# **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

public static AuthenticationTokenSingleton Instance

public AuthenticationToken AuthenticationToken { get; set; }

## DiscussionThreadObserver

public event EventHandler<EventArgs> NewDiscussionThreadVersion;

public DiscussionThread LatestDiscussionThread { get; set; }

public DiscussionThread StartObservingDiscussionThread(int questionId)

public void Dispose()

## NewDiscussionThreadVersionEventArgs

public DiscussionThread NewDiscussionThread { get; set; }

## NavigationManager

event EventHandler<NavigationRequestedEventArgs> NavigationRequested;

## NavigationRequestedEventArgs

public object ViewModelToNavigate { get; set; }

## App.xaml

## MainWindow

public object MainVM

public NotificationsViewModel NotificationsVM

public MainMenuViewModel MainMenuVM

public MainToolbarViewModel MainToolbarVM

## AnswerView.xaml

## AnswerViewModel

public AnswerModel Answer

public ICommand RankUp { get; set; }

public ICommand RankDown { get; set; }

public ICommand Delete { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

## MainMenu.xaml

## MainMenuViewModel

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

## MainToolbarViewModel

public string SearchText { get; set; }

public ICommand Search { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

## NewAnswer.xaml

## NewAnswerViewModel

public ICommand PublishAnswer { get; set; }

public AnswerModel Answer

public event PropertyChangedEventHandler PropertyChanged;

## NewQuestion.xaml

## NewQuestionViewModel

public ICommand PublishQuestion { get; set; }

public QuestionModel Question { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

## QuestionList.xaml

## QuestionListViewModel

public ICommand NavigateToQuestion { get; set; }

public ObservableCollection<QuestionInfo> Questions

public event PropertyChangedEventHandler PropertyChanged;

## QuestionView.xaml

## QuestionViewModel

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;

## RegistrationControl.xaml

## RegistrationViewModel

public UserInfo User { get; set; }

public UserCredentials Credentials { get; set; }

public ICommand RegisterUser { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

## UserManagementView.xaml

## UserManagementViewModel

public ObservableCollection<UserAccountModel> Users

public ICommand ApplyChanges { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

## NotificationsView.xaml

## NotificationsViewModel

public ObservableCollection<Notification> Notifications;

## Styles.xaml