לא קיימות ברשימה יש למלא בתיבה המיועדת לכך.

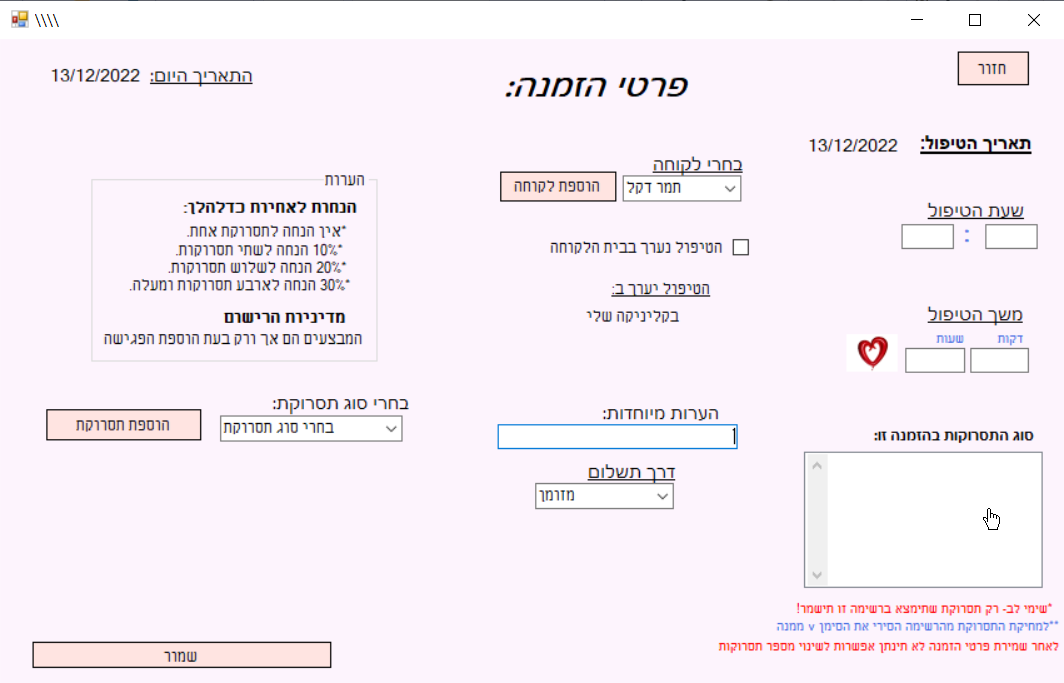
וכמובן לשמור.



בלחיצה על "הוספת לקוחה חדשה" יפתח טופס בו יש למלא את הפרטים ולשמור

(טופס זה מאפס את הפרטים שכבר מולאו בדף להוספת ההזמנה).



בלחיצה על "הוספת סוג תסרוקת" יפתח טופס ובו יש למלא את הפרטים ולשמור (טופס זה מאפס את הפרטים שכבר מולאו בדף להוספת ההזמנה).בלחיצה על כפתור העידכון בכל שלב יפתח מסך עידכון ההזמנה מלא בפרטי -

הלקוח לשינוי ושמירת השינויים.

## 

## **רפלקציה**

פרויקט זה העניק לי רבות.

מהבחינה הטכנית כי מטעויות תמיד לומדים, נהניתי מהלמידה בצוותא ולחוד וכמובן גם ניסיתי לפתור תקלות בכוחות עצמי.

בנוסף לכך היה לי לאתגר גדול מאד יצירת הפרויקט כשתי בנות בודדות הנחשבות ל"מוחרגות" (שמשלימות את לימודי התכנות לבד ) ללא התחשבות יתירה מצד המטלות הלימודיות (הכוללות את תוכנית הלימוד של משרד החינוך ובנוסף את תוכנית הלימוד של אולטרה קוד (רמה ב')).

הבנתי שבתכנות צריך לנסות לבדוק להסיק מסקנות וגם דברים שלא למדנו באופן

פרונטלי ניתן להבין לבד ולבדוק לבד.

כמובן שהמורה ליוותה אותנו צעד בצעד ולמדה אותנו בסבלנות וביעילות אך "הזמן קצר והמלאכה מרובה...".

יש לציין שנהניתי גם מהכתיבה עצמה וההתקדמות שהסבה לי סיפוק רב.

תודה לבוראי שהוליכני והביאני עד הלום והחכמני בינה ודעה.

וכמובן לך בודקת יקרה.

בהצלחה!