# **שכבת ההצגה**

שכבה זו מכילה את הממשק למשתמש שמכיל את המסכים שבהם המשתמש יכול לצפות, ולבצע אינטראקציה. השכבה גם מכילה validations בצד לקוח, פניות לשכבת הBL לקבלת שירותים, שמירה על מצב המשתמש המחובר כעת, קבלת התראות משכבת הBL והצגתן, ועוד...

**עקרונות מנחים בשכבת ההצגה:**

* שכבת ההצגה ממומשת באמצעות MVVM (Model-View-ViewModel) כך שיש הפרדה מלאה בין המסכים (views), הלוגיקה של המסכים (view models) והמחלקות מידע (models). כך במידת הצורך בעתיד, ניתן בקלות לשנות את המסכים מבלי לשנות את הלוגיקה של המסכים, ולהפך.
* קישור הView לViewModels יעשה דרך DataTemplates, והצגת המסכים יעשו באמצעות ContentControl שיכיל את הViewModel. באופן זה, בקוד אין צורך להתעסק כלל בViews, והView המתאים יוצג באופן אוטומטי כאשר נבחר בקוד להציג את הViewModel שלו.
* בכל פעולה על המסך יהיה שימוש בICommand. במידת הצורך אם יותר ממסך אחד צריכים את אותו ICommand יהיה ניתן לשתף בין המסכים את אותו Command.
* שימוש בDesign Patterns במקומות רלוונטיים, שישמרו על עיצוב השכבה גמיש לשינויים.

רשימת המחלקות בשכבה זו:

**מודלים**

## AbstractPost

מחלקה אבסטרקטית המתארת פוסט כלשהו במערכת. כמו כן המחלקה מממשת את INotifyPropertyChanged כך שכל שינוי בתכונות המחלקה יגרמו לשינוי מתאים במסך המוצג למשתמש.

public DateTime DatePosted

public int VotesCount

public string Author

public string Content

public event PropertyChangedEventHandler PropertyChanged;

## DelegateCommand

מממשת את ICommand. זהו מימוש של ICommand שמקבל שני delegates אחד עבור CanExecute ואחד עבור Execute (שהם חלק מהInterface של ICommand). כאשר wpf משתמש בCanExecute כדי להחליט אם ניתן להפעיל את הCommand או לא.

public event EventHandler CanExecuteChanged;

public bool CanExecute(object parameter)

public void Execute(object parameter)

public void RaiseCanExecuteChanged()

## AnswerModel

מחלקה המייצגת תשובה לשאלה במערכת, היודעת לעדכן את המסך במקרה ויש שינוי של אחד הערכים בה. מחלקה זו יורשת מAbstractPost. ניתן לצפות במודלים מסוג Answer במסכים שמראים Discussion Thread שלם, כלומר, שאלה, והתשובות שכתבו עבורה או בהזנה של תשובה חדשה לשאלה. כמו כן, ניתן להמיר כל מופע של מחלקה זו במופע של המחלקה ששאר המערכת מכירה (מחלקה בCommon).

public bool Answered

public Answer ToExternal()

## QuestionInfo

מחלקה המייצגת מידע תצוגתי של מידע על שאלה במערכת. יש לשים לב שלמחלקה זו אין את כל המידע על השאלה (למשל אין את תוכן השאלה). זאת מאחר שנעשה שימוש במודל זה רק עבור המסך תוצאות חיפוש – שבו מוצג רק חלק מהמידע על השאלה בכל שורה מבין התוצאות חיפוש. כמו כן המחלקה מממשת את INotifyPropertyChanged כדי להודיע למסך על עדכונים.

public ObservableCollection<string> Tags

public bool IsRecommended

public string Author

public string Title

public int VotesCount

public int AnswersCount

public Question ToExternal()

public event PropertyChangedEventHandler PropertyChanged;

## QuestionModel

מחלקה המייצגת מידע תצוגתי של שאלה במערכת. במחלקה זו משתמשים כדי להזין שאלה חדשה למערכת, וכדי לצפות בDiscussion Thread שלם – בו יש שאלה ותשובות שהוזנו עבורה. כמו כן, מחלקה זו מממשת את INotifyPropertyChanged כדי ששינויים בכל מופע של המחלקה ייראו על המסך. כמו כן ניתן להמיר אותה למחלקה שכל השכבות מכירות DiscussionThread.

public int AnswerCount

public ObservableCollection<AnswerModel> Answers

public ObservableCollection<string> Tags

public bool Recommended

public string Title

public DiscussionThread ToExternal()

## UserAccountModel

מחלקה זו משמשת למידע תצוגתי של משתמש במערכת. השימוש במחלקה זו הוא ביצירת משתמש חדש, ובמסך ניהול משתמשים אליו רכז ההוראה והמרצים יכולים לגשת. גם מחלקה זו מממשת את INotifyPropertyChanged וניתן להמיר אותה למחלקה שכל השכבות מכירות UserInfo.

public string Username

public int Ranking

public bool IsAdmin

public bool IsMuted

public UserInfo ToExternal()

public event PropertyChangedEventHandler PropertyChanged;

## NotificationsObserver

מחלקה זו פונה באופן פריודי לBL, ומתשאלת על התראות חדשות שמותאמות למשתמש שכרגע מחובר. במידה ויש התראות חדשות, יוקפץ event מתאים שיודיע על כך, עם ההתראות החדשות.

public event EventHandler<NewNotificationsEventArgs> NewNotifications;

## NewNotificationsEventArgs

מחלקת המידע שבה משתמשים באירוע של קבלת התראות חדשות.

הNotificationsObserver יוצר מופע של מחלקה זו בכל פעם שזיהה שיש התראות חדשות.

public IEnumerable<Notification> NewNotifications { get; private set; }

## AuthenticationTokenSingleton

מחלקה זו שומרת על הAuthentication Token שחזר לאחר תהליך הLog-in של המשתמש (אם בוצע), כך שכל מחלקה שרוצה לפנות לשכבת הBL וצריכה את הAuthentication Token על מנת להזדהות, תוכל לקחת אותו ממקום אחד בקלות.

המחלקה הזו מממשת את הDesign Pattern של Singleton מאחר ויכול בכל רגע נתון להיות רק משתמש אחד מחובר בכל עמדת קצה. לכן לא יכולה להיות במערכת שני Authentication Tokens שונים בו זמנית.

public static AuthenticationTokenSingleton Instance

public AuthenticationToken AuthenticationToken { get; set; }

## DiscussionThreadObserver

מחלקה זו מקשיבה לשינויים שקורים בDiscussion-Thread מסויים (שינוי בשאלה הנישאלת\דירוג שלה\סימון שלה כמומלצת או כל שינוי אחר באחד מהתשובות שהוזנו לשאלה) ומודיעה על כך שיש גרסה חדשה של הDiscussion Thread, וכך בעצם המסך יכול להתעדכן אוטומטית כשקורה איזהשהו שינוי בDiscussion Thread שעכשיו צופה בו המשתמש.

public event EventHandler<EventArgs> NewDiscussionThreadVersion;

public DiscussionThread LatestDiscussionThread { get; set; }

public DiscussionThread StartObservingDiscussionThread(int questionId)

public void Dispose()

## NewDiscussionThreadVersionEventArgs

מחלקת המידע שבה משתמשים באירוע שבו יש גרסה חדשה של הDiscussion Thread שנצפה כרגע.

הDiscussionThreadObserver יוצר מופע חדש של מחלקה זו בכל פעם שיש שינוי בDiscussionThread.

public DiscussionThread NewDiscussionThread { get; set; }

## NavigationManager

מחלקה זו מודיעה על בקשה של מעבר למסך אחר.

כל הViewModels שמהם ניתן לנווט למסך אחר, יכירו את מחלקה זו, ובכל פעם שהViewModel רוצה להחליף מסך, הוא פונה לNavigationManager עם הViewModel שהוא רוצה לעבור אליו. לאחר מכן הNavigationManager מעלה event עם בקשת השינוי, המסך הראשי מקשיב לאירוע זה, ומחליף את המסך הראשי בהתאם.

public void RequestNavigation(object viewModel);

event EventHandler<NavigationRequestedEventArgs> NavigationRequested;

## NavigationRequestedEventArgs

מחלקת המידע שמכילה את הViewModel אליו רוצים לבצע navigation.

בכל פעם שהNavigationManager רוצה להודיע על navigation הוא יוצר מופע של מחלקה זו.

public object ViewModelToNavigate { get; set; }

## App.xaml

הxaml הראשי של שכבת ההצגה למשתמש.

קובץ זה מציין מי החלון הראשון שנפתח בעליית התכנית, וכמו כן מגדיר DataTemplates שמקשרים בין כל הViews לViewModels שלהם, באופן כזה שבכל פעם שנרצה להציג ViewModel מסויים על המסך, באופן אוטומטי אנחנו נראה את הView שלו, והDataContext של הView יהיה הViewModel המתאים לו שרצינו "להציג".

## MainWindow

החלון הראשי של המערכת.

החלון מחלק את המסך למספר חלקים שבכל אחד מהם יוצג View אחר.

הMainMenu וMainToolbar הם כפי שניתן לראות במסמך אפיון ומדריך למשתמש. הNotificationsViewModel זה הViewModel שיציג התראות למשתמש על פעולות שהתבצעו שיעניינו את אותו משתמש, והMainVM זהו ViewModel שמתחלף כל הזמן בהתאם לנוויגציה של המשתמש במערכת – כלומר זה החלק היחיד שמשתנה במסך הראשי, כאשר המשתמש רוצה להירשם למערכת החלק הזה יכיל את הViewModel של הרישום, כאשר המשתמש צופה בשאלה, חלק זה יכיל את הDiscussion Thread שהוא צופה בו וכו...

public object MainVM

public NotificationsViewModel NotificationsVM

public MainMenuViewModel MainMenuVM

public MainToolbarViewModel MainToolbarVM

## MainMenu.xaml

View זה יכיל את החלק הכפתורי נוויגציה למסכים הראשיים. ניתן לראות את View זה במדריך למשתמש.

## MainMenuViewModel

זהו הViewModel שמתאים לMainMenu. הViewModel יכיל את כל הCommands שיתבצעו בלחיצה על הכפתור הרלוונטי בMainMenu.

public bool IsUserAdmin { get; set; }

public ICommand NavigateHome { get; set; }

public ICommand NavigateMyQuestions { get; set; }

public ICommand NavigateNewQuestion { get; set; }

public ICommand NavigateUserManagement { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

## MainToolbar.xaml

זהו הView של הToolbar העליון שמכיל את כפתור הLogin, Registration ואת החיפוש המהיר.

## MainToolbarViewModel

זהו הViewModel של הMainToolbar. הוא מכיל את כל הCommands של הפעולות האפשריות מהToolbar, ואת הטקסט של החיפוש החופשי.

public string SearchText { get; set; }

public ICommand Search { get; set; }

public ICommand Login { get; set; }

public ICommand Register { get; set; }

public ICommand Logout { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

## NewAnswer.xaml

זהו המסך המוצג בעת כתיבה של תשובה חדשה לשאלה.

ניתן לראות את מסך זה במדריך למשתמש.

## NewAnswerViewModel

זהו הViewModel של מסך כתיבת התשובה החדשה.

יש בו את הCommand של פרסום התשובה החדשה, ויש בו את הModel שמחזיק את הפרטים של התשובה החדשה שהמשתמש מזין.

public ICommand PublishAnswer { get; set; }

public AnswerModel Answer

public event PropertyChangedEventHandler PropertyChanged;

## NewQuestion.xaml

זהו המסך המוצג בעת כתיבה של שאלה חדשה לשאלה.

ניתן לראות את מסך זה במדריך למשתמש.

## NewQuestionViewModel

זהו הViewModel של מסך כתיבת השאלה החדשה.

יש בו את הCommand של פרסום השאלה החדשה, ויש בו את הModel שמחזיק את הפרטים של השאלה החדשה שהמשתמש מזין.

public ICommand PublishQuestion { get; set; }

public QuestionModel Question { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

## QuestionList.xaml

זהו הView שמכיל רשימה של שאלות (ללא תשובותיהם).

View זה מוצג כאשר רוצים להציג תוצאות של חיפוש, או כאשר מסתכלים על "השאלות שלי" (השאלות שהמשתמש הזין) על ידי לחיצה בMainMenu.

במסך יש ItemsControl שמגדיר ItemTemplate לכל אחד מהשאלות שהוא מציג, כך שבאופן אוטומטי, כל שאלה שמצויינת בViewModel תוצג על ידי ItemTemplate זה.

## QuestionListViewModel

זהו הViewModel של QuestionList. ViewModel זה מחזיק את רשימת השאלה שברצוננו להציג למשתמש (כתוצאות של חיפוש או כאשר המשתמש צופה ב"שאלות שלי"). כמו כן הViewModel מכיל Command של צפייה בDiscussionThread ספציפי על ידי לחיצה על אחד השאלות במסך QuestionList – השאלה שנלחצה תעבור כפרמטר לCommand של NavigateToQuestion.

public ICommand NavigateToQuestion { get; set; }

public ObservableCollection<QuestionInfo> Questions

public event PropertyChangedEventHandler PropertyChanged;

## QuestionView.xaml

החלק התצוגתי של שאלה – בתוך מסך צפייה בשאלה תשובותיה (Discussion Thread). הView הזה יופיע לשאלה הראשית שמוצגת כרגע במסך.

## QuestionViewModel

זהו הViewModel של השאלה (היחידה) שנצפית כרגע במסך.

לViewModel זה יש properties שיהיה ניתן לעשות אליהם Binding מהView.

זה כולל את הModel (QuestionModel) וCommands שניתן להפעיל בלחיצה על כפתור בView כמו דירוג השאלה לחיוב\שלילה.

כמו כן הViewModel מממש את INotifyPropertyChanged כך שבכל שינוי שלו, הView יתעדכן אוטומטית בהתאם.

public QuestionModel Question

public ICommand RankUp { get; set; }

public ICommand RankDown { get; set; }

public ICommand Delete { get; set; }

public ICommand AddAnswer { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

## AnswerView.xaml

החלק התצוגתי של תשובה – בתוך מסך צפייה בשאלה ותשובותיה (Discussion thread). לכל תשובה שמוצגת על המסך יהיה מופע של הView הזה.

## AnswerViewModel

הViewModel של תשובה שנצפית על המסך.

לViewModel יש properties שיהיה ניתן לעשות אליהם Binding מהView.

זה כולל את הModel (AnswerModel) וCommands שניתן להפעיל בלחיצה על כפתור בView כמו דירוג התשובה לחיוב\שלילה, או מחיקת התשובה.

כמו כן הViewModel מממש את INotifyPropertyChanged כך שבכל שינוי שלו, הView יתעדכן אוטומטית בהתאם.

public AnswerModel Answer

public ICommand RankUp { get; set; }

public ICommand RankDown { get; set; }

public ICommand Delete { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

## RegistrationControl.xaml

מסך זה הוא מסך הרישום למערכת.

במסך זה מזינים את כל האינפורמציה הדרושה כדי להירשם למערכת.

ניתן לראות את מסך זה במדריך למשתמש.

## RegistrationViewModel

הViewModel של מסך הרישום למערכת.

מסך זה מכיל את הCommand של רישום המשתמש לאחר הזנת הפרטים – שיוודא שכל הפרטים שהוזנו הם תקינים על ידי שימוש בValidations, ולאחר מכן ישמור את המשתמש החדש במערכת.

הUser והUserCredentials יתמלאו על ידי המשתמש כאשר ימלא את השדות הרלוונטים במסך.

public UserInfo User { get; set; }

public UserCredentials Credentials { get; set; }

public ICommand RegisterUser { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

## UserManagementView.xaml

זהו מסך ניהול המשתמשים שאליו רק מרצים ורכז ההוראה יכולים לגשת.

במסך זה ניתן להשתיק משתמשים, להפוך משתמש למרצה, ועוד...

ניתן לראות את מסך זה במדריך למשתמש.

## UserManagementViewModel

זהו הViewModel של מסך הניהול למשתמש. הViewModel מכיל בObservableCollection את כל המשתמשים הקיימים במערכת, ואת הCommand של שמירת השינויים שהתבצעו במסך.

public ObservableCollection<UserAccountModel> Users

public ICommand ApplyChanges { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

## NotificationsView.xaml

הView של הצגת התראות למשתמש. הView יכיל את כל ההתראות האחרונות שרלוונטיות למשתמש המחובר.

## NotificationsViewModel

הViewModel של המצגת התראות למשתמש.

ViewModel זה יכיל בObservableCollection את כל ההתראות האחרונות שיוצגו על המסך. כמו כן הViewModel ייצור מופע של הNotificationsObserver כך שיוכל לדעת על התראות חדשות משכבת הBL ולהציג אותם למשתמש על ידי עדכון הproperty: Notifications

public ObservableCollection<Notification> Notifications;

## Styles.xaml

מכיל Styleים שונים לControlים בכל שכבת הPL.