

**RentingGown**

שמלה מושלמת במרחק נגיעה

חוה ברמן ותמר פרישמן | סמינר הרב וולף | אוקטובר 2018

## תודות

## כותרת 2

אם התמונה שבחרנו לעמוד השער אינה מתאימה לדוח שלך, תוכל להחליף אותה בקלות בתמונה משלך.

פשוט מחק את תמונת מציין המיקום. לאחר מכן, בכרטיסיה 'הוספה', לחץ על 'תמונה' כדי לבחור אחד מהקבצים שלך.

# **תוכן ענינים**

תודות---------------------------------------------------------------------------------------------- 2

תוכן ענינים----------------------------------------------------------------------------------------3

הצהרה---------------------------------------------------------------------------------------------4

הצעה לפרויקט גמר------------------------------------------------------------------------------

אישור הפרויקט-----------------------------------------------------------------------------------

גוף העבודה---------------------------------------------------------------------------------------

# **גוף העבודה**

כאשר אדם זקוק לשרותי השכרת שמלות הוא חפץ למצוא את ההתאמה הקרובה ביותר לחלומותיו שתענה במקסימליות על דרישותיו הטכניות, ויכולתיו, תוך דגש על נוחות גבוהה כבר בשלב האיתור, רבים מתייאשים כבר בשלב האיתור, בלי כל קשר לממצאים הקיימים בשוק.

נוסף על כך בכדי לשפר עבורו את נתוני האיתור וההתאמה, יהיה מן הראוי לנהל מאגר שיאסוף מידע מחיפושים ואיסופים קודמים.

ועל סמך ידיעות אלו יוכל הלקוח לבטח לבחור ולקבל את ההתאמה הטובה ביותר עבורו.

אך על מנת שאכן כל דרישותיו תתמלאנה עליו להשקיע זמן רב בבירורים מה –שלא כל אדם יוכל להרשות לעצמו, ומפאת חוסר אותו זמן יזדקק להתפשר ולוותר על חלק מדרישותיו והמה כמובן יבואו לידי ביטוי בהתאם, נוסף על כך, חרף הבירורים הרבים לא יוכל להגיע למירב התוצאות משום שחיפושיו הינם רק בקרב מכריו ותו לא.

אי לכך ובהתאם לזאת ראינו צורך לתכנת אתר שבזמן מינימלי מפנה לתוצאה המתאימה מקסימלית.

התוכנה תכיל ממשקים שונים עבור הרשאות שונות:

• ממשקים לאורח מזדמן האורח יכול לבצע חיפוש והמידע שיופק מחיפושו – יכלול את הנתונים שהזין. ויתאפשר לו לשכור פריט וכן יוכל להרשם למערכת. לא תנתן אופציה להעלות שמלות למאגר או לצפות בנתוני המשכיר, וכן לא לצפות במידע על נתח השוק) אלו יינתנו רק בתנאי שהמשתמש הינו – רשום במערכת(.

• ממשקים למשתמש הרשום במערכת המשתמש רשאי לבצע חיפוש, לצפות בנתוניו האישיים, להעלות שמלות למאגר, לבצע תשלום בנוסף יוכל המשתמש לשנות ולעדכן פרטים אישיים שלו במערכת.

• ממשקים למשתמש הרשום במערכת כבעל עסק למשתמש זה תנתן הרשאה לצפיה בפילוחי מידע וסטטיסטיקות, בנוסף לשאר הגישות הקיימות למשתמש רשום. (למעט העלאת שמלות למאגר).

• ממשקים למנהל מערכת המנהל יוכל לבטל השכרה, למחוק משתמשים, תהיינה לו הרשאות גישה לצפיה במאגר, לחיפוש, לצפיה בפילוחי מידע ואפשרות לשינוי המידע.

האתר מעניק למשתמש חיפוש שמלה על פי קריטריונים לבחירתו ומבחר קטגוריות לסינון,

השוכר-משתמש יכול ליצור עם המשכיר קשר טלפוני\ דרך מייל המשכיר.

יצירת אתר המכיל מידע למשתמשים ובעלי עסק בצורה מאורגנת, מקוטלגת, ממוינת ונוחה

לחיפוש.

האתר יעוצב בצורה נוחה וברורה לשימוש המאפשרת למשתמש איתור שמלות לפי מספר קריטריונים רב ומיון תוצאות החיפוש עפ"י מרחקם ממיקום המשתמש, הוספת שמלות למאגר.

# **1 . מדריך למתכנת:**

1.1 אסטרטגיות טכנולוגיות:

השתמשנו בשפות הבאות בכתיבת הפרויקט:

**C#**• **:** שפת C# היא שפת תכנות שפותחה ע"י מיקרוסופט ונחשבת לאחת

משפות התכנות הפופולריות בעולם. היא מיועדת לפיתוח כללי של מגוון

אפליקציות בכל התחומים מאתרי Web דרך משחקים, מאפליקציות למכשירי

מובייל וטאבלטים ועד לשירותי ענן. התחביר והעקרונות שלה הם פשוטים מצד

אחד אך עשירים ביכולות מצד שני.

**: Html (hypertext markdown lang)** •

זוהי השפה הטבעית ליצירת דפי אינטרנט ברשת. זו שפת תגיות פשוטה ,

אוניברסלית המאפשרת לעורכי אתרים ליצור דפים מורכבים שמכילים טקסט

ותמונות , שיכולים להראות בידי כל המשתמשים ברשת האינטרנט ללא תלות

בסוג המחשב או בסוג הדפדפן .

JavaScript: **JavaScrip**t • היא שפת תכנות של HTML והאינטרנט. מאפשרת

למפתחים להטביע קטעי סקריפט )"תוכניות קטנות"( בתוך מסמכי HTML . ובכך

להפוך את הדפים לדינמיים יותר. ניתנת לקריאה ע"י דפדפן.

CSS: **CSS (casding style sheets)** •היא שפת עיצוב המגדירה תבנית העיצוב של מסמכי HTML. CSS

למשל מטפלת בגופנים, צבעים, גבולות, שורות, גובה, רוחב, תמונות רקע,

מיקום מתקדם ודברים רבים נוספים.

JQuery: **jQuery** •היא תשתית פיתוח(ספריה) עבור JavaScript ההופכת פעולות

נפוצות ב JavaScript לקלות הרבה יותר, עוזרת לשפר את חווית המשתמשים

באתרים ומטפלת בהבדלים באופן המימוש של JavaScript בין דפדפנים שונים.

כתבנו את הפרויקט בטכנולוגיית **Asp.net MVC .**

**Asp.net MVC** היא הטכנולוגיה המתקדמת ביותר מבית מיקרוסופט לפיתוח אפליקציות

WEB. הטכנולוגיה שמה דגש על פיתוח מהיר, שימוש בארכיטקטורה נכונה והפרדת

שכבות, שילוב ספריות קוד פתוח ותמיכה מובנית בבדיקות אוטומטיות.

**Asp.net MVC** הינה תשתית אשר משמשת לבניית יישומי אינטרנט, ע"י החלת

העקרונות של תבנית מודל –תצוגה- בקר( Design Pattern Model-View-Controller) על

תשתית ASP.NET .

תבנית MVC מהווה כבר שנים רבות תבנית ארכיטקטית חשובה בתחום מדעי

המחשב. התבנית מייצגת דרך יעילה להפרדה בתוך היישום )למשל הפרדת רכיב

הגישה לנתונים מרכיב הטיפול בתצוגה( והיא יעילה לעיצוב יישומי אינטרנט.

ההפרדה המוחלטת של המרכיבים מוסיפה מידת מה של מורכבות ליישום, אולם

התועלת הרבה שמתקבלת בתמורה מצדיקה את המאמץ.

תבנית MVC מפרידה את היישום לשלושה היבטים עיקריים:

11

Model .1 קבוצה של מחלקות המתארות את הנתונים עליהם פועלים. המודלים -

יהוו ברוב המקרים מעין שכבת גישה לנתונים, תוך שימוש בכלים כגון Entity Framework .

View .2 הגדרת אופן ההצגה של ממשק המשתמש. -

. Controller .3 השכבה האחראית על היחסים בין ה Model לבין ה View. מגיב

לקלט מהמשתמש, - מדבר עם המודל, וקובע איזו תצוגה צריך לממש.

השתמשנו בסביבות העבודה הבאות:

✓ :Visual Studio 2017

סביבת הפיתוח המרכזית בעולם המייקרוסופטי נקראת Visual Studio . סביבה זאת

מכילה מאות אפשרויות . ותומכת במספר שפות תכנות .בנוסף, יש לציין כי רוב שפות

התכנות והטכנולוגיות הנתמכות על ידי סביבת פיתוח זו הן חלק מתשתית הפיתוח של

חברת מייקרוסופט ..NET Framework תשתית זו מכילה מספר עצום של ספריות קוד

וטכנולוגיות המקלות עלינו בתהליך יצירת התוכנה.

✓ (localDB)SQL Server 2014 :

Microsoft SQL Server היא פלטפורמת שרת מידע ומבנה נתונים המיועדת למפתחים,

ארגונים קטנים וגדולים כאחד אשר מציעה חבילה טכנולוגית שלמה של כלים

ארגוניים המאפשרים להשיג את הערך המרבי מתוך המידע שלנו במחיר ובעלות

הכוללת הנמוכים ביותר. תוכנת ניהול השרת מיועדת למשתמשים הזקוקים לפתרונות

ניהול מידע באמצעות בסיסי נתונים בצורה יעילה ומאובטחת. בנוסף ניתן לפתח

ולנהל תוכנות למחשבים שולחנים ותוכנות מבוססות רשת העושות שימוש בבסיס

נתונים המנוהל תחת שרת SQL Server במהלך העבודה עימן.

**1.2 תיאור מבנה הפרויקט:**

ה-Solution שלנו מכיל את הפרויקט העיקרי ממנו מובנה האתר : **RentingGown**.

הפרויקט מחולק לשלוש שכבות עיקריות:

**Models:**

קבוצה של מחלקות הכתובות בשפת C# המתארות את הנתונים עליהם מתבסס המידע באתר.

כל מחלקה כוללת מאפיינים לפי הצורך השימוש בפרויקט על סמך סנכרון הנתונים מתוך ה-Database

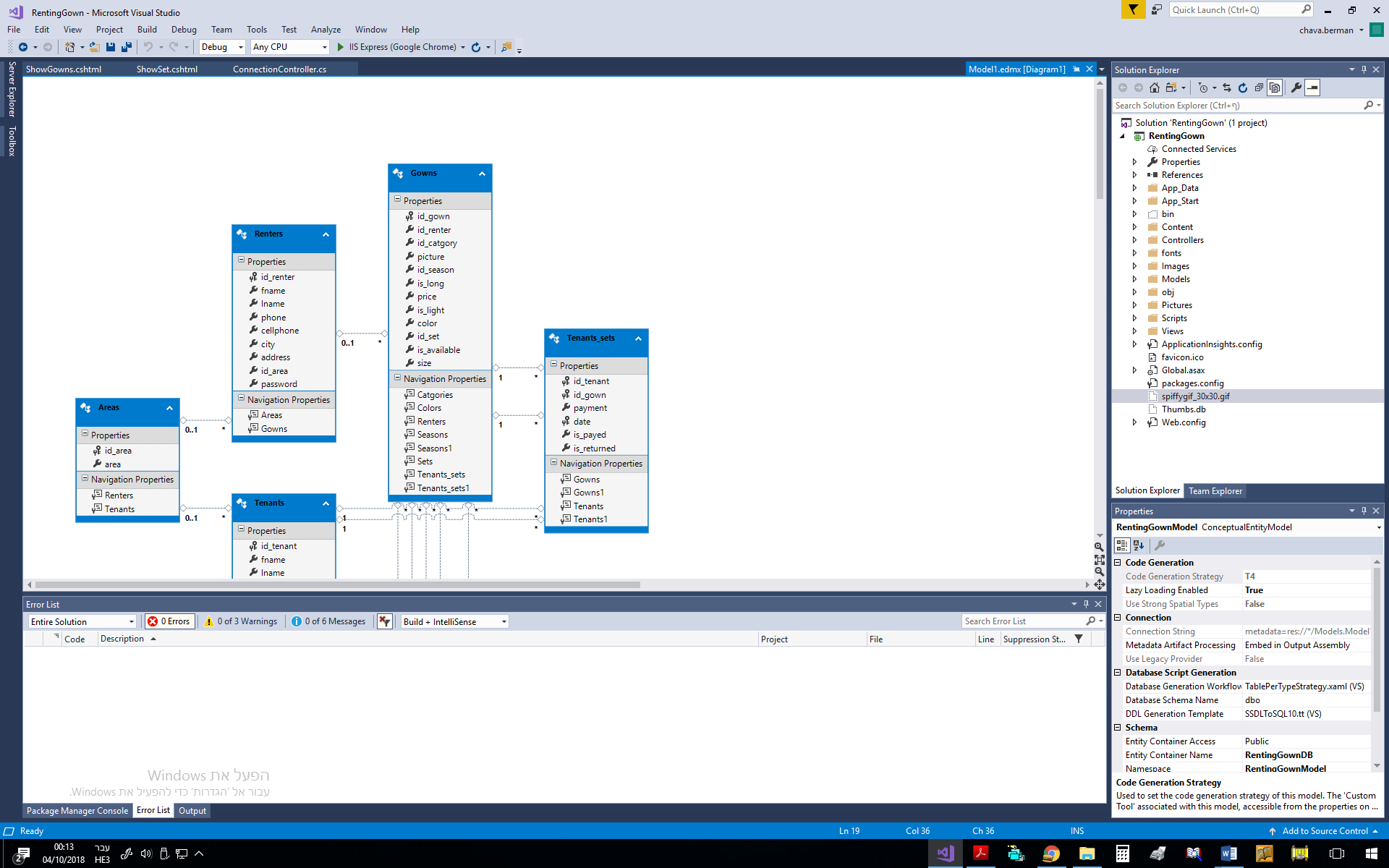
לדוגמא: מחלקת Sets המתארת נתונים עבור סטים של שמלות ומכילה את המאפיינים הבאים: מזהה סט,

כמות מהסט, תמונה.

**Views:**

הגדרת אופן ההצגה של ממשק המשתמש. כל View מציין דף Html אחר בדפדפן, וב-View מגדירים את תוכן וצורת התצוגה.

**Controllers:**

השכבה האחראית על היחסים בין ה-Model לבין ה- View מגיב לקלט מן המשתמש, "מדבר" עם ה- Model, וקובע איזו תצוגה צריך לממש. ה-Controller מקבל נתונים מה-View , עושה חישובים שונים עם שימוש ב-Database וב- Model, ומחזיר נתונים לתצוגת ה- View הרצוי.

**1.3 עקרונות התכנון/הבניה/הניתוח:**

1.3.1 עקרונות תיאורטיים:

* **הורשה:**

בהורשה משתמשים בתכנות מונחה-עצמים, ירושה היא דרך לבסס יחס "סוג של" (is-a ) בין עצמים. המימוש הנפוץ של הגדרה זו הוא בעזרת מחלקות. מחלקה יכולה לקבל בירושה תכונות והתנהגות של מחלקת-האם שלה (נקראת גם "מחלקת-על", "מחלקת אב" או "מחלקת בסיס"). היחסים בין מחלקות מגדירים היררכיה של מחלקות. מנגנון הירושה מסייע במידול של תחומים בדרך דומה לזאת המוגדרת באופן "טבעי" ע"י העוסקים בהם.

לדוגמה: אם ריבוע הוא סוג-של (או מקרה פרטי של) מצולע, ניתן להגדיר יחס ירושה בין המחלקה מרובע למחלקה מצולע; על פי רוב, אובייקט של המחלקה היורשת (מרובע) יתמוך בכל הפעולות המוגדרות על מצולע כלשהו, ובנוסף יתמוך בפעולות נוספות, המיוחדות למרובע. בדומה, ניתן

להגדיר יחס ירושה בין המחלקה ריבוע למחקה מרובע.

*שימושים:*

ירושה מאפשרת להשיג מספר מטרות:

1. שימוש חוזר בקוד. בעזרת ירושה ניתן לכתוב מחלקות בעלות טווח התנהגות נרחב, תוך כתיבת קטעי קוד קטנים יחסית שמרחיבים מחלקה קיימת.
2. מידול של העולם האמיתי באופן דומה לזה המוגדר על ידי האדם. למשל, ניתן לכתוב היררכיה של מחלקות המקבילה לטקסונומיה המקובלת בביולוגיה (יען היא סוג של ציפור, האדם הוא סוג של יונק, וכן הלאה).
3. פולימורפיזם (רב צורתיות) של זמן-ריצה. משתנה מטיפוס (סטטי) מסוים יכול להתנהג בצורות שונות, לפי הטיפוס של האובייקט שהוא מתייחס אליו בזמן ריצה. למשל משתנה מהמחלקה בעל חיים שנדרש לבצע פעולה מסוג "השמע קול" יבצע פעולות שונות לחלוטין במקרה שהאובייקט הוא מהמחלקה כלב או חתול. ניתן לבצע זאת גם ללא ירושה (בעזרת פקודות תנאי), אך ירושה מאפשרת מימוש מודולרי, גמיש וקל להרחבה.

בפרויקט השתמשנו בהורשה במקרים רבים:

* מחלקה ה-RentingGownDB

יורשת ממחלקת ה DbContext המקושרת ל-System.Data.Entity

ממחלקה זו נוצר מאגר הנתונים ע"י שיטת –Dbfirst עם חיבור ל-entity framework

public partial class RentingGownDB : DbContext

{

public RentingGownDB()

: base("name=RentingGownDB")

{

}

* מחלקות ה-views:

יורשות ממחלקות ה- models

לדוגמא: ה-view של ShowGowns יורש ממחלקת ה-Gowns

ז

@model RentingGown.Models.Gowns

@{

ViewBag.Title = "showGown";

}

<!-- Styles -->

<link href="~/Content/bootstrap.min.css" rel="stylesheet" />

<!-- Bootstrap -->

<link href="~/Content/line-awesome.min.css" rel="stylesheet" />

<!-- Icons -->

<link href="~/Content/style.css" rel="stylesheet" />

<!-- Style -->

<link href="~/Content/responsive.css" rel="stylesheet" />

<!-- Responsive -->

<link href="~/Content/colors/colors.css" rel="stylesheet" />

<!-- color -->

<section>

<div class="block gray ">

<div class="container">

<div class="row">

<div class="col-md-6 col-md-offset-3">

<div class="account-popup-sec static-popup">

<div class="acount-popup register-popup">

<h3>פרטי השמלה</h3>

<form dir="rtl">

<div class="field-form">

<img src="~/Images/@Model.picture" />

</div>

<div class="field-form">

<p><b> מחיר: </b>@Model.price</p>

</div>

<div class="field-form">

<p><b>מידה:</b> @Model.size</p>

</div>

<div class="field-form">

<p><b>עונה:</b> @ViewBag.season</p>

</div>

<div class="field-form">

<p><b>קטגוריה:</b> @ViewBag.catgory</p>

</div>

<div class="side-search-form">

<div class="field">

<a href="/Gowns/AddToTheBasket?id=@Model.id\_gown" class="filter-btn">הוסף לסל הבגדים</a>

* **הפרדת שכבות:**

כל תוכנה/אתר שנפגוש בעולם מבוססת על ארכיטקטורת שכבות הנקראת בשם Three-Tier-Application.

ז"א שלש שכבות במבנה שך DAL-BI-UI.

זוהי תבנית עיצוב בסיסית שמגדירה הפרדת האפליקציה ל:שכבת נתונים, שכבת לוגיקה ושכבת ממשק משתמש.

לתבנית עיצוב זו יתרונות רבים:

1. *תחזוקה:*

ניתן להחליף או לתקן מימוש פנימי של שכבה אחת בארכיטקטורה בלי לשנות שכבה אחרת.

1. *נוחות פיתוח:*

אדם אחד עובד על רכיב בתוכנה, אדם אחר עובד על רכיב אחר, כל עוד שהחתימות זהוץ ניתן לשלב כוחות ולייעל נתוני פיתוח.

1. *בדיקות:*

תקלה כלשהיא מבודדת בכל שכבה בנפרד, לדוגמא אם לא קיבלנו רשימת נתונים לתצוגה נבדוק קודם את שלב הביניים (שכבת הBL ) אם הנתונים שם תקינים נדע בוודאות שגם ברמת שכבת הDAL הנתונים תקינים וכל שנותר הוא לפתור את התקלה ברמת הUI

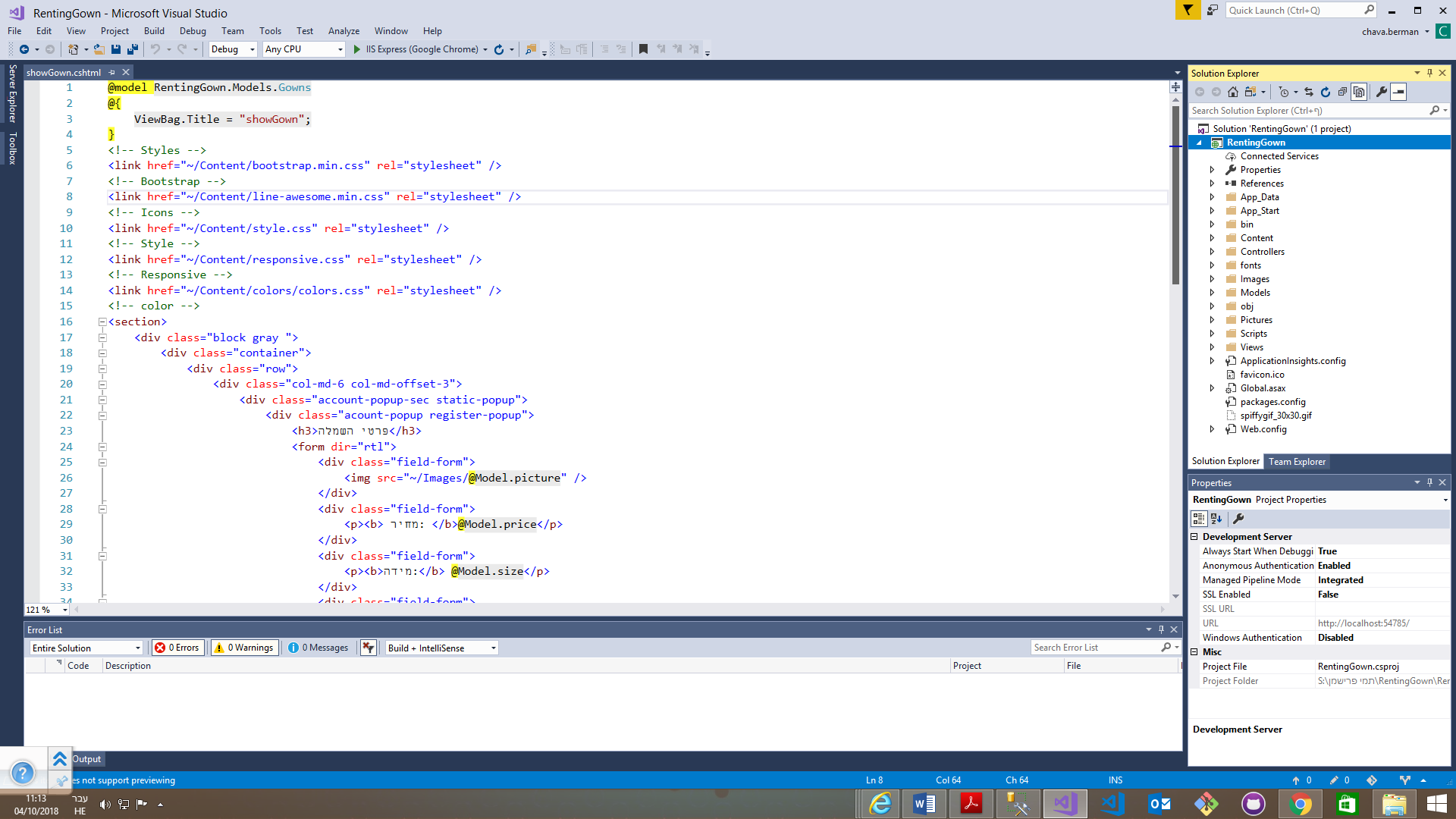
1. *שימוש חוזר:*

נניח שהרכיבים BL + DAL אהובים עלינו ועובדים היטב, ניתן להחליף את פלטפורמת ה UI לטכנולוגיה אחרת במינימום מאמץ.

1. *אבטחה:*

נוח יותר להגדיר Interface-API לשכבה מסוימת בלי לחשוף מבני נתונים, logic או DATA שאינם רלוונטיים למשתמש, כמו גם חסימה בפני האקרים (רלוונטי יותר בטכנולוגיות WEB ) ברמות שונות.

הפרויק נכתב בשפת C# ומיושם בטכנולוגית mvc. טכנולוגיה זו בנויה על שלושת השכבות:



MVC

שלושת השכבות

* Dal-models
* BI-controllers
* Ui-views

1.3.2 עקרונות תכנותיים:

* **Session:**

שמירת מידע בצד שרת-

הוא משאב אישי ללקוח, המידע אינו משותף ואינו נגיש ללקוחות אחרים. כשאר הלקוח מבקר באתר, השרת מקצה לו אובייקט Session ושומר אותו בזכרון. Session האובייקט נשמר בזכרון כל עוד המשתמש נשאר באתר. אם במשך זמן (מוגדר מראש) האתר ננטש, אובייקט הSession ימחק, הוא ימחק גם כאשר סוגרים את הדפדפן או כאשר נותנים הוראה למחיקתו.

יישום:

לאחר איתור השמלות הנבחרות, מאפשר למשתמש להוסיף את השמלות שיאהב לסל הבגדים., סל זה הוא כמובן אישי לכל גולש ואין אפשרות לשיתוף בחירות בין הגולשים, כמו"כ הסל ישמר גם במעבר בין הדפים.

Gowns gown = db.Gowns.FirstOrDefault(p => p.id\_gown == id);

if (Session["listOfGowns"] == null)

{

Session["listOfGowns"] = new List<Gowns>();

(Session["listOfGowns"] as List<Gowns>).Add(gown);

}

else

{

var q = (Session["listOfGowns"] as List<Gowns>).FirstOrDefault(p => p.id\_gown == gown.id\_gown);

if (q == null)

(Session["listOfGowns"] as List<Gowns>).Add(gown);

}

* **Localstorage:**

שמירת מידע בצד לקוח-

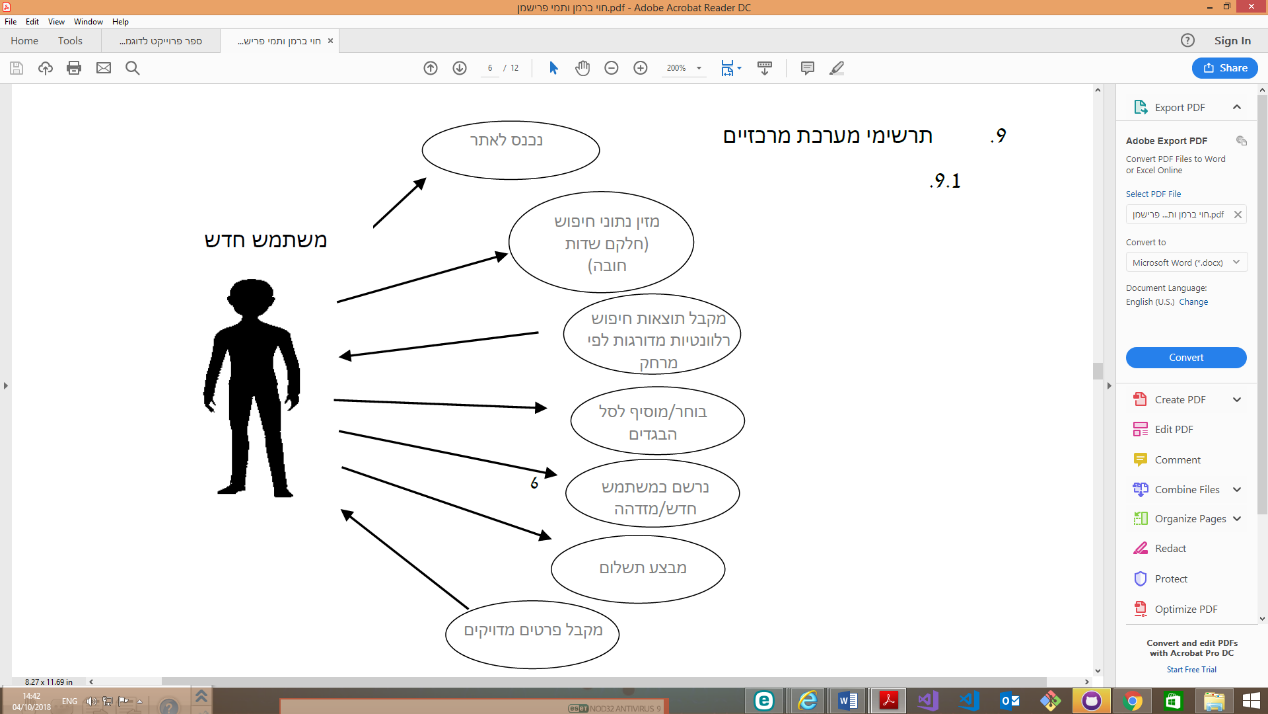
נתונים שנשמרים באמצעות ה- localStorage לא נמחקים עד שנמחוק אותם שהמשתמש ימחוק אותם ידנית, כלומר, גם כאשר הוא יכבה וידליק את המחשב הנתונים האלה ישמרו לנו.

מידע זה נשמר לאורך זמן רב (אפילו שנים).

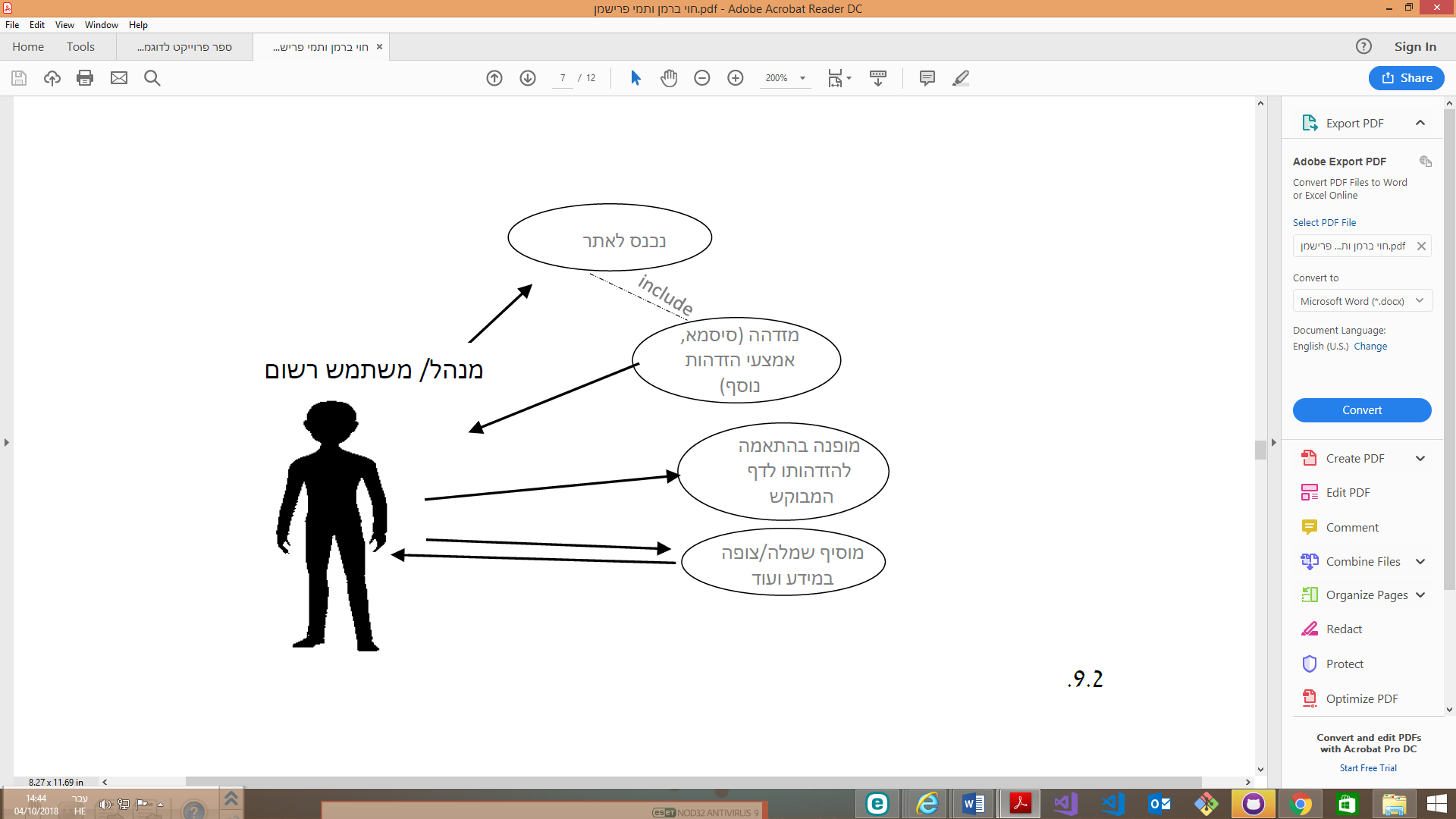
יישום:

בכדי לדרג את השמלות עפ"י מרחק ממיקום הגולש, שמרנו את המידע החוזר מmaps api ב-localstorage עם מספור עוקב, בכדי שנוכל למיין תוצאות.

הנתונים נמחקים מהאחסון בגמר תפקוד הקוד.

**1.4 תרשימים:**

1.4.1תרשימי uml:



1.4.2עץ תהליכים:

המשתמש נכנס לאתר

מנהל מזדהה עם סיסמה

המשתמש יכול לערוך חיפוש שמלה עפ"י קריטריונים וקטגוריות.

המנהל מקבל גישה לצפיה ושינוי המאגר, המשתמשים ועוד.

משתמש נכנס ומזדהה עם סיסמה

בעל עסק נכנס ומזדהה עם סיסמה

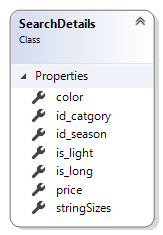
בעל עסק יכול לצפות במידע וניתוחי סטטסיטקות

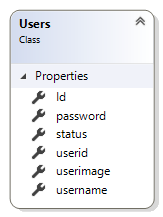
המשתמש יכול להוסיף את בחירתו,לבצע תשלום ולקבל דרכים ליצירת קשר עם המשכיר.

משתמש יכול לערוך שינויים בפרופיל שלו.

משתמש יכול להוסיף שמלה למאגר.

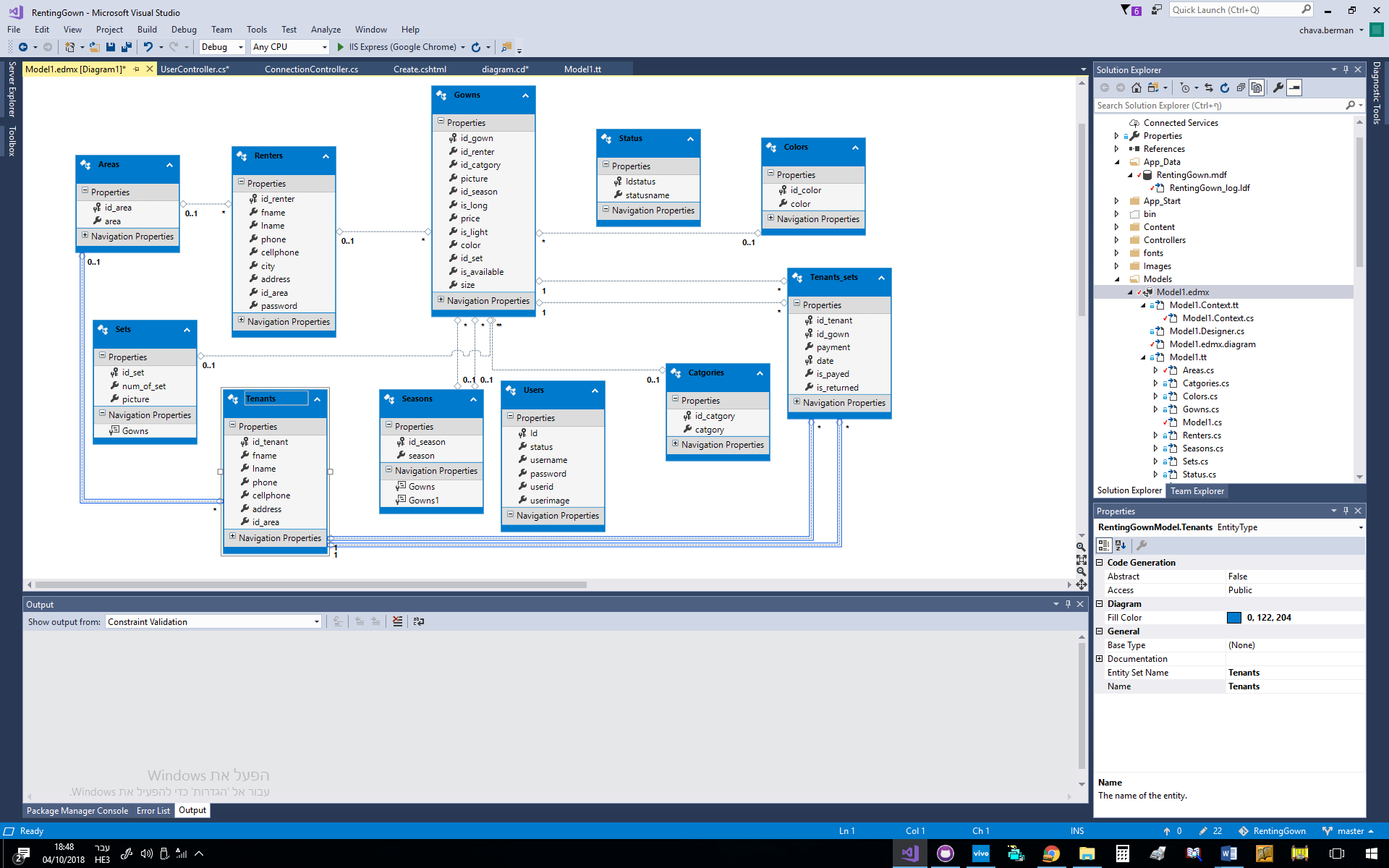
משתמש יכול לערוך שינויים בפרופיל שלו.

1.4.3 תרשים מראה המחלקות:



1.5 מבנה נתונים מאוכסנים:

1.5.1 תרשים SQL:



1.5.2 מבנה קבצים ותיקיות:

1