# פרויקט ניהול חדרי כושר

# מסמך תכנון ועיצוב

## תוכן עניינים

[פרויקט ניהול חדרי כושר 1](#_Toc534132759)

[מסמך תכנון ועיצוב 1](#_Toc534132760)

[תוכן עניינים 2](#_Toc534132761)

[הקדמה לקריאת המסמך 3](#_Toc534132762)

[הצגת הצורך במערכת 3](#_Toc534132763)

[תיאור כללי של המערכת 3](#_Toc534132764)

[הנחות עבודה 3](#_Toc534132765)

[מוסכמות רישום 3](#_Toc534132766)

[מבנה בסיס הנתונים 5](#_Toc534132767)

[טבלת מתאמנים – Trainee 5](#_Toc534132768)

[טבלת מאמנים –Trainer 5](#_Toc534132769)

[טבלת מנויים – Subscription 6](#_Toc534132770)

[טבלת אימונים אישיים – PrivateTraining 6](#_Toc534132771)

[תרשים של ה-DB 6](#_Toc534132772)

[שכבת ה-BL (Business Logic Layer) – שכבת הלוגיקה 8](#_Toc534132773)

[דאטאבייס: 8](#_Toc534132774)

[ישויות: 9](#_Toc534132775)

[שכבת ה-GUI (Graphical User Interface) – שכבת ההצגה והממשק 10](#_Toc534132776)

[שימוש ברכיבי שפה ודפוסי עיצוב 10](#_Toc534132777)

[שיפורים ושינויים עתידיים 11](#_Toc534132778)

## הקדמה לקריאת המסמך

### הצגת הצורך במערכת

חדר הכושר M.T הוא ארגון למטרות רווח המנהל כמות מתאמנים רבה בחדר כשר עצמו באופן עצמאי ו\או עם מדריך מטעם החדר כושר ,לאימון אישי.

במסגרת מטרה זו יש לנהל כמות גדולה של מתאמנים ויש צורך ליצור מעקב אוטומטי אחרי המינויים כמו כן לאפשר שיבוץ נכון של מדריכים לפעילות השוטפת של החדר כושר וגם לאימונים אישיים.

כתוצאה מהמבנה הארגוני הנ"ל, על הארגון להקים ולתחזק מאגר מתאמנים גדול. על מנת שמאגר זה ישמש את הארגון בצורה מיטבית עליו להיות עדכני, מדויק ונוח לשימוש.

מטרת המערכת היא לאפשר בנייה וניהול של מאגר כזה תוך שימת דגש על איכות הנתונים הנשמרים והקישור בינם, נוחות השימוש במאגר ומיצוי מיטבי ויעיל של המידע המאוחסן - על מנת להפוך אותו לידע שישמש את הארגון במהלך פעילותו השוטפת.

### תיאור כללי של המערכת

המערכת הינה מערכת 2 שכבות הכתובה ב-C# על פלטפורמת .Net 4 ומשתמשת בבסיס נתונים מסוג Sql. מודל השכבות כולל חלוקה לשכבות הבאות:

* שכבת הלוגיקה (Logic Layer) שמופרדת לשתי שכבות:
  + שכבת הלוגיקה העסקית (Business Logic) שמגדירה את האובייקטים במערכת ומבצעת את מירב הלוגיקה
  + שכבת הגישה ל-DB (Data Access Layer) שמבצעת את כל העבודה מול ה-DB באופן בלעדי
* שכבת התצוגה (Presentation Layer)

כל שכבה מתקשרת עם שתי השכבות שמעליה ומתחתיה באמצעות ממשק קבוע. מודל זה מאפשר גמישות ועמידות לשינויים בעתיד כאשר ניתן בקלות יחסית להחליף את אחת השכבות בלי לפגוע בשכבות האחרות.

### הנחות עבודה

1. קיים רק משתמש אחד – המנהל
2. למנהל יש הרשאות לבצע כל דבר במערכת
3. יש Thread יחיד בלבד. לכן אין צורך לדאוג לבעיות סנכרון.
4. ישנו רק מחשב אחד היכול לגשת אל הממשק לניהול חדר כושר ספציפי.

### מוסכמות רישום

1. ENUM-ים במערכת יוגדרו עם הסיומת Enum. בנוסף, Enum-ים מתחילים באות גדולה
2. משתנים במחלקה יוגדרו עם תחילית של m\_
3. כל שם מחלקה, מתודה, enum, Property, תכתב בצורת UpperCamelCase
4. לממשקים נוספת האות I לפני שם המחלקה
5. TODO: add your stuff

## מבנה בסיס הנתונים

שכבת הDatabase היא שכבה מרכזית במוצר העוסק בניהול מידע, ועל כל פעולה שמבוצעת בממשק המשתמש של המוצר יש פעולה מקבילה לה שרצה על הDatabase. על כן, נבחר לממש פלטפורמת ORM קטנה, המנהלת את המידע.

בסיס הנתונים של המערכת מבוסס Sql היות שזהו DB שמותאם היטב לעבודה עם שפת C# ופלטפורמת ה-.Net כחלק מסל מוצרי Microsoft.

המערכת מבוססת כרטסות ולכן כל סוג של כרטסת נשמר בטבלה נפרדת. טבלאות אלו מקושרות לטבלאות מימד אשר מגדירות ערכים של פריטים מידע מורכבים יותר שכלולים בכרטסת. הקישור בין הכרטסות מתבצע ע"י טבלאות מקשרות בין הטבלאות המתאימות ב-DB.

כל טבלה מכילה כמובן מפתח ראשי ומפתחות זרים המקושרים לטבלאות אחרות, כאשר ה-DB אוכף קיום של קשרים אלו ובודק באופן אוטומטי את שלמותם ותקינותם.

נפרט כעת על הטבלאות השונות:

### טבלת מתאמנים – Trainee

* תפקיד הטבלה – טבלה זו שומרת את כל המידע הרלוונטי של מתאמן במערכת. כמו מידע אישי והתכתבות.
* מפתח ראשי – מספר תעודת הזהות של המתאמן משמש כמפתח ראשי ייחודי ומשמש לזיהוי של המתאמן בתוך ה-DB בטבלאות אחרות.
* שדות מרכזיים – מידע אישי (תאריך לידה, שם פרטי ושם משפחה, תמונה), פרטי התקשרות (טלפונים זמינים, כתובת ומייל), הערה כללית של טקסט חופשי וימי התאמנות.
* טבלאות מקושרות –
  + Subscription – המנויים של מתאמן.
  + PrivateTraining – רישומים לאימון אישי

### טבלת מאמנים –Trainer

* תפקיד הטבלה – טבלה זו שומרת את כל המידע הרלוונטי של מאמן במערכת. במידע זה כלולים כל הנתונים הנדרשים להצגה בכרטסת כמו שם, תמונה, וימי עבודה.
* מפתח ראשי – מספר תעודת הזהות של המאמן משמש כמפתח ראשי ייחודי ומשמש לזיהוי של המאמן בתוך ה-DB בטבלאות אחרות.
* שדות מרכזיים – מידע אישי (תאריך לידה, שם פרטי ושם משפחה, תמונה), פרטי התקשרות (טלפונים זמינים, כתובת ומייל), הערה כללית של טקסט חופשי וימי עבודה.
* טבלאות מקושרות –
  + PrivateTraining – אימון אישי למתאמן מסויים.

### טבלת מנויים – Subscription

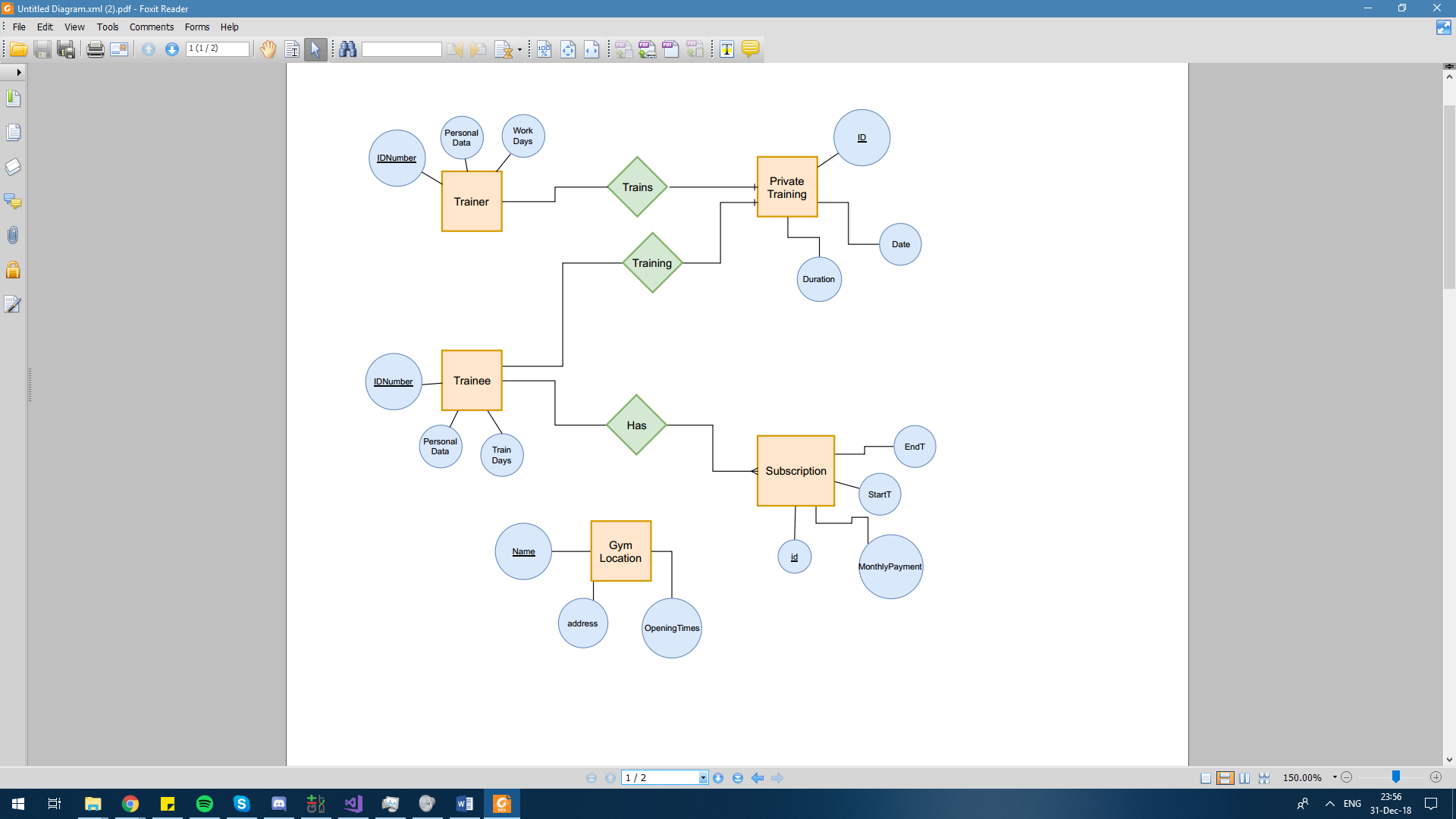
* תפקיד הטבלה – טבלה זו שומרת את כל המידע הרלוונטי של מנוי במערכת. במידע זה כלולים כל הנתונים הנדרשים להצגה בכרטסת.
* מפתח ראשי – מזהה מספרי אוטומטי שמוקצה ע"י ה-DB בצורה של מספר רץ ומשמש כמפתח ראשי ייחודי בטבלה. מזהה זה משמש לזיהוי של המנוי בתוך ה-DB במקרה של עדכון
* שדות מרכזיים – תאריך התחלת המנוי, תאריך סיום המנוי, מחיר חודשי והמתאמן.
* טבלאות מקושרות –
  + Trainee – המתאמן המנוי.

### טבלת אימונים אישיים – PrivateTraining

* תפקיד הטבלה – טבלה זו שומרת את כל המידע הרלוונטי של האימונים האישיים.
* מפתח ראשי – מזהה מספרי אוטומטי שמוקצה ע"י ה-DB בצורה של מספר רץ ומשמש כמפתח ראשי ייחודי בטבלה. מזהה זה משמש לזיהוי של המנוי בתוך ה-DB במקרה של עדכון
* שדות מרכזיים – המאמן, המתאמן, תאריך ומשך.
* טבלאות מקושרות –
  + Trainee – המתאמן שעובר את האימון האישי
  + Trainer – מדריך האימון האישי

### תרשים של ה-DB

להלן תרשים ERD של ה-DB:



# שכבת ה-BL (Business Logic Layer) – שכבת הלוגיקה

שכבה זו היא שכבת הבסיס, המהווה תשתית לוגית לאפליקציה הגראפית, שאינה תלויה בישויות גראפיות.

הסיבה המרכזית לעבודה בצורה זו היא עמידות לשינויים – הפרדה לוגית ותוכנתית בין התוכנה לגרפיקה מאפשרת להחליף בעתיד את הממשק היצוגי למשתמש (כמו למשל, CLI).

להלן פירוט המחלקות בשכבה זו:

### דאטאבייס:

#### Database.cs

* תיאור המחלקה – מחלקה זו עוטפת את ה-SqlConnection ומייצאת פונקציונליות לשמירה, עדכון, ושליפה על פי אובייקטים. זהו Singleton, מכיוון שישנו אך ורק מסד נתונים יחיד, ועל כן נייצגו באמצעות מופע יחיד של המחלקה.
* פונקציות עיקריות – שמירה, עדכון ושליפת אובייקטים.
* מערכות יחסים עם אובייקטים אחרים: עובד מול כל מחלקה המממשת את IDatabaseSerializable. משתמשת בDatabaseStream על מנת לעבוד מול ה IDatabaseSerializable.
* משתנים מרכזיים – אובייקט SqlConnection, ואובייקט liveObjects המייצג את האובייקט שכרגע נטענים, על מנת לפתור בעיות הצבעה מעגליות של אובייקטים.

#### IDatabaseSerializable.cs

* תיאור הממשק –זהו ממשק שמייצג אובייקטים שיכולים להיות מסורלזים לתוך המסד נתונים.
* פונקציות עיקריות – סריאליזציה לתוך stream וטעינה.
* יחסים עם מחלקות נוספות: הסריאליזציה תבוצע לתוך IDatabaseStream, והטעינה תשתמש בDatabase על מנת להשיג איברים נוספים.

#### IDatabaseSerializableWithId.cs

* תיאור המחלקה – זהו ממשק שמייצג אובייקט שניתן לסרלוז לתוך המסד נתונים, המיוצג על ידי id יחיד. הבדל בינו לבין קודמו הוא שמחלקה זאת ניתן לעדכן במסד נתונים בצורה פשוטה, שכן מעדכנים על פי הID.
* פונקציות עיקריות – השגת הID והגדרתו.
* יורש מ: IDatabaseSerializable

#### IDatabaseStream.cs

* תיאור המחלקה – זהו ממשק שמייצג stream לתוך המסד נתונים. מכיוון שישנם המון סוגי מסדי נתונים, השארתי את אופן הסריאליזציה פתוח לבחירת המשתמש באמצעות המחלקה הזאת. באמצעות מחלקה זאת ניתן לפרק אובייקטים מורכבים לאובייקטים פשוטים יותר.
* פונקציות עיקריות – הוספת עמודה וערך כלשהו, שניתן להמירו לערך בעמודה בצורה פשוטה.

### ישויות:

#### GymLocation.cs

* תיאור המחלקה - מייצג חדר כושר במיקום מסוים
* פונקציות עיקריות – יצוג המידע, ומימוש הממשק עבודה מול הדאטאבייס
* מממש – IDatabaseSerializableWithId
* משתנים מרכזיים: שם (המזהה היחודי), מיקום, ושעות פתיחה.
* עבודה עם מחלקות נוספות: Database, דרכו ניתן לעדכן ולטעון, למחוק ולשמור אובייקטים אל המסד נתונים

#### ObservableObject.cs

* TODO

#### Person.cs

* תיאור המחלקה – מחלקה אבסטרקטית המייצגת אדם כלשהו.
* פונקציות עיקריות – יצוג המידע, ומימוש הממשק עבודה מול הדאטאבייס
* מממש – IDatabaseSerializableWithId, ObservableObject
* משתנים מרכזיים: תעודת זהות (המזהה היחודי), שם פרטי, שם משפחה, כתובת, טלפון בבית, טלפון סלולרי, אימייל, תאריך לידה, והודעה טקסטואלית
* עבודה עם מחלקות נוספות: Database, דרכו ניתן לעדכן ולטעון, למחוק ולשמור אובייקטים אל המסד נתונים

#### PrivateTraining.cs

* תיאור המחלקה - מייצג אימון אישי.
* פונקציות עיקריות – יצוג המידע, ומימוש הממשק עבודה מול הדאטאבייס
* מממש – IDatabaseSerializableWithId
* משתנים מרכזיים: מזהה יחודי, המאמן, המתאמן, תאריך ומשך האימון.
* עבודה עם מחלקות נוספות: Database, דרכו ניתן לעדכן ולטעון, למחוק ולשמור אובייקטים אל המסד נתונים

#### Subscription.cs

* תיאור המחלקה - מייצג מנוי.
* פונקציות עיקריות – יצוג המידע, ומימוש הממשק עבודה מול המסד נתונים
* מממש – IDatabaseSerializableWithId
* משתנים מרכזיים: מזהה יחודי, המתאמן, תאריך התחלה, תאריך סיום, עלות חודשית, והאם פעיל,
* עבודה עם מחלקות נוספות: Database, דרכו ניתן לעדכן ולטעון, למחוק ולשמור אובייקטים אל המסד נתונים

#### TimeSpanOfWeek.cs

* תיאור המחלקה – מייצג טווח זמן, בחלק מסויים בשבוע
* פונקציות עיקריות – יצוג המידע, ומימוש הממשק עבודה מול המסד נתונים
* משתנים מרכזיים: היום בשבוע, זמן התחלה וזמן סיום
* עבודה עם מחלקות נוספות: Utils, כלי עזר דרכו ניתן להמיר אובייקט זה אל אובייקט המתאים למסד נתונים.

#### Trainee.cs

* תיאור המחלקה - מייצג מתאמן.
* פונקציות עיקריות – יצוג המידע, ומימוש הממשק עבודה מול המסד נתונים
* מממש – IDatabaseSerializableWithId
* משתנים מרכזיים: משתני האדם, בנוסף, המנויים שלו וימי האימון.
* עבודה עם מחלקות נוספות: Database, דרכו ניתן לעדכן ולטעון, למחוק ולשמור אובייקטים אל המסד נתונים
* יורש מ: Person

#### Trainer.cs

* תיאור המחלקה - מייצג מאמן.
* פונקציות עיקריות – יצוג המידע, ומימוש הממשק עבודה מול המסד נתונים
* מממש – IDatabaseSerializableWithId
* משתנים מרכזיים: משתני האדם וימי עבודה, שהם TimeSpanOfWeek.
* עבודה עם מחלקות נוספות: Database, דרכו ניתן לעדכן ולטעון, למחוק ולשמור אובייקטים אל המסד נתונים
* יורש מ: Person

### דוחות

#### NewSubscribersReport.cs

* תיאור המחלקה – מחלקה זו מאפשר השגת המתאמנים החדשים, בטווח תאריכים מסויים.
* פונקציות עיקריות – השגת המתאמנים החדשים והנגשתם.
* משתנים מרכזיים: המתאמנים החדשים
* עבודה עם מחלקות נוספות: Database, דרכה משיגה את המתאמנים החדשים. ומחלקת המתאמנים, כמובן.

#### PersonalTrainingReportForTrainers.cs

* תיאור המחלקה – מחלקה זו מגישה דוח על מאמן על האימונים האישיים שיש לו.
* פונקציות עיקריות – השגת האימונים האישיים והנגשתם.
* משתנים מרכזיים: האימונים האישיים
* עבודה עם מחלקות נוספות: Database, דרכה משיגה את האימוניים האישיים. וכמובן, האימונים האישיים, אותה היא מכילה.

#### SubscriptionNearEnd.cs

* תיאור המחלקה – מחלקה זו מגישה דוח על מנויים שעומדים להיגמר.
* פונקציות עיקריות – השגת המנויים והנגשתם.
* משתנים מרכזיים: המנויים.
* עבודה עם מחלקות נוספות: Database, דרכה משיגה את המנויים. וכמובן, המנויים.

# שכבת ה-GUI (Graphical User Interface) – שכבת ההצגה והממשק

תפקידה של שכבה זו הוא להציג את הנתונים בממשק משתמש נוח וברור, ולאפשר למשתמש לבצע את הפעולות השונות במערכת תוך חווית שימוש טובה.

כללי:

לכל ישות במערכת שהוגדרה בשכבת ה-BL מוגדר רכיב משתמש מתאים בשכבת ה-GUI, בדומה לכך גם עבור כל דו"ח שהוגדר בשכבת ה-BL מוגדר רכיב משתמש מתאים בשכבת ה-GUI.

שכבת הGUI נבנתה בטכנולוגית WPF ומשתמשת במנגנון ה-BINDING על מנת לתקשר עם המודל שלה

כל ישות ברמת ה-BL יורשת ממחלקה ObservableObject אשר מממש את INotifyPropertyChanged

ובכך מאפשר להשתמש באופן רציף במנגנון הBINDING של WPF בכך שבכל השמה קוראים לפונקציה OnPropertyChanged(string name)

למשל:

/// <summary>

/// The person's picture

/// </summary>

public byte[] Picture

{

get { return m\_Picture; }

set

{

OnPropertyChanged("Picture");

m\_Picture = value;

}

}

private byte[] m\_Picture;

המערכת בנויה בתצורה של Single Page Application - ז"א שכל המערכת נמצאת בדף הראשי והVIEW וMODEL שמיוצגים ע"י User Control בהם ה-קובץ ה XAML מייצג את ה-VIEW וקובץ הcs שנועד לcode behind שומש גם בתור הMODEL של אותו User Control (השימוש בUser Control יפורט בהמשך).

## המסך הראשי

MainWindow.xaml/.cs –

המסך הראשי משתמש בין היתר ב TabControl שם הוא דואג לניווט רציף בין הUserControls

<TabControl Grid.Column="0" FlowDirection="RightToLeft" Grid.ColumnSpan="100" HorizontalAlignment="Stretch" Grid.Row="2" Grid.RowSpan="100" VerticalAlignment="Stretch" >

<TabItem Header="מתאמנים">

<ContentControl Content="{Binding CurrentTraineeUC}"></ContentControl>

</TabItem>

<TabItem Header="מאמנים">

<ContentControl Content="{Binding CurrentTrainerUC}"></ContentControl>

</TabItem>

<TabItem Header="דוחות">

<ContentControl Content="{Binding CurrentReportUC}"></ContentControl>

</TabItem>

<TabItem Header="אימונים אישיים">

<ContentControl Content="{Binding CurrentPrivateTrainingUC}"></ContentControl>

</TabItem>

</TabControl>

כל אחד מהטאבים מכיל BINDING ל content שלו CurrentPrivateTrainingUC

לUC (לשם נוחות UC קיצור לuserControl

ומתחלף לפי הצורך למשל CurrentTraineeUC = new NewTraineeUC();

ינווט את הטאב של מאמן לתצוגה של הוספת מאמן חדש.

אירועים במסך הראשי(Routed Events):

המסך הראשי מאזין לכל האירועים החיצוניים שמצריכים אותו להתערב (ניווט וכו')

#region Trainers Events Regs

AddHandler(TrainerUC.NewTrainertEvent,

new RoutedEventHandler(NewTrainertEvent\_handler));

AddHandler(NewTrainerUC.NavToTrainerRetriveEvent,

new RoutedEventHandler(NavToTrainerRetriveEvent\_handler));

AddHandler(TrainerFullView.NavToTrainerRetriveEvent,

new RoutedEventHandler(NavToTrainerRetriveEvent\_handler));

AddHandler(TrainerUC.ViewTrainerEvent,

new RoutedEventHandler(ViewTrainerEvent\_handler));

#endregion

#region Trainee Event reg

AddHandler(TraineeUC.NewTraineetEvent,

new RoutedEventHandler(NewTraineetEvent\_handler));

AddHandler(NewTraineeUC.NavToTraineeRetriveEvent,

new RoutedEventHandler(NavToTraineeRetriveEvent\_handler));

AddHandler(TraineeUC.ViewTraineeEvent,

new RoutedEventHandler(ViewTraineeEvent\_handler));

AddHandler(TraineeFullView.NavToPrivateTraining,

new RoutedEventHandler(NavToPrivateTraining\_handler));

AddHandler(TraineeFullView.NavToTraineeRetriveEvent,

new RoutedEventHandler(NavToTraineeRetriveEvent\_handler));

AddHandler(PrivateTrainingUC.NavToFullViewTrainerEvent,

new RoutedEventHandler(ViewTraineeEvent\_handler));

AddHandler(PrivateTrainingUC.NavToTraineeRetriveEvent,

new RoutedEventHandler(NavToTraineeRetriveEvent\_handler));

#endregion

וכך בעזרת ה handlerים המוגדרים מבצע ה.main פעולות שUC הפנימי דורש

דוגמאות UC –

PrivateTrainingsViewUC.xaml

**שימוש:** תצוגה וטיפול של אימונים אישיים

**ירושה:** UserControl, INotifyPropertyChanged

**אירועים:** אין

ReportsMenuUC.xaml

**שימוש:** שליפת דוחות

**ירושה:** UserControl, INotifyPropertyChanged

**אירועים:** אין

TraineeUC.xaml

**שימוש:** מסך שליפה של מתאמנים וניווט למסכי ישות מתאמן(עדכון, חדש וכו')

**ירושה:** UserControl, INotifyPropertyChanged

**אירועים:** ViewTraineeEvent**,**  NewTraineetEvent

מחלקה ExcelExportUtils – מחלקה סטטית אחראית על הצגת נתונים מרכיב DataGrid לאקסל.

מחלקה ImageUtils - מחלקת סטטית אחראית על טעינה מקובץ והמרה ל byte[] (כך נשמר בDB)

Converters:

DayOfWeekConverter – מחלקה שמממשת את IValueConverter ומשמשת את הXAML של UC של המאמנים בתצוגה נוחה יותר של היום בשבוע – ממיר מenum שמייצג את יום בשבוע – לשפה העברית.

ByteArrayToImageSourceConverter – ממש את IValueConverter ממירה את byte[] ל BitmapImage ובכך מאפשר לאלמנט Image שנמצא בXAMLים השונים ומאפשר להציג את התמונה מ-MODEL לVIEW

# שימוש ברכיבי שפה ודפוסי עיצוב

במערכת נעשה שימוש במספר Design Patterns:

* Singleton – במערכת שלנו אנחנו משתמשים בתבנית העיצוב Singleton על מנת לייצג את הDatabase.
* Visitor Pattern - הIDatabaseStream הוא בעצם visitor שמבקר בכל השדות של המחלקה. דרכו מסרלזים אובייקטים אל תוך מסד הנתונים.
* Factory – הDatabase הוא בעצם Factory שמייצר אובייקטים חדשים, באמצעות נתונים הקיים במסד הנתונים.

בנוסף לדפוסי העיצוב שנמצאים בשימוש באין לידי ביטוי גם רכיבים ויכולות של השפה:

* שימוש בטמפלייטים – במחלקת הDatabase ניתן להעביר כל מחלקה אשר מממשת IDatabaseSerializable, וישנה התייחסות לטיפוס בעת יצרתו ובעת שימוש בשם המחלקה בתור טבלת הנתונים.
* שימוש בפולימורפיזם – למשל, הIDatabaseSerializable תלוי רק בIDatabaseStream, אותו מממש המקרה הפרטי DatabaseSerializable.
* שימוש בממשקים (Interface) – כמו לדוגמא IDatabaseStream וIDatabaseSerializable
* העמסת מתודות – למשל מתודות ToString של אובייקטי הישויות השונים
* TODO: להרחיב

# שיפורים ושינויים עתידיים

* הוספת נתונים נוספים על המתאמנים – ישנו מידע נוסף על המתאמנים שניתן לשמור, אך כרגע לא מצאנו לו ביקוש רב בקרב הלקוחות. מידע נוסף זה כולל משקל, מין שעליו ניתן לבצע סטטיסטיקות מעניינות, ומידע זה עלול גם להשתנות לאורך זמן.
* היסטוריית כניסה לחדר כושר – לראות מי באמת מגיע לחדר כושר. בין אם זה מתאמנים, שמנצלים את אפשרותם להכנס לחדר כושר, או שמא מאמנים, שעובדים ודופקים שעון.
* תוכנית אימונים – ליצור למתאמנים תוכנית אימונים כלשהי המתארת ימים בשבוע שצריך להגיע בהם, אילו מכשירים להפעיל, וכדומה. בנוסף לתוכנית אימונים, שיהיה ניתן לעקוב אחרי התוכנית ולוודא שהיא באמת מתקיימת.
* הוספת דו"חות למערכת – סביר להניח שלאחר שימוש במערכת יעלו דרישות לדו"חות נוספים שמציגים מימדים אחרים של המידע. מכיוון שהעבודה מול המסד נתונים נוחה מאוד, אין סיבה הנראת לעין שתמנע להוסיף דוחות נוספים המשתמשים במידע הקיים במערכת.
* ניהול הרשאות – שכל מתאמן, מאמן ומנהל יהיו בעצם משתמשים שיכולים להתחבר למערכת בצורה נפרדת, ולכל אחד יהיה סט של הרשאות מסויימות.
* דוחות אוטומטיים – שיהיה ניתן ליצור מערכת של דוחות כל כמה זמן, שיווצר וישלח לאימייל מסויים (כמו למשל, דוחות חודשיים – אילו מנויים נרשמו וכדומה)