בע"ה

קורס במיני פרויקט במערכות חלונות (דוט נט)
הנחיות למיני פרויקט –
מערכת לאיתור והתאמת נופש ממוקד לקוח
ה'תש"פ

תוכן עניינים

מבוא

היעד: כתיבת תוכנה (=פרויקט) מתפתחת במהלך הקורס.

המטרה: תרגול ויישום של:

- עקרונות שפת #C,
- עקרונות ארכיטקטורה של תוכנה,
- יצירת ממשקי משתמש באמצעות תשתית הפתוח המודרנית WPF
- שימוש בשירותי רשת.
- חשיבה תכנותית ולמידה עצמית.

הערה: ההכוונה במסמך זה היא כללית בלבד, שכן חלק מן הדרישה היא להפעיל חשיבה יצירתית. כמו כן, חלק מהמבחן הסופי יתבסס על נושאים תכנותיים שתתבקשו להתמודד איתם במהלך פיתוח הפרויקט. חשוב להדגיש שציון מתחת ל-85 בקורס זה כמו בכל קורס מעשי אחר איננו משמעותי בעיניהם של המעסיקים בתעשייה...

דגשים:

- העבודה תתבצע אך ורק בזוגות (במקרים מיוחדים, ניתן לקבל אישור מהמרצה לעבודה עצמית, אך ודאי לא בשלישיות). הזוגות יהיו קבועים לכל אורך הסמסטר (לא ניתן לעבור במהלך הסמסטר מקבוצה לקבוצה. לא ניתן להיות שותף עם חבר שרשום בקבוצה שונה. בכל אחד מן המקרים האלה התרגיל ייפסל).
- כל אחד מהשותפים חייב להיות שותף מלא בכל אחד מן השלבים. (במידה ויתברר כי שותף אחד עשה שלב מסוים ואלו שותף אחר עשה את החלק האחר, הציון יהיה פרופורציונלי למספר השלבים שנעשו ע"י כל אחד מן השותפים)
- בסוף הסמסטר, ייבחר פרויקט מצטיין מכל קבוצת תרגול. מבין אלו, ייבחרו הפרויקטים המצטיינים, והם יזכו את יוצריהם בפרס. **המדד העיקרי למצוינות הוא מקוריות, למידה עצמית, וכמובן תכנות נכון**.

תהליך ההגשה:

- הפרויקט מחולק לשלושה שלבים המבוססים זה על זה.
- בסיום כל אחד מן השלבים נקבע מועד הגשה והצגה של הנעשה עד כה.
- בדיקת שני השלבים הראשונים תעשה בצורה כללית (בעיקר תיבדק הרצה תקינה, הערות על איכות העבודה, ועמידה מדויקת בלוח הזמנים הנדרש). הציון על שני שלבים אלו לא יועבר לידי הסטודנטים.
- עם הגשת השלב האחרון והמסכם תתבצע בדיקה יסודית של העבודה שתכלול גם הגנה (מעיין בחינה בע"פ ע"י בוחן חיצוני) של שני השותפים על העבודה, כ"א בנפרד.
  הציון הסופי על הפרויקט יתייחס לכל רכיביו, וכן - לאיכות ההצגה, ולעמידה בזמני ההגשה.
- ההגנה על השלב האחרון תתבצע בשיעור האחרון של הסמסטר ולא יאוחר מכך.

- ציון המעבדה יהיה 50% מהציון הסופי (מורכב מ10% תרגילי בית ו40% פרוייקט) והוא ישוקלל עם ציון המבחן רק בתנאי שהציון במבחן הוא לפחות 55.
- למרות שהפרויקט הוא המטלה הדורשת את עיקר זמנכם בקורס, בסופו של דבר המודד העיקרי לציון הסופי הן שאלות המבחן שנגזרות מדיוק והבנת הפרטים של כל הנעשה. וזאת בדומה לשאלות שנשאלות בראיון עבודה. לכן לאורך כל הדרך – חשוב להבין את הפרטים הקטנים, ולא להסתכל רק על תמונת הפרויקט הגדולה.

הנחיות להגשת הקבצים במערכת המודל:
בכל אחד מן השלבים, יש להעלות, **עוד לפני ההגנה**, קובץ zip עם הפתרון למודל (לפי ההנחיות בתרגיל המבוא):
שם הקובץ יהיה Project01_xxxx_yyyy_dotNet5780
dotNet5780 = שם הקורס והשנה העברית
01 = שם השלב (01,02,03)
xxxx = 4 ספרות אחרונות בתעודת הזהות של בן הזוג הראשון
yyyy = 4 ספרות אחרונות בתעודת הזהות של בן הזוג השני
נא להקפיד על פורמט זה על מנת למנוע מצב של אי קבלת ציון על שלב מסוים.

הערה:
מומלץ ביותר לעבוד עם אחת מתוכנות ה GIT לניהול גרסאות ושיתוף פעולה בין שני בני הזוג.

הערה:
יש לוודא שתכני הפרויקט (כולל תמונות וכדומה) יתאימו לרוח ההלכה – גם לפי הדעות המחמירות ביותר.

תיאור כללי

בפרויקט זה נכתוב מערכת (חלקית בלבד) להתאמה של אירוח ונופש ללקוחות.

מערכת איתור הנופש הינה רעיון חדש ומהפכני, שמטרתו לפשט ולהקל על המעוניינים בנופש לקבל מענה מתאים, כך ש:

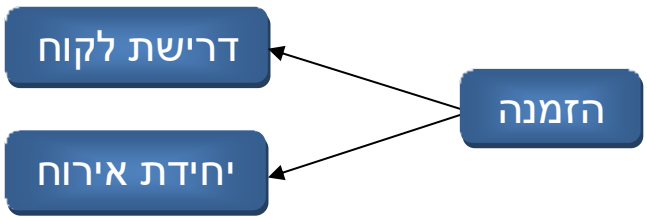
- הלקוחות ימנעו ממאמצי בירור וחיפוש מיותרים, וחלף זאת – לאחר ציון חד פעמי של דרישותיהם, יקבלו ישירות הצעות המותאמות לדרישותיהם.
- וגם מהמארחים (מנהלי יחידות האירוח) יחסך מענה לטלפונים לא רלוונטיים.

המיזם החדשני מנוהל ע"י מערכת אינטרנטית ומכיל מספר ישויות. אנחנו נדמה את המערכת באופן חלקי. אמנם, את הפרויקט הזה עדיין לא נכתוב כאתר אינטרנטי בתבנית של שרת/לקוח (מפני שזה חורג מנושאי הקורס) אך נכתוב את רוב השכבות בצורה כזאת שהרוב יהיה חופף למערכת אינטרנטית מקבילה.

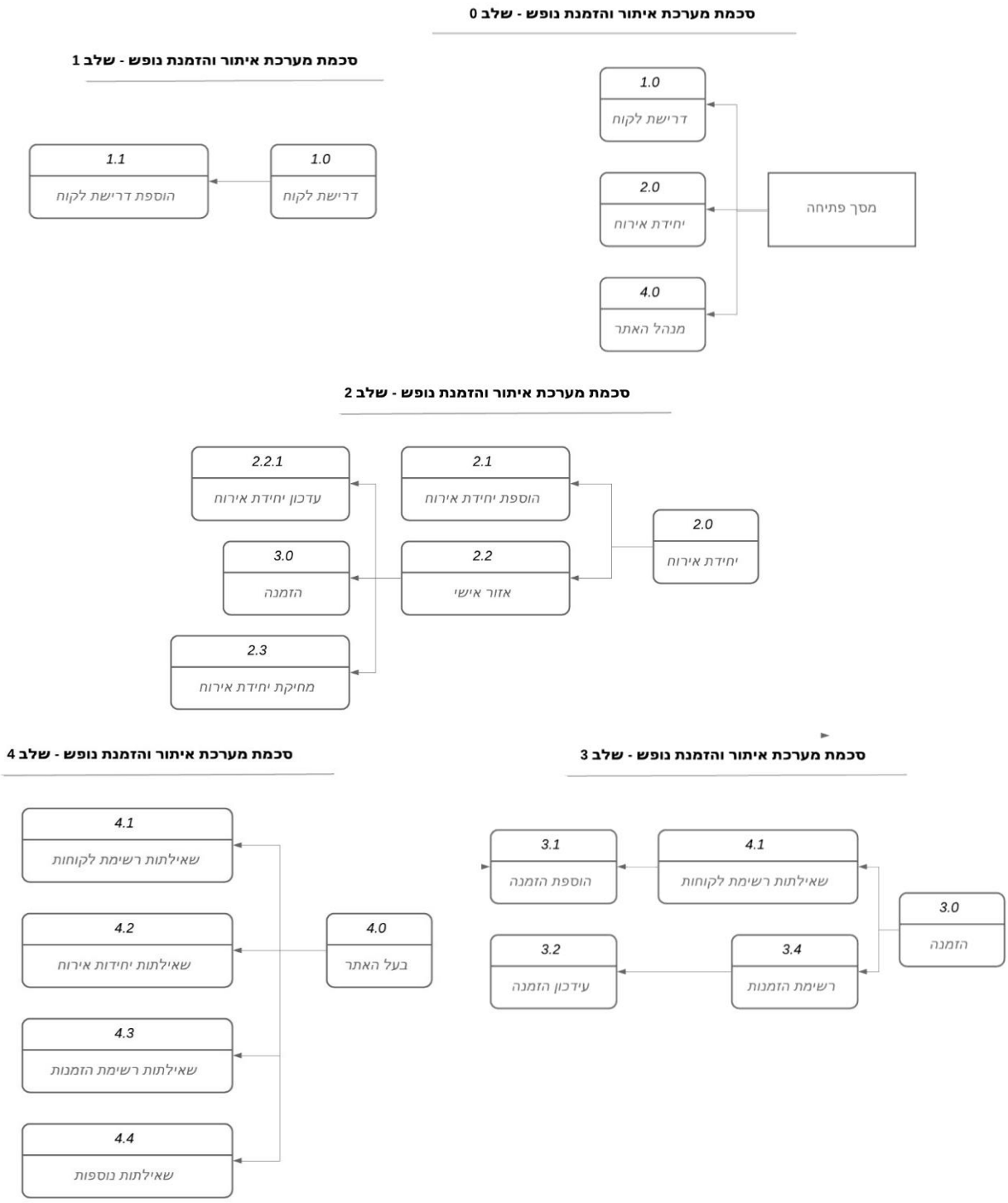
כל לקוח שמעוניין בנופש, יכול לרשום את עצמו לאתר, תוך ציון פרטי הנופש הרצויים עבורו (כגון זמני הנופש, אזור בארץ, האם מעוניין בצימר/מלון/דירת נופש, האם נדרש ג'קוזי ביחידה).
הלקוח אינו נדרש לפניות נוספות למערכת.
המארחים (=מנהלי יחידות האירוח) יוכלו לסרוק ולסנן את בקשות הלקוחות לפי קריטריונים שונים, וליצור הזמנה, המבוצעת ע"י משלוח מייל עם הצעת אירוח ללקוחות שיש התאמה בין דרישותיהם למה שהם יכולים להציע (כגון: זמני אירוח פנויים, בריכה ועוד). הערה: נניח שהמארחים ישרים ויסננו את ההצעות באופן הגון. למעוניינים – ניתן לדרוש שהתכנה תבדוק את ההתאמה – ולהוסיף שדות בהתאם. מאידך - כתובת המייל של הלקוח (אמצעי הקשר היחיד איתו) אינה גלויה למארח, והדרך ליצור קשר ראשוני עם לקוחות תהיה בלעדית דרך המערכת.

במידה והלקוח מעוניין באירוח שהוצע לו, הוא יאשר את ההזמנה מול המארח (בתכתובת מייל ביניהם מחוץ למערכת), והמארח הוא זה שידאג לשנות בהתאם את סטטוס ההזמנה, שתהפוך את פרטי הלקוח לבלתי זמינים (כך שלקוח לא יקבל הצעות ממארחים נוספים, וכל ההזמנות האחרות שנשלחו אליו יסגרו).
בעל האתר יוכל לראות את מצב ההזמנות, ועבור כל הזמנה שנסגרת בהצלחה ייגבה באופן אוטומטי תשלום (= מספר ימי הנופש X עמלה קבועה) מחשבון הבנק של המארח.

תרשים המתאר את הישויות במערכת והקשרים ביניהן:



להלן תיאור סכמתי של תפריטי / מסכי המערכת.



(תיאור המסכים הנ"ל מהווה הצעה בלבד שנועדה להסביר כיצד המערכת פועלת. ניתן לשנות את המעברים בין המסכים וכן את תוכנם – כל עוד שומרים על היגיון ורמת ביצוע)

תיאור ישויות המערכת

דרישת לקוח
ישות המכילה פרטים מזהים עבור הלקוח (כגון: שם משפחה, שם פרטי, כתובת מייל), מספר מזהה ייחודי (זהו המפתח), וכן את דרישות האירוח של אותו לקוח עבור אירוח ספציפי (למשל: תאריך תחילת הנופש, תאריך סיום החופש, אזור רצוי בארץ, סוג יחידת האירוח הרצוי, מספר מבוגרים, מספר ילדים, האם מעוניין ב:בריכה, ג'קוזי, גינה, מרפסת, אטרקציות לילדים), סטטוס דרישת הלקוח (פתוחה, נסגרה עסקה דרך האתר, נסגרה כי פג תוקפה).

יחידת אירוח
ישות המכילה מספר מזהה ליחידה, וכן פרטי המארח (כגון: מספר מארח, שם משפחה, שם פרטי, מספר טלפון נייד, כתובת מייל) כולל פרטי חשבון בנק והרשאה לחיוב חשבון – שיבואו לידי שימוש במקרה שנסגר אירוח ויש לשלם עמלה לבעל האתר.
כמו כן, הישות מכילה מטריצה של תאריכי הזמנות שנתית (12 חודשים כפול 31 ימים) – במטריצה זו יסומן האם היחידה תפוסה או פנויה עבור כל תאריך. המטריצה מעודכנת בכל עת לחודש קודם לאותה עת ול-11 חודשים קדימה (כך שניתן לקבוע אירוח עד 11 חודשים מראש).
הערה: במציאות אמתית, היה נכון להפריד את המארח ויחידת האירוח לשתי ישויות נפרדות. על מנת להקל על הדרישות, שתי הישויות חוברו, ואינכם נדרשים לטפל בלוגיקה למניעת תקלות כתוצאה מחיבור זה (כגון: מספר מארח זהה עם שם מארח שונה בשתי יחידות אירוח שונות)

הזמנה
ישות המקשרת בין בקשת הלקוח ליחידת אירוח של מארח ומכילה את הפרטים הבאים: מספר מזהה ייחודי להזמנה, מספר מזהה של בקשת הלקוח, מספר מזהה של יחידת האירוח, תאריך יצירת ההזמנה, תאריך משלוח המייל ללקוח, סטטוס ההזמנה (טרם טופל, נשלח מייל, נסגר מחוסר הענות של הלקוח, נסגר בהיענות של הלקוח).

בעל האתר
למעשה זו אינה ישות בפני עצמה – אך ניתן להוסיף עבורה מספר זיהוי / סיסמה

תיאור חלקי הפרויקט בקצרה

<u>שלב ראשון:</u>
נבנה את המערכת על פי מודל השכבות.
מקור הנתונים יופיע כפרויקט נפרד ויכיל רשימות (List<T>) של אובייקטים.
הגישה למקור הנתונים תתבצע באמצעות שכבת ה- DAL.
שכבת ה BL תכיל את הלוגיקה הנדרשת,
ושכבת ה UI תהיה ממשק console פשוט.

<u>שלב שני:</u>
החלפת שכבת ה UI.
פיתוח ממשק המשתמש לממשק ויזואלי גרפי.
תשתית הפתוח הנדרשת של הממשק הוויזואלי הינה WPF.

<u>שלב שלישי:</u>
החלפת שכבת ה- DAL.
מקור הנתונים יתבסס על קבצי xml. חלקם קבצים מרשת האינטרנט.
כמו כן – יתבצע שדרוג שכבת הBL: תוספת של שאילתות רשת (שליחת מיילים, חיפוש פרטי בנק), תוספת של תהליכונים (סגירת הזמנות שפג תוקפן, שליחת שאילתת רשת חוזרת).

הערה:
כל תקלה כתוצאה מפעולה כלשהי, תגרור אחריה טיפול בחריגות לפי מנגנון החריגות הקיים ב-#C.
יש לטפל בחריגות עבור כל שכבה בנפרד, וכן לטפל בהעברת חריגות במקרה הצורך משכבה לשכבה.
עיין בנספח 5 – נהלים לטיפול בחריגות

שלב ראשון – הגדרת הפרויקטים עבור מודל השכבות ומימושן

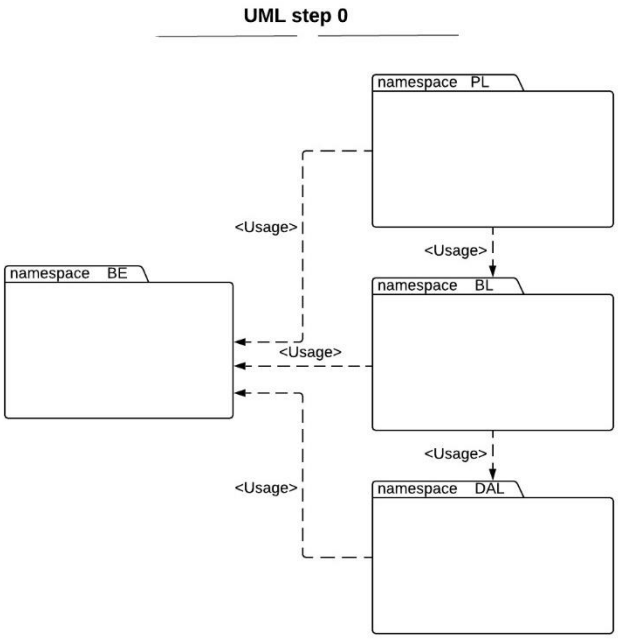
בחלק זה של הפרויקט נגדיר את כל אחת מן השכבות שיהוו יחד את המיני פרויקט שיאפיין את מערכת איתור והתאמת הנופש שלנו.

שלב זה כולל 4 חלקים כדלהלן:

חלק א' - הגדרת המחלקות עבור הישויות, כלומר שכבת ה- BE.
חלק ב' - הגדרת דרך העבודה מול מקור הנתונים, כלומר שכבת ה- DAL.
חלק ג' – הגדרת הלוגיקה של המערכת, כלומר שכבת ה- BL
חלק ד' – הגדרת ממשק המשתמש של המיני פרויקט, כלומר שכבת ה- UI.

פעולה מקדימה:

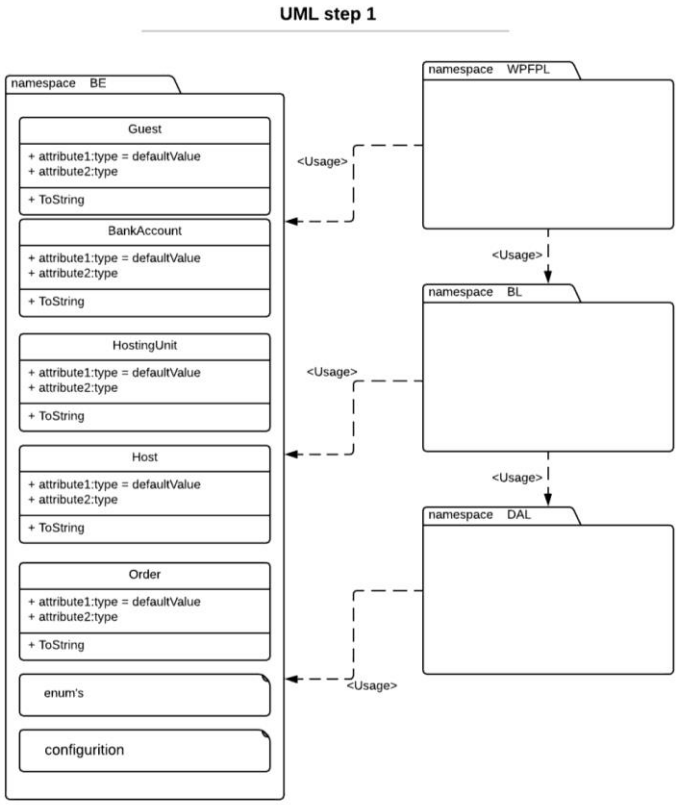
- ב visual studio בנה solution ריק,
- על פי מודל השכבות הוסף ארבעה projects ריקים:
  - BE - מסוג Class Library.
  - DAL - מסוג Class Library, וצור אצלו reference ל BE.
  - BL - מסוג Class Library, וצור אצלו reference ל BE ול- DAL.
  - PL - מסוג Console Appliction, וצור אצלו reference ל BE ול- BL.



חלק א – ישויות – שכבת ה BE

להלן נפרט שדות שיכללו במחלקות. **חובה** להוסיף שדות ופעולות נוספים.
שים לב! יש להימנע משימוש בפונקציות public במחלקות.
בכלל זה, משמע - אין לכתוב בנאים כלל (אלא להשתמש בבנאי ברירת מחדל).
(ניתן לכתוב ToString ציבורי שכן הוא משמש לצורך debug)
הלוגיקה המרכזית מתבצעת בשכבת ה- BL כפי שיפורט בהמשך.

הוסף את הישויות לפרויקט **מסוג Class Library** בשם **BE** . להלן תרשים המתאר את הממשק, והנחיות רשומות בהמשך. חובה להגדיר כל מחלקה בקובץ נפרד.



בקובץ נפרד יש להגדיר את כל הטיפוסים הסודרים (enum) הנדרשים (כגון: סוג אתר הנופש (צימר, דירת אירוח, חדר במלון, מאהל), אזורים בארץ (צפון, דרום, מרכז, ירושלים), סטטוס ההזמנה (טרם טופל, נשלח מייל, נסגר מחוסר הענות של הלקוח, נסגר בהיענות של הלקוח). סטטוס דרישת הלקוח (פתוחה, נסגרה עסקה דרך האתר, נסגרה כי פג תוקפה) וכדומה)

בקובץ נפרד נוסף יש להגדיר **מחלקה בשם Configuration** שתכלול את כל המשתנים הגלובליים כשדות סטטיים (ראו בהמשך: מספר רץ לכל מחלקה, סכום העמלה מכל עסקה, מספר ימים לפגות תוקף של הזמנה וכדומה)

מחלקה בשם GuestRequest שמייצגת דרישת אירוח של לקוח ותכלול:

מאפיין	דוגמא / אפשרויות	
מספר הבקשה לאירוח – מזהה ייחודי **(קוד רץ ייחודי בן 8 ספרות)**	10000001	**GuestRequestKey**
שם פרטי של הלקוח	אפרת	PrivateName
שם המשפחה	כהן	FamilyName
כתובת מייל	Osheri@gmail.com	MailAddress
סטטוס בקשת האירוח	Active	Status
תאריך רישום למערכת	3.1.2020	RegistrationDate
תאריך רצוי לתחילת הנופש	20.8.2020	EntryDate
תאריך רצוי לסיום הנופש	23.8.2020	ReleaseDate
אזור הנופש הרצוי בארץ	All\North\South\Center\Jerusalem	Area
תת-אזור רצוי (לא חובה)	Tel Aviv	SubArea
סוג יחידת האירוח הרצוי	Zimmer\Hotel\Camping\ Etc	Type
מספר המבוגרים	2	Adults
מספר ילדים	7	Children
האם מעוניין בבריכה	הכרחי/אפשרי/לא מעוניין	Pool
האם מעוניין בגקוזי (לא חובה)	הכרחי/אפשרי/לא מעוניין	Jacuzzi
האם מעוניין בגינה (לא חובה)	הכרחי/אפשרי/לא מעוניין	Garden
האם מעוניין באטרקציות לילדים	הכרחי/אפשרי/לא מעוניין	ChildrensAttractions
מתודת הדפסה		*ToString*
מאפיינים נוספים לפי הצורך		Etc….

מחלקה בשם BankBranch המתארת סניף בנק וכוללת:

מאפיין	דוגמא / אפשרויות	
מספר בנק	11	**BankNumber**
שם הבנק	דיסקונט	BankName
מספר הסניף	41	BranchNumber
כתובת הסניף	יפו 220	BranchAddress
עיר הסניף	ירושלים	BranchCity
מתודת הדפסה		*ToString*

מחלקה בשם Host שמייצגת מארח ותכלול:

מאפיין	דוגמא / אפשרויות	
מספר ת.ז. של המארח	12345678	**HostKey**
שם פרטי של המארח	Efrat	PrivateName
שם המשפחה	Choen	FamilyName
מספר טלפון	0547201224	FhoneNumber
כתובת מייל	efratush@gmail.com	MailAddress
פרטי סניף בנק	לפי המחלקה לעיל	BankBranchDetails
מספר חשבון הבנק	166685	BankAccountNumber
אישור גביה מחשבון הבנק	Yes/no	CollectionClearance
מתודת הדפסה		*ToString*
מאפיינים נוספים לפי הצורך		Etc….

מחלקה בשם HostingUnit שמייצגת יחידת אירוח ותכלול:

מאפיין	דוגמא / אפשרויות	
מספר יחידת אירוח – מזהה ייחודי **(קוד רץ ייחודי בן 8 ספרות)**	10000009	**HostingUnitKey**
בעל היחידה – מסוג מארח	מטיפוס Host	Owner
שם יחידת האירוח	נווה מחמד	HostingUnitName
מצב יחידת האירוח לטווח של שנה	מטריצה של 12 חודשים x 31 ימים, עבור כל יום סימון אם היחידה פנויה/ תפוסה. המטריצה צריכה להיות מוגדרת כך: bool[,] Diary כדי שתתאים לעבודה בהמשך עם קבצי XML. ולא כמערך של מערכים.	Diary
מתודת הדפסה		*ToString*
מאפיינים נוספים לפי הצורך		Etc….

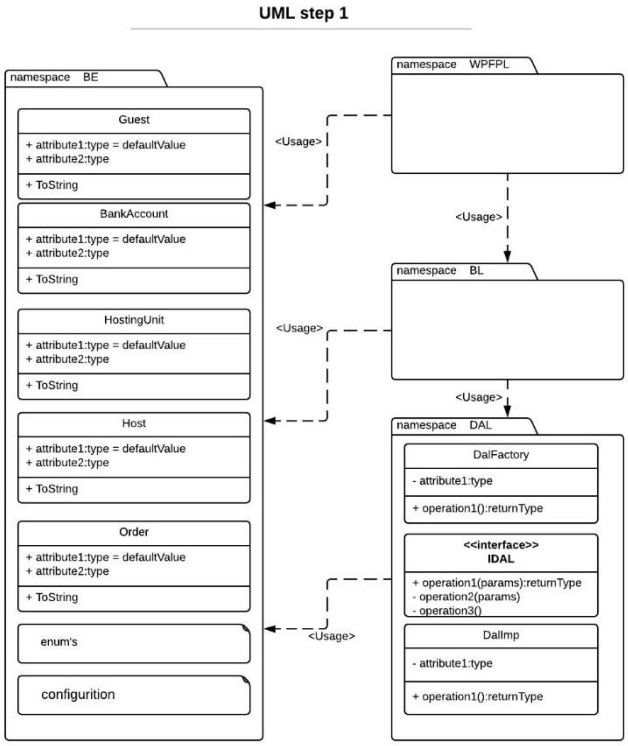
מחלקה בשם Order שמייצגת הזמנה (כלומר את הקשר בין לקוח ליחידת אירוח) ותכלול:

מאפיין	דוגמא / אפשרויות	
מספר מזהה של יחידת האירוח	10000009	**HostingUnitKey**
מספר מזהה של דרישת אירוח	10000002	**GuestRequestKey**
מספר מזהה של ההזמנה	20000183	**OrderKey**
סטטוס ההזמנה	טרם טופל, נשלח מייל, נסגר מחוסר הענות של הלקוח, נסגר בהיענות של הלקוח	Status
תאריך יצירת ההזמנה	5.1.2020	CreateDate
תאריך משלוח המייל ללקוח	7.1.2020	OrderDate
מתודת הדפסה		*ToString*
מאפיינים נוספים לפי הצורך		Etc….

חלק ב' – DAL במימוש רשימות

להלן נפרט פעולות אפשריות. **חובה** להוסיף פעולות נוספות.

הוסף את המחלקות המתאימות לצורך בניית שכבת הנתונים לפרויקט מסוג **class library** בשם **DAL.** להלן תרשים המתאר את הממשק, והנחיות רשומות בהמשך.



א. הגדר (interface) ממשק בשם Idal .

בממשק הנ"ל הגדר חתימה של פונקציות שימושיות עבור האפליקציה כגון:

- הוספת דרישת לקוח.
- עדכון דרישת לקוח (שינוי הסטטוס).

- הוספת יחידת אירוח
- מחיקת יחידת אירוח
- עדכון יחידת אירוח ***את מה מעדכנים?***

- הוספת הזמנה
- עדכון הזמנה (שינוי הסטטוס)

- קבלת רשימת כל יחידות האירוח
- קבלת רשימת כל דרישות הלקוחות.
- קבלת רשימת כל ההזמנות

- קבלת רשימת כל סניפי הבנק הקיימים בארץ (מחזיר אוסף של סניפים מסוג המבנה המתאים שהוגדר ב-BE)

ב. צור פרויקט חדש בשם: DS והגדר בתוכו מחלקה בשם DataSource שתכיל את הנתונים (רשימות) של הישויות שלנו.
מחלקה זו תכיל 3 רשימות סטטיות (אוספים מסוג <>list) של המחלקות הנמצאות ב- BE.
(הערה: זוהי מחלקה זמנית לשלב זה של הפרויקט. ניתן לאתחל את הרשימות באמצעות קלט, אך מומלץ לאתחל את הרשימות במספר ערכים בקוד האתחול, כדי להקל על העבודה)

ג. הגדר מחלקה בשם: Dal_imp אשר מממשת את ממשק ה- Idal הנ"ל.
הפונקציות של המחלקה הזאת יעבדו מול הרשימות שבמחלקה DataSource.
(אין להוסיף פונקציונאליות מעבר לCRUD [=יצירה, קריאה, עדכון, מחיקה] בסיסי לDAL)

הערה לגבי הפונקציה שמחזירה את רשימת סניפי הבנק:
בשלב זה יש לממש כך שהפונקציה תחזיר רשימה של 5 סניפים שתגדיר בתוך הפונקציה.
בחלק הבא של הפרויקט הפונקציה תחזיר את המידע על הסניפים בארץ ישירות מהמידע שמופיע בבנק ישראל ולכן אין צורך להמציא יותר מ 5 סניפים בשלב זה.
(בניגוד לפונקציונאליות עבור שאר במחלקות, במחלקה סניף נרצה רק לקבל סניפים קיימים ממאגר המידע שלנו ולא להוסיף, לעדכן או למחוק סניפים)

דגש - שים לב!

בגלל חיוניות ההפרדה בין השכבות, שכבת ה DAL לא תאפשר לשכבות האחרות גישה למקור הנתונים– כלומר, במקרה הצורך, שכבה זו תשלח עותק של הנתונים, ולא את הנתונים עצמם.
קרא הדרכה בנספח מספר 1

הערה:
על שכבת ה- DAL לוודא שמספר הת.ז. איננו קיים כבר.
שכבת ה- DAL היא זו שמוסיפה לישויות "לקוח" ו"יחידת אירוח" מספר מזהה רץ.
על מנת לקבוע את המספר המזהה הרץ ההתחלתי, יש לשמור נתון מתאים במחלקת Configuration. לכל ישות הזקוקה למספר רץ, יש לשמור שדה נפרד ב Configuration.

דרישות תכנותיות מינימליות עבור שכבה זו:

- ☒ שימוש לכל הפחות ב - 4 ביטויי Linq to object לצורך השאילתות.
- ☒ השימוש בביטויי לינק צריך להיות מגוון, ולכלול שימוש ב new select וב let.
- ☒ יש לעשות שימוש ב- Factory Method.
- ☒ יש לוודא שאנו לא יוצרים כל פעם מופע חדש של הDAL.
- ☒ מחלקת config היא אשר תשמור ותקדם את הקוד הייחודי לכל ישות בשכבה זו. ע"י שימוש בשדה שמשמש כמספר רץ.
- ☒ שימו לב. פונקציית ההוספה של ה DAL תבדוק אם לעצם ישנו כבר קוד ייחודי (למשל – במקרה של עדכון) ובמידה והוא לא קיים – תוסיף אותו לעצם.

חלק ג' – שכבת ה- BL

הגדר ממשק (interface) בשם: IBL שבו החתימות של השיטות בדיוק כמו ב- IDAL

על שכבת ה BL לבצע את אכיפת הלוגיקה הבאה (וכמובן – עליכם להוסיף לוגיקה משלכם, לפי הגיון פשוט ודרישות המתאימות לפרויקט שלכם):

- תאריך תחילת הנופש קודם לפחות ביום אחד לתאריך סיום הנופש
- בעל יחידת אירוח יוכל לשלוח הזמנה ללקוח (שינוי הסטטוס ל "נשלח מייל"), רק אם חתם על הרשאה לחיוב חשבון בנק.
- יש לוודא בעת יצירת הזמנה ללקוח, שהתאריכים המבוקשים פנויים ביחידת האירוח שמוצעת לו.
- לאחר שסטטוס ההזמנה השתנה לסגירת עיסקה – לא ניתן לשנות יותר את הסטטוס שלה.
- כאשר סטטוס ההזמנה משתנה בגלל סגירת עסקה – יש לבצע חישוב עמלה בגובה של 10 ₪ ליום אירוח. (עיין הערה למטה)
- כאשר סטטוס ההזמנה משתנה בגלל סגירת עסקה – יש לסמן במטריצה את התאריכים הרלוונטיים.
- כאשר סטטוס הזמנה משתנה עקב סגירת עסקה – יש לשנות את הסטטוס של דרישת הלקוח בהתאם, וכן לשנות את הסטטוס של כל ההזמנות האחרות של אותו לקוח.
- לא ניתן למחוק יחידת אירוח כל עוד יש הצעה הקשורה אליה במצב פתוח.
- לא ניתן לבטל הרשאה לחיוב חשבון כאשר יש הצעה הקשורה אליה במצב פתוח.
- כאשר סטטוס ההזמנה משתנה ל"נשלח מייל" – המערכת תשלח באופן אוטומטי מייל ללקוח עם פרטי ההזמנה. ניתן לדחות את הביצוע בפועל של שליחת המייל לשלב הבא, וכעת רק להדפיס הודעה על המסך.

הערה: כל הנתונים המספריים (כגון: גובה העמלה) צריך להיות שמורים במשתנים סטטיים שבמחלקה configuration, על מנת שניתן יהיה לשנותם באופן פשוט, במידה ויוחלט על שינוי בנהלים.

בנוסף על שכבת ה BL להוסיף את חתימת הפונקציות הבאות:

- פונקציה שמקבלת תאריך ומספר ימי נופש ומחזירה את רשימת כל יחידות האירוח הפנויות בתאריך זה.
- פונקציה שמקבלת תאריך אחד או שניים. הפונקציה מחזירה את מספר הימים שעברו מהתאריך הראשון ועד לשני, או במיה והתקבל רק תאריך אחד – מהתאריך ראשון ועד היום.
- פונקציה שמקבלת מספר ימים, ומחזירה את כל ההזמנות שמשך הזמן שעבר מאז שנוצרו / מאז שנשלח מייל ללקוח גדול או שווה למספר הימים שהפונקציה קיבלה.

- פונקציה שיכולה להחזיר את **כל דרישות הלקוח** שמתאימים לתנאי מסוים (הכוונה שהפונקציה מקבלת delegate שמתאים למתודה שפועלת על דרישת לקוח ומחזירה bool וכך מוגדר התנאי)
- פונקציה שמקבלת דרישת לקוח, ומחזירה את מספר ההזמנות שנשלחו אליו.
- פונקציה שמקבלת יחידת אירוח ומחזירה את מספר ההזמנות שנשלחו / מספר ההזמנות שנסגרו בהצלחה עבור יחידה זו דרך האתר.

הגדר פונקציות המחזירות את הקבוצות הבאות (ע"י שימוש ב-Grouping)

- רשימת דרישות לקוח מקובצת (Grouping) ע"פ אזור הנופש הנדרש.
- רשימת דרישות לקוח מקובצת (Grouping) ע"פ מספר הנופשים.
- רשימת מארחים מקובצת (Grouping) ע"פ מספר יחידות האירוח שהם מחזיקים
- רשימת יחידות אירוח מקובצת (Grouping) ע"פ אזור הנופש הנדרש.

הגדר מחלקה חדשה בשכבת ה- BL שתממש את הממשק IBL הנ"ל.

<u>הערות:</u>

על שכבה זו לוודא שמתקיימים חוקים לוגיים בסיסים כמו לדוגמה:

- אם מוסיפים הזמנה, אזי יש לוודא שהלקוח ויחידת האירוח אכן קיימים.
- לא ניתן לקבוע אירוח לתאריך שכבר תפוס ע"י לקוח אחר

<u>דרישות תכנותיות מינימליות עבור שכבה זו:</u>

☒ שימוש לכל הפחות ב - 4 ביטויי Linq to object לצורך השאילתות.
☒ השימוש בביטויי לינק צריך להיות מגוון, ולכלול שימוש בnew select וב let.
☒ שימוש לכל הפחות ב- 4 ביטויי למבדא.
☒ שימוש בdelegates anonymous
☒ שימוש ב- predicates ב- ב- FUNC.
☒ חובה לעשות שימוש בכישרון היצירתיות שלכם ולהוסיף לפחות עוד 6 פונקציות שמתאימות להיות בשכבת ה-BL ועובדות על הנתונים המוחזרים מה- DAL.
☒ יש לעשות שימוש ב- Factory Method.

חלק ד – יצירת שכבת UI

לשם הבדיקה של הדברים אנו ניצור פרויקט זמני בשם PL.
תוכל לממש אותו בעזרת console application או ממשק גרפי WPF פשוט. אפשר בשלב זה לעשות הכל בחלון אחד.
בכל מקרה עליך לקרוא לפונקציות הנמצאות ב- BL ולבדוק אותן.

כאן תתבצע ההגשה הראשונה.

<u>דרישות תכנותיות מינימליות נוספות עבור הגשה זו:</u>

- ☒ החלוקה בין השכבות נכונה
- ☒ בכל שכבה יש ממשק ומימוש נפרדים
- ☒ טיפול בחריגות באופן מתאים
- ☒ חריגות עוברות משכבה לשכבה
- ☒ בין השכבות עוברים רק עותקים של המידע ולא המקור
- ☒ תיעוד נכון

<u>הערה:</u>

כדי לעבוד בשלבים מומלץ בהתחלה ליצור משהו שבודק את ה- BE אח"כ את הפונקציונליות ב- DAL כדי לראות שאכן זה עובד ואז לחבר את ה- BL.

כמו כן יש לבצע תיעוד מסודר:

- ✓ מעל הפונקציות בממשקים של IDAL ו- IBL תוך שימוש ב \\\
- ✓ על מנת להקל על התנועה במסך הקוד יש להשתמש בregion
- ✓ להוסיף תיעוד עבור מימושים רק מעל פונקציות יצירתיות ב- BL שאוכפות נהלים מסוימים שהמצאתם כמו למשל "מצא את כל הלקוחות שמעוניינים בנופש בירושלים עם בריכה" בקיצור, תיעוד רק עבור דברים שאינם טריוויאליים או שימוש בשאילתות מורכבות. כל השאר אין צורך לתעד.
- ✓ יש להשתמש בשמות משמעותיים למשתנים

שלב שני – עדכון שכבת ה- UI

בשלב זה ניצור ממשק גרפי לפרויקט באמצעות תשתית הפתוח WPF.

ניצור פרויקט חדש מסוג : WPF
ונקרא לו **PLWPF**.
כמובן שיהיה עליך ליצור לו reference מתאים ל-BL וכן ל- BE .

עליך להגדיר ישות בשם: myBL מטיפוס: IBL

לתכנן את המסכים אשר יקראו ויאפשרו למשתמש לגשת לפונקציונאליות הנמצאת ב- BL.
עליך לתכנן מסך לכל ישות, כאשר יהיו לפחות המסכים הבאים (ניתן להיעזר בשרטוט המופיע בעמוד 6):

- מסך הוספה של דרישת לקוח.
- מסך הוספת, עדכון, מחיקה של יחידת אירוח.
- מסך הוספת, עדכון של הזמנה.
- בנוסף – לפחות עוד 3 מסכים המציגים שאילתות שמימשתם ב- BL.

התכנית תיפתח במסך ראשי המפנה לאפשרויות השונות.

הערה: כאשר ישנו מסך המאפשר הוספת או עדכון ערך כמו "אזור הנופש הרצוי" וכדו' שמוגדרים במערכת, יש לאפשר בחירה של שדה זה ע"י פקד ComboBox שבו נקבל רשימה ממנה יש לבחור.

כמו כן כאשר אנו מאפשרים עדכון, לאחר בחירה מתוך רשימת הישויות לעדכון, יופיע המפתח על המסך ויתמלאו יתר השדות באופן אוטומטי, כאשר לא ניתן יהיה כמובן לשנות את ערך המפתח.
מומלץ מאד שניתן יהיה לחפש את הרשומה לעדכון גם על סמך נתונים אחרים בישות (למשל – שם, שם משפחה וכדומה), ולא רק על סמך המפתח (מספר).
ישנה אפשרות שהוספה עדכון ומחיקה יהיו למעשה אותו מסך, בהתאם לצורך באותו רגע.

כמו כן עליך לדאוג להוסיף זריקה של חריגות במקומות המתאימים ותפיסה של החריגות כדי שהתכנית לא "תעוף" במקרה שפעולה מסוימת נכשלה בזמן הרצה.

בשכבה זו עלייך לטפל בתהליכון המזמן את שירות הרשת לשליחת מייל ללקוח, כך שיפעל מהעצם המזמן אותו. ניתן לדחות את הביצוע של כך לשלב השלישי של הפרויקט.

לאחר השלמת לימוד כל הWPF יש צורך לדאוג שכל האלמנטים שנלמדו בWPF מוטמעים בפרויקט. היכן ואיך לממש, זה תלוי בכל אחד ואחת לפי יצירתיותם.
כמו כן, יש לטפל בחריגות שנזרקות ומטופלות ששייכות לשכבת UI בעצמה כמו למשל שמשתמש לא מילא את כל השדות הנדרשים או שמילא שדות נומריים בתווים וכו'.

<u>דרישות תכנותיות מינימליות עבור שכבה זו:</u>

- ☒ שימוש ב dependency property
- ☒ שימוש ב data context
- ☒ שימוש בResources (גם ברמת האפליקציה)
- ☒ שימוש ב data binding
- ☒ שימוש ב converter
- ☒ שימוש בתהליכון. (2 תהליכונים, כפי שידרשו בשלב שלישי של הפרוייקט)
- ☒ שימוש בdata template
- ☒ שימוש בstyle
- ☒ שימוש ב trigger

כאן תתבצע ההגשה השניה במספר.

שלב שלישי – עדכון שכבת ה- DAL
מימוש באמצעות Linq-To-XML והוספת תהליכונים ושאילתות רשת

1. הוספה ומימוש מחלקה נוספת ב-DAL המממשת את ה- IDAL באמצעות – Linq – to XML:

עליך להוסיף לפרויקט **DAL** מחלקה נוספת בשם Dal_XML_imp

יש להכין קבצי xml שיחליפו את האוספים שיצרנו כבר (כלומר יחליפו את מקור הנתונים) אחד לכל אחת מהישויות, אשר ייכתבו בפורמט המתאים למבנה הישות אותה הוא מייצג.
קבצים אלו יהיו בתוך תיקיה נפרדת ב- solution.

יש ליצור את האתחולים, אפשרות שמירה וטעינה של קובץ וכן אפשרות תשאול ע"י שאילתת Linq.
לאחר מכן יש לממש את כל המתודות של הממשק של ה- IDAL.
לגבי עבודה עם קבצי XML מקומיים, עיין בנספח 6.

2. הוספת שאילתות רשת

יש להוסיף שאילתת רשת כדי לקבל את רשימת סניפי הבנקים מבנק ישראל (עיין בנספח 2)
יש להוסיף שאילתת רשת על מנת לשלוח את הודעות המייל ללקוחות (עיין בנספח 3).
לשאילתות אלו יש להיעזר בתהליכון כפי שמוסבר בסעיף הבא.

3. הוספת תהליכונים

- מכיוון שהרשת עמוסה לעיתים, אזי בקשת רשת יכולה לארוך כמה חלקי שניות וכן להיענות רק לאחר כמה ניסיונות.
  לכן, עליכם <u>לעטוף את שאילתות הרשת בתוך תהליכון</u> **BackgroundWorker**.
  התהליכון ימשיך ויבצע נסיונות לבקשת רשת עד שתתקבל תשובה.
  ההמלצה היא לבצע בתוך התהליכון השהיה של 2 שניות לפחות בין בקשת רשת אחת לשניה.
- יש להוסיף תהליכון שמופעל אחת ליום ומעדכן את הסטטוס של כל ההזמנות שפג תוקפן (עבר למעלה מחודש מאז שנשלח המייל מהמערכת) – זהו תהליכון שמופעל ע"י שכבת הBL, ולכן לא משתמש ב BackgroundWorker אלא אך ורק ב thread.

<u>דרישות לוגיות לשכבה זו:</u>

- ☒ לקוח יכול לצפות רק במסכי לקוח.
- ☒ כשלקוח נרשם למערכת/כשנוצרת הזמנה – הסטטוס נקבע אוטומטית

- ☒ תאריך ההזמנה – ברירת מחדל תאריך של אותו יום (לאפשר גם קביעה שלו עצמאית – כדי לבדוק אם התהליכון עובד)
- ☒ מארח יכול לבצע שאילתות על לקוחות
- ☒ מארח יכול לצפות ברשימת ההזמנות שלו.
- ☒ בעל האתר יוכל לצפות במסכי שאילתות, המאפשרות לו למיין את הנתונים באתר, לקבל חישובי עמלה וכדומה – כיד הדמיון הטובה עליכם.

<u>דרישות תכנותיות מינימליות עבור שכבה זו:</u>

- ☒ עבור קובץ ה- config ועוד קובץ אחד של אחת המחלקות, יש להשתמש ב- linq to xml עבור כל פעולות ההוספה עדכון ומחיקה ושליפה.
- ☒ במימוש של שאר המחלקות ניתן להשתמש ב- serialize, כלומר לעבוד עם הנתונים ב- list ובסופו של דבר לשמור את זה ל- XML באמצעות xmlSerialize
- ☒ עבור המטריצה, יש להשתמש בattribute של xmlIgnore (עיין נספח מספר 4)

כאן תתבצע בעז"ה ההגנה הסופית

בהצלחה !

תאריכי הגשות

תאריכי ההגשה הללו מחייבים.
איחור בהגשה של השלבים הראשונים עלול להביא לקנס בנקודות.
איחור בהגשה של החלק השלישי עלול להביא לכך שהפרויקט לא ייבדק ולא יתקבל עליו ציון כלל.

הגשה 1 – שלב א (חלקים א-ג עם UI בסיסי). תאריך הגשה: עד לג' בטבת
הגשה 2 – שלב ב (חלק ד, UI עם WPF). תאריך הגשה: עד לכ"ג בטבת
הגשה 3 – לאחר שלב ג' סופי – **תאריך הגשה** : מפגש אחרון בסמסטר (עד לה' בשבט)

(ההגנה בשבוע האחרון בשעות המעבדה). לאחר סיום הסמסטר לא יתבצעו בדיקות.

להזכירכם: ההגנה מתבצעת מול מרצה עמית (לא המרצה שלימד אתכם), כאשר הציון הסופי הוא שקלול של חוות הדעת על הצגת ההרצה למרצה העמית, ושל חוות הדעת של המרצה שלימד אתכם על איכות כתיבת הקוד.

נספח 1

יצירת העתקים להעברה בין השכבות

נתוני הזיכרון של התכנית (או לפחות חלקם) מוחזקים, כידוע, בשכבת ה- DAL.
בגלל עקרון ההפרדה בין השכבות, לא נרצה ששכבת הDAL תאפשר ל BL גישה דרך הממשק שלה לנתונים השמורים בה.
כמו כן, לא נרצה ששכבת ה DAL תשמור ישויות שהתקבלו מ-BL בנתונים שלה (כי ייתכן שהמופעים ישמשו את שכבת ה BL לצרכים נוספים)
כלומר: נרצה ששכבה תעביר לשכבה אחרת עותק ממשי של הנתונים, ולא הפניות לישויות ולאוספים הנמצאים בה (מפני שכאשר מועברת הפניה, הנתונים עלולים להיפגע ולהשתנות ללא ביקורת של השכבה בה הם מוחזקים).
לכן כל ישות בודדת או אוסף של ישויות העובר מ DAL אל BL וההפך, חייב לעבור תהליך העתקה/שיבוט.
ההעתקה חייבת להיות עמוקה ותעשה בעזרת הוספה של מחלקה סטטית עם פונקציות הרחבה בשכבת ה DAL בלבד ולא באף שכבה אחרת.

<u>העברת נתונים מ-DAL – דרך 1</u>

ניצור בשכבה DAL מחלקה **סטטית** Cloning
המחלקה תכיל פונקציות **הרחבה** Clone() עבור כל אחת מהמחלקות ב-BE, למשל:

```
public static class Cloning
    {
        public static Student Clone(this Student original)
        {
            Student target = new Student();
            target.id = original.id;
      ...
      return target;
        }

       public static Lecture Clone(this Lecture original)
        {
            Lecture target = new Lecture();
            target.id = original.id;
      ...
      return target;
        }

       //and so on for each entity…

    }
```

העברת נתונים מ-DAL – דרך 2

ניצור בפרויקט המכיל את הממשק של שכבת BE ממשק **ריק** בשם Clonable
כל מחלקות BE יממשו את הממשק הנ"ל (זהו ממשק סימון לצורך פונקציית ההרחבה).
בשכבה DAL ניצור מחלקה Cloning שתכיל פונקציית הרחבה ()Clone עבור הממשק Clonable
הפונקציה הנ"ל תשתמש ב reflaction על מנת לבצע העתקה

```
public static class Cloning
 {
     public static Clonable Clone(this Clonable original)
     {
         Clonable target = Activator.CreateInstance(original.GetType());
          //...
          return target;
     }
 }
```

העברת נתונים מ-DAL – דרך 3

בשכבה DAL ניצור מחלקה סטטית Cloning שתכיל פונקציית הרחבה גנרית.
הפונקציה הנ"ל תשתמש ב reflection על מנת לבצע העתקה
יש להבין את הקוד לפני השימוש בו!

```
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
public static T Copy<T>(this T source)
        {
            var isNotSerializable = !typeof(T).IsSerializable;
            if (isNotSerializable)
                throw new ArgumentException("The type must be serializable.", "source");

            var sourceIsNull = ReferenceEquals(source, null);
            if (sourceIsNull)
                return default(T);

            var formatter = new BinaryFormatter();
            using (var stream = new MemoryStream())
            {
                formatter.Serialize(stream, source);
                stream.Seek(0, SeekOrigin.Begin);
                return (T)formatter.Deserialize(stream);
            }
        }
```

דוגמת מימוש בשכבת הDAL

שאילתת אובייקט

```
public Student GetStudent(int id)
{
    Student stud = DataSource.Students.FirstOrDefault(stud->stud.Id == id);
    return stud == null ? null : stud.Clone();
}
public bool CheckStudent(int id)
{
    return DataSource.Students.Any(stud->stud.Id == id);
}
```

שאילתת רשימה

```
public IEnumerable<Student> Student GetStudents()
{
    return from stud in DataSource.Students
           select stud.Clone();
}
```

הוספת אובייקט

```
public void AddStudent(Student stud)
{
    if (!CheckStudent(stud.Id))
        throw new DuplicateIdException("Student", stud.Id);
    DataSource.Students.Add(stud.Clone());
}
```

מחיקת אובייקט

```
public void DelStudent(int id)
{
    int count = DataSource.Students.RemoveAll(stud->sutd.Id == id);
    if (count == 0)
        throw new MissingIdException("Student", id);
}
```

עדכון אובייקט

```
public void UpdStudent(Student stud)
{
    int count = DataSource.Students.RemoveAll(stud->sutd.Id == id);

    if (count == 0)
        throw new MissingIdException("Student", id);
    AddStudent(stud);
}
```

העברת נתונים מ-DAL – דרך 4

זוהי דרך יצירתית, העושה שימוש בעובדה שהפונקציה **CopyTo** שמבצעת העתקה ממערך לרשימה, מבצעת העתקה עמוקה.

```
public List<Student> Get StudentsList()
{
    Student[] studentsArr = new Student[DataSource.students.Count];
    DataSource.students.CopyTo(studentArr);
    return studentsArr.ToList();

   //instead of:
    //return DataSource.students.Select(item => item).ToList();
}
```

נספח 2
שאילתת רשת לסניפי הבנק

עבור קבלת רשימת סניפי הבנק הקיימים בארץ, נשתמש במסמך ה XML הבא שנמצא באתר בנק ישראל :
http://www.boi.org.il/he/BankingSupervision/BanksAndBranchLocations/Lists/BoiBankBranchesDocs/atm.xml

מכיוון שיש זמנים שבהם האתר הנ"ל מושבת, יש לכם גיבוי של כל המידע (נכון ל 09.12.16)
גם בכתובת הבאה:
https://drive.google.com/file/d/1FpcqslnRD6naLHOjrCvKArCg3Ihkb9hR/view?usp=sharing

כיצד להוריד את הקובץ ?
ניתן להשתמש במחלקה WebClient של דוט נט כך:

```
const string xmlLocalPath = @"atm.xml";

WebClient wc = new WebClient();
try
{
    string xmlServerPath =
@"http://www.boi.org.il/he/BankingSupervision/BanksAndBranchLocations/Lists/BoiBankBranchesDocs/atm.xml";
    wc.DownloadFile(xmlServerPath, xmlLocalPath);

}
catch (Exception)
{
    string xmlServerPath = @"http://www.jct.ac.il/~coshri/atm.xml";
    wc.DownloadFile(xmlServerPath, xmlLocalPath);
}
finally
{
    wc.Dispose();
}
```

הסבר:
המשתנה xmlLocalPath מכיל את הנתיב לשמירת הקובץ במחשב
שימו לב שמנסים להוריד מהאתר של בנק ישראל ובמידה ולא מצליחים מנסים להוריד מאתר השני.
אפשרות נוספת:
אם נעביר לפונקציה Load של XElement את כתובת האינטרנט, להפתעתנו הרבה גם זה יעבוד ונקבל XElement שתואם לקובץ ה xml
אבל, בכל אופן לענ"ד כדאי להוריד את הקובץ למחשב ואותו לטעון ב-XElement

עליך, להסתכל במבנה קובץ ה XML של בנק ישראל ולשלוף ממנו את המידע המתאים לרשימת הבנקים האפשריים (וכמובן, רק את המידע הרלוונטי).


```
▼<ATMs xmlns:ns0="BOI_atm">
  ▼<ATM>
      <קוד_בנק/>13<קוד_בנק>
      <שם_בנק/>בנק אגוד לישראל בע"מ-13001<שם_בנק>
      <קוד_סניף/>157<קוד_סניף>
      <כתובת_ה/>כתובת_ה-ATM<ויצמן 32 -ATM>
      <ישוב/>תל אביב -יפו<ישוב>
      <עמלה/>לא<עמלה>
      <סוג_ATM/>סוג מכשיר מידע/או מתן הוראות_ATM>
      <מיקום_ה/>מיקום_ה-ATM<על קיר הסניף/ביחס_לסניף-ATM_ביחס_לסניף>
      <גישה_לנכים/>לא<גישה_לנכים>
      <קואורדינטת_X>34.789425</קואורדינטת_X>
      <קואורדינטת_Y>32.084588</קואורדינטת_Y>
    </ATM>
  ▼<ATM>
      <קוד_בנק/>13<קוד_בנק>
      <שם_בנק/>בנק אגוד לישראל בע"מ-13001<שם_בנק>
      <קוד_סניף/>192<קוד_סניף>
      <כתובת_ה/>כתובת_ה-ATM<הגדוד העברי 10 -ATM>
      <ישוב/>אשקלון<ישוב>
      <עמלה/>לא<עמלה>
      <סוג_ATM/>סוג מכשיר מידע/או מתן הוראות_ATM>
      <מיקום_ה/>מיקום_ה-ATM<על קיר הסניף/ביחס_לסניף-ATM_ביחס_לסניף>
      <גישה_לנכים/>כן<גישה_לנכים>
      <קואורדינטת_X>34.585473</קואורדינטת_X>
      <קואורדינטת_Y>31.665121</קואורדינטת_Y>
    </ATM>
  ▼<ATM>
      <קוד_בנק/>13<קוד_בנק>
      <שם_בנק/>בנק אגוד לישראל בע"מ-13001<שם_בנק>
      <קוד_סניף/>192<קוד_סניף>
      <כתובת_ה/>כתובת_ה-ATM<הגדוד העברי 10 -ATM>
      <ישוב/>אשקלון<ישוב>
      <עמלה/>לא<עמלה>
```


למען האמת מה שמופיע בקובץ זה מידע על כל הכספומטים של הבנקים.
אבל, מתוך הנחה שלכל סניף יש לפחות כספומט אחד, נוכל באמצעות הקובץ לקבל את כל הסניפים הקיימים בארץ (וכמובן כדאי להוריד כפילויות למקרה שיש לסניף יותר מכספומט אחד)
שים לב שהאלמנטים אמנם כתובים בעברית אבל זה לא מפריע לנו לקרוא בצורה זו את הנתונים במערכת הדוט-נט.

שים לב שייקח זמן עד שהקובץ ירד ובזמן זה התוכנית לא תוכל להגיב
ולכן, מומלץ שהבנאי של ה DAL יבצע הורדה מהשרת לקובץ המקומי בתהליכון נפרד.
שאר הפונקציות שצריכות לגשת לקובץ זה יבדקו האם ההורדה הסתיימה ורק אם היא הסתיימה יאפשרו את הפעלת הפונקציה, אחרת תיזרק שגיאה בהתאם.
(ניתן להגדיר משתנה בוליאני שיציין האם ההורדה הסתיימה)

נספח 3
שאילתת רשת לשליחת מייל

מתבסס על ההדרכה של גיא לוין באתר "עץ הדעת" –
http://www.ets-daat.com/Articles/Article.aspx?articleid=1009

בדוט נט קיימת ספריה מובנית לטיפול יעיל בשליחת מיילים.
נוסיף בראש הקוד של המחלקה שלנו (או ל Code Behind בהתאם לצורך) :

```
using System.Net.Mail;
```

בתוך ספריה זו יש אובייקט בשם MailMessage שמגדיר הודעת מייל, עם כל הנתונים והפרטים הדרושים. האובייקט מכיל הרבה מתודות ואפשרויות - נזכיר את החשובים מבניהם:

- To - למי נשלחת ההודעה (נמען אחד או יותר)
- From - ממי נשלחת ההודעה
- Subject - נושא ההודעה
- Body - תוכן ההודעה (טקסט רגיל או ערוך כקוד HTML)
- IsBodyHTML - מגדיר את סוג תוכן ההודעה.

יצירת אובייקט חדש מתבצעת על ידי:

```
MailMessage mail = new MailMessage;
```

כעת יש לטפל בשליחת ההודעה עצמה. זה מתבצע ע"י מנגנון בשם SmtpClient – שהוא עצם המגדיר את כל הפרטים הדרושים לשליחת אוביקט ה MailMessage - האובייקט מכיל הרבה מתודות ואפשרויות - נזכיר את החשובים מבניהם:

- Host - כתובת השרת המארח
- Credentials - צורת חיבור המשתמש לשרת, כאן נגדיר גם את שם המשתמש והסיסמה
- EnableSsl - אפשור תקשורת Ssl, (חובה עבור שרתים מסויימים)
- Send() - שליחת ההודעה

כמו כן מנגנון ה SmtpClient -מספק לנו דרך מצוינת לנטר שגיאות בשליחת המייל, ע"י חריגות מובנות:

- ArgumentNullException - ההודעה ריקה (null)
- SmtpException - שגיאה בעת התחברות לשרת

- SmtpFailedRecipientsException - ההודעה לא יכלה להישלח לחלק מהנמענים
- InvalidOperationException - בעיה בנתונים שהוכנסו בהודעה (נתונים חסרים או שגויים)

שליחת מייל דרך GMAIL

```
//MailMessage יצירת אובייקט
MailMessage mail = new MailMessage();

// כתובת הנמען(ניתן להוסיף יותר מאחד)
mail.To.Add("toEmailAddress");

// הכתובת ממנה נשלח המייל
mail.From = new MailAddress("fromEmailAddress");

// נושא ההודעה
mail.Subject = "mailSubject";

//( תוכן ההודעה (נניח שתוכן ההודעה בפורמט HTML
mail.Body = "mailBody";

// הגדרה שתוכן ההודעה בפורמט HTML
mail.IsBodyHtml = true;

// יצירת עצם מסוג Smtp
SmtpClient smtp = new SmtpClient();

// הגדרת השרת של gmail
smtp.Host = "smtp.gmail.com";


// הגדרת פרטי הכניסה (שם משתמש וסיסמה) לחשבון ה gmail
smtp.Credentials = new System.Net.NetworkCredential("myGmailEmailAddress@gmail.com",
"myGmailPassword");
// ע"פ דרישת השר, חובה לאפשר במקרה זה SSL
smtp.EnableSsl = true;
try
{
    // שליחת ההודעה
    smtp.Send(mail);
}
catch (Exception ex)
{
    // טיפול בשגיאות ותפיסת חריגות
    txtMessage.Text = ex.ToString();
}
```

כמובן, שליחת המייל מתבצעת ע"י חשבון קיים, ולכן יש לאפשר לשרת להיכנס לחשבון ממנו נשלח המייל. לצורך כך מגדיר הCredentials את שם המשתמש והסיסמה.
במקום להגדיר את ה- Credentials ישירות באובייקט, יש אפשרות להוסיף אותו בקובץ קונפיגורציה, לדוגמה:

```
<configuration>
  - <system.net>
      - <mailSettings>
          - <smtp from="myEmail@hotmail.com">
              <network password="Password" userName="UserName" port="PortNumber"
                  defaultCredentials="false" host="HostSMTPAddress"/>
            </smtp>
        </mailSettings>
    </system.net>
</configuration>
```

נספח 4
שמירת מטריצה בXML

כאמור, אנו מעוניינים לשמור את בסיס הנתונים שלנו בקבצי XML.
לצורך כך יש לבצע סריאליזציה של האובייקטים בהכנסה לקובץ XML ודה-סריאלזציה בשליפת האובייקטים מקובץ הXML (כפי שלמדנו, בעזרת XmlSerializer וכן, עיין בנספח מספר 6).
מבנה הנתונים מטריצה, לא יכול להישמר כמות שהוא בקובץ XML, מאחר והוא לא מבנה נתונים "שטוח". לכן, נדרש מעט תחכום על מנת לשמור אותו כראוי ולהצליח לקרוא אותו מן הקובץ.
הרעיון באופן כללי הוא להוסיף Attribute בשם [XmlIgnore] מעל הproperty של המטריצה, שמסמן למחשב להתעלם מהמטריצה בזמן סריאליזציה/דה-סריאלזציה של האובייקט כולו לתוך/מ- קובץ XML.
ובמקביל ליצור property נוסף מסוג מערך חד מימדי, שתפקידו לשרשר את המטריצה למערך חד מימדי, ובכיוון ההפוך - לחלץ מטריצה ממערך חד מימדי. (אופציונאלי: להוסיף מעל המערך Atrribute בשם [XmlArray("")] כדי לקבוע את שם האלמנט שבו יקרא המערך בקובץ)

להלן דוגמא:

נניח שקיימת לנו המחלקה הבאה:

```
public class Player
{
    public string name { get; private set; }
    public bool[,] Board { get; set; }
}
```

נתקן את המחלקה כך:

```
namespace BE
{
    public class Player
    {
        public string Name { get; set; }

        // tell the XmlSerializer to ignore this Property.
        [XmlIgnore]
        public bool[,] Board { get; private set; }

        //optional. tell the XmlSerializer to name the Array Element as'Board'
        // instead of 'BoaredDto'
        [XmlArray("Board")]
        public bool[] BoardDto
        {
            get { return Board.Flatten(); }
            set { Board = value.Expand(5); } //5 is the number of roes in the matrix
        }

    }
```

ניצור מחלקת עזר:

```
    public static class Tools
    {
        public static T[] Flatten<T>(this T[,] arr)
        {
            int rows = arr.GetLength(0);
            int columns = arr.GetLength(1);
            T[] arrFlattened = new T[rows * columns];
            for (int j = 0; j < columns; j++)
            {
                for (int i = 0; i < rows; i++)
                {
                    var test = arr[i, j];
                    arrFlattened[i + j * rows] = arr[i, j];
                }
            }
            return arrFlattened;
        }

        public static T[,] Expand<T>(this T[] arr, int rows)
        {
            int length = arr.GetLength(0);
            int columns = length / rows;
            T[,] arrExpanded = new T[rows, columns];
            for (int j = 0; j < rows; j++)
            {
                for (int i = 0; i < columns; i++)
                {
                    arrExpanded[i, j] = arr[i + j * rows];
                }
            }
            return arrExpanded;
        }
    }
}
```

וכעת נוכל לבצע סריאליזציה/דה-סריאלזציה כרגיל, כך:

```
using System.Xml.Serialization;

public static void SaveToXML<T>(T source, string path)
{
    FileStream file = new FileStream(path, FileMode.Create);
    XmlSerializer xmlSer = new XmlSerializer(source.GetType());
    xmlSer.Serialize(file, source);
    file.Close();
}

public static T LoadFromXML<T>(string path)
{
    FileStream file = new FileStream(path, FileMode.Open);
    XmlSerializer xmlSer = new XmlSerializer(typeof(T));
    T result = (T)xmlSer.Deserialize(file);
    file.Close();
    return result;
}
```

המטריצה תשמר בקובץ כמערך חד מימדי.

נספח 5

עקרונות מחייבים לטיפול בחריגות

כל תקלה כתוצאה מפעולה כלשהי, תגרור אחריה טיפול בחריגות לפי מנגנון החריגות הקיים ב-#C. לדוגמא: הכנסת מספר מזהה לא תקין, ניסיון לקבוע נופש ליחידת אירוח שלא קיימת במערכת.
החריגות צריכות להתבצע עבור כל שכבה בנפרד, ללא תלות בטיפול בחריגה זו בשכבות האחרות.
כאשר נזרקת חריגה בשכבת הDAL או הBL – יש לתפוס אותה בשכבה שמעליה (BL או UI – בהתאמה) - שימו לב שייתכן והטיפול יהיה זריקת חריגה מחדש או זריקת חריגה חלופית.
כמובן שבשכבת ממשק המשתמש לא תיזרקנה חריגות אלא ינתן טיפול מתאים המבקש מהמשתמש לתקן נתונים או המודיע למשתמש שלא ניתן לבצע פעולה שבחר תוך ציון הסיבה לסירוב.
על כל שכבה לזרוק חריגה מתאימה.
במידה לא נמצאת כזו בתשתית הקיימת יש להגדיר חריגה מותאמת בחוזה הנתונים המתאים.
אין להשתמש בחריגה מסוג Exception בשום חלק של הפרויקט.

לסיכום: כל שכבה בודקת את מה שקשור אליה, ומעלה חריגה לשכבות מעל.
המידע של החריגה יכלול תיאור השגיאה וכן את השכבה ממנה נזרקה החריגה.
לדוגמא:
שכבת ה DAL תזרוק חריגות הקשורות לנתונים, כגון: ניסיון לעדכן יחידת אירוח שלא קיימת במערכת, או הכנסת מארח חדש עם תעודת זהות שכבר קיימת.
שכבת ה BL תתפוס את החריגות משכבת ה DAL ותטפל במה שהיא יכולה. אם אינה יכולה, החריגה תעלה הלאה לשכבת ה UI.
שכבת ה BL תזרוק חריגות לוגיות הקשורות לשכבה הזו. כגון: נסיון לקבוע אירוח לתאריך שכבר תפוס במערכת.
שכבת ה UI תטפל בחריגות מהשכבה BL. ותזרוק חריגות משלה על הנתונים שהזין המשתמש.

נספח 6
עבודה עם קבצי XML מקומיים

ניתן להיעזר בחומרים שנלמדו בהרצאה בנוגע XElement וכן ל XmlSerializer.

ראינו דוגמא לעבודה XmlSerializer עם בנספח 4. ניתן להיעזר.
בעבודה עם XElement עבור אתחול קבצי הXML:

יש ליצור עצם חדש עבור האלמנט הראשי של כל קובץ.
לדוגמה:

```
studentRoot = new XElement("students");
```

כעת נוכל כל פעם להוסיף \להוריד ולעדכן אלמנטים בקובץ
במידה והקובץ כבר קיים, פשוט לא ניצור אותו.
הקוד יהיה בערך כך:

```
public XmlSample()
 {
     if (!File.Exists(FPath))
         CreateFiles();
 }

 private void CreateFiles()
 {
     studentRoot = new XElement("students");
 }
```

עבור המספר הרץ של חלק מהמחלקות ועוד כמה הגדרות:

הבעיה:
כשמימשנו את המספר הרץ השתמשנו במשתנה סטטי שגדל במהלך התוכנית.
כעת שנשמור את הנתונים ב- xml בפעם הבאה שנפתח את התוכנית אותו משתנה יתאפס
שוב ואז המספר הרץ יתחיל מ- 0

הפתרון:
נגדיר קובץ XML בשם config.xml ושם נגדיר את המספר הרץ ועוד הגדרות שנרצה
כל פעם שנשמור אובייקט שמשתמש במספר הרץ נעדכן גם את קובץ ה config.xml
בפעם הראשונה שה dal נוצר הוא ידאג לעדכן זאת במחלקה.