# **שכבת הלוגיקה**

שכבה המקשרת בין שכבת ההצגה למשתמש לשכבת הגישה לנתונים. כלפי שכבת ההצגה למשתמש, היא נותנת ממשקים אליהם ניתן לפנות על מנת לבצע פעולות שונות ולקבל מידע. כלפי שכבת בסיס הנתונים, זו השכבה שמבצעת את הפניות אליו כדי לשמור, לעדכן ולמחוק נתונים וכן, לקבל נתונים מבסיס הנתונים.

בשכבת הלוגיקה יתבצעו פעולות שונות שיבטיחו שמצד אחד המידע שנשמר בבסיס הנתונים הוא תקין ובפורמט המתאים, ומצד שני יבטיח שהמידע שמגיע מבסיס הנתונים לשכבת ההצגה למשתמש יגיע לאחר סינונים שונים ובהתאם למה שרלוונטי לשכבת ההצגה.

שכבת הלוגיקה שלנו חושפת את השירותים שלה לשכבת התצוגה למשתמש על ידי טכנולוגיית WCF.

**עקרונות מנחים בשכבת הלוגיקה:**

* אימות נתונים - שכבת הלוגיקה תספק אימות (VALIDATIONS) על הנתונים שמתקבלים משכבת ההצגה למשתמש בביצוע פעולות.
* אימות משתמש – שכבת הלוגיקה תזהה את המשתמש בכל כניסה למערכת, ותנפיק למשתמש Token אותו הוא יצטרך להציג בביצוע כל פעולה כדי להזדהות.
* ניהול הרשאות – שכבת הלוגיקה תמנע ממשתמש לבצע פעולה שהוא לא מורשה אליה, למשל – משתמש שאיננו מרצה או מרכז, שמנסה להשתיק משתמש אחר. המטרה היא למנוע אפשרות של עקיפת שכבת ההצגה למשתמש על ידי כלים חיצוניים.
* מימוש לוגיקות שונות בהתאם לאפיון המערכת. למשל – דירוג של שאלה\תשובה משנה גם את דירוג הכותב שלה.
* עקרון הכימוס בא לידי ביטוי בשכבת הלוגיקה בכמה אופנים. קודם כל, כמעט כל שתי מחלקות ייגשו אחד לשני רק באמצעות interfaceים, כך שניתן בקלות להחליף את השימוש של מחלקה מסוימת למחלקה אחרת ללא שינויי קוד רבים וללא תלות במימוש הפנימי של המחלקה.
* Dependency Injection – כל מחלקה תקבל את הממשקים עליהן יש תלות בConstructor. כך בקלות ניתן להחליף מימושים של הממשקים בלי שינוי של קוד באותה מחלקה, וכמעט ללא שינוי קוד. כמו כן, יהיה קל להוסיף Unit Tests בשלב יותר מאוחר של הפרויקט – כמובן שלא נוסיף אותם במסגרת פרויקט זה.

# **מחלקות כלליות**

## AuthTokenCache

מחלקה זו משמשת כCache של Authentication tokens כדי לזרז את האימות של פעולת משתמש.

המחלקה תאפשר לנו לא לפנות לDAL בכל פעם שמשתמש מנסה לבצע פעולה ומזדהה עם הauthentication token שלו.

אם הAuthentication token שנבדק לא נמצא בCache, נוכל להפיק מהAuthentication Token את הCredentials של המשתמש ולאמת את זהותו בפניה לDAL. אם זהותו אומתה, נוסיף את הToken לCache.

הCache משתמש בConcurrentBag על מנת שיהיה קל לשלוף ולהכניס אליו Tokens באופן אסינכרוני באופן שהוא Thread-Safe.

ConcurrentBag<string> TokenCache

## IAuthTokenSerializer

ממשק שנותן ליצור Authentication Token מUserCredentials ולהפך.

AuthenticationToken Serialize(UserCredentials userCredentials)

UserCredentials Deserialize(AuthenticationToken authToken)

## AuthTokenSerializer

מחלקה זו מייצרת Authentication Token מUserCredentials ולהפך. מחלקה זו יורשת מIAuthTokenSerializer. כאשר מתקבלים UserCredentials של משתמש, אנחנו רוצים להחזיר למשתמש AuthenticationToken שמתאים ספציפית לUserCredentials שהוא הזין (ואומתו בהצלחה) כך שבכל פעולה שהוא יבצע הוא יעביר את הAuthenticationToken וכך הBL יוכל לזהות את המשתמש בהצלחה ללא צורך בבדיקת הCredentials מול הDB. מה שיכול למהר מאוד את תהליך אימות המשתמש בכל פעולה. כמו כן ניתן להרחיב את הAuthenticationToken כן שהוא יהיה תקף רק לפרק זמן מוגבל, וכך במידה והAuthenticationToken נגנב על ידי פורץ, הToken יוכל להיות בשימוש רק לפרק זמן מוגבל. במימוש אמיתי של המערכת שמעבר לגבולות הפרויקט, היה ניתן גם לייצר את הAuthenticationToken מהIP Address של המשתמש וגם מהUserCredentials שלו, ובכל שלב שמתקבל Token, לאמת אותו מול הClient IP Address, כך שגם אם הToken נגנב, לא ניתן להשתמש בו ממחשב אחר.

public AuthenticationToken Serialize(UserCredentials userCredentials)

public UserCredentials Deserialize(AuthenticationToken authToken)

## INotificationAggregator

ממשק שמשמש לאיחוד התראות דומות לכדי רשימה מצומצמת של התראות.

IEnumerable<Notification> AggregateNotifications(IEnumerable<Notification> notifications)

## NotificationAggregator

מחלקה זו משתמשת לאיחוד התראות דומות לכדי רשימה מצומצמת של התראות. המחלקה יורשת מINotificationAggregator.

הפעולה הראשית של המחלקה מקבלת רשימה של התראות שונות – כך שיש התראה אחת לכל פעולה למשל, אם היו 3 תשובות חדשות שנכתבו לשאלה שמשתמש מסוים שאל, אז היינו רוצים לאחד את שלושת ההתראות האלה של שלושת התשובות החדשות לכדי התראה בודדה שתכיל את הטקסט "שלוש תשובות חדשות הוזנו לשאלה ששאלת" במקום שהמשתמש יראה 3 הודעות שונות של "תשובה חדשה הוזנה לשאלה ששאלת".

public IEnumerable<Notification> AggregateNotifications(IEnumerable<Notification> notifications)

## NotificationService

מחלקה זו מספקת שירות לשכבת התצוגה שנותן לה את כל ההתראות על פעולות שהתרחשו שהמשתמש עדיין לא מודע אליהם.

הפעולה הראשית מקבל Authentication Token שבאמצעותו ניתן לזהות את המשתמש שמבקש לקבל את ההתראות החדשות שלו. המחלקה תבדוק מול שכבת הDAL איזה התראות עדיין לא "נמשכו", ואז תשתמש בNotificationsAggregator כדי לאחד התראות דומות, ותחזיר את התוצאה למשתמש.

public IEnumerable<Notification> GetNotifications(AuthenticationToken authToken)

## PostQueries

מחלקה זו מספקת שירותי שאילתות על שאלות ותשובות שהוזנו במערכת.

המחלקה יורשת מממשק מתאים שחושף אותה כשירות WCF לשכבת ההצגה למשתמש.

כל הפעולות במחלקה זו מקבלות AuthenticationToken או AuthenticatedOperation כפרמטר על מנת שיהיה אפשר לזהות את המשתמש שמקבל לבצע את הפעולה.

הפעולות שניתן לבצע על ידי מחלקה זו הם חיפוש חופשי לפי טקסט, חיפוש לפי תגיות, חיפוש לפי שאלות שהמשתמש שקורא לפעולה הזין למערכת, וקבלת התוכן של שאלה וכל התשובות שהוזנו עבורה לפי id של השאלה.

public IEnumerable<Question> FreeSearch(AuthenticatedOperation<string> searchString)

public IEnumerable<Question> TagsSearch(AuthenticatedOperation<IEnumerable<string>> tags)

public IEnumerable<Question> GetMyQuestions(AuthenticationToken authToken)

public DiscussionThread GetDiscussionThreadById(AuthenticatedOperation<int> id)

## PostServices

מחלקה זו מספקת שירות לשכבת תצוגה של פעולות הזנה ומחיקה על שאלות ותשובות. כל הפעולות במחלקה זו מקבלות AuthenticationToken או AuthenticatedOperation על מנת לזהות את המשתמש שביצע את הפעולה ולאמת אותו ואת ההרשאות שלו.

המחלקה יורשת מממשק מתאים שחושף אותה כשירות WCF לשכבת ההצגה למשתמש.

עבור כל פעולה שדורשת הרשאות לא בסיסיות, המחלקה תבדוק האם למשתמש יש את ההרשאות הנחוצות. עבור פעולות הזנה ומחיקה, המחלקה תוודא שהמשתמש איננו מושתק.

בין הפעולות שניתן לבצע במחלקה זו הם הזנת שאלה חדשה, מחיקת שאלה קיימת לפי id, הזנת תשובה לשאלה, מחיקת תשובה לפי id, העלאת דרגה של שאלה\תשובה לפי id, הורדת דרגה של שאלה\תשובה לפי id, וסימון שאלה כממולצת לפי id.

לכל פעולה יתבצעו Validations מתאימים על המידע המוזן, ועל ההרשאות של המשתמש עצמו.

public void AddQuestion(AuthenticatedOperation<Question> question)

public void DeleteQuestion(AuthenticatedOperation<int> questionId)

public void AddAnswer(AuthenticatedOperation<Answer> answer)

public void DeleteAnswer(AuthenticatedOperation<int> answerId)

public void VoteUpQuestion(AuthenticatedOperation<int> questionId)

public void VoteDownQuestion(AuthenticatedOperation<int> questionId)

public void VoteUpAnswer(AuthenticatedOperation<int> answerId)

public void VoteDownAnswer(AuthenticatedOperation<int> answerId)

public void RecommendQuestion(AuthenticatedOperation<int> questionId)

## UserServices

מחלקה זו משתמש שירות לשכבת התצוגה לביצוע פעולות על משתמשים.

המחלקה יורשת מממשק מתאים שחושף אותה כשירות WCF לשכבת ההצגה למשתמש.

על פעולה מקבלת Authentication Token או AuthenticatedOperation על מנת שיהיה אפשר לזהות את המשתמש שמבצע את הפעולה ולוודא שיש לו את ההרשאות הנדרשות לביצוען. באופן זה לכל פעולה יבוצעו Validations שיוודאו את תקינות המידע שהתקבל, וההרשאות של המשתמש.

בין הפעולות שניתן לבצע במחלקה הם הוספה של משתמש חדש למערכת (זה מתבצע בתהליך הרישום למערכת), כניסה למערכת וקבלת הAuthenticationToken המתאים, קבלת המידע על כל המשתמשים במערכת (רק מרצים ורכזים יכולים לבצע זאת), שמירת מידע חדש על המשתמשים (זה יתבצע בשמירה במסך הניהול של הרכז והמרצים לאחר עדכונו), וקבלת מידע על משתמש ספציפי.

public OperationResult AddNewUser(UserInfo newUser, UserCredentials userCredentials)

public OperationResult<AuthenticationToken> LogIn(UserCredentials userCredentials)

public IEnumerable<UserInfo> GetAllUsersData(AuthenticationToken authToken)

public OperationResult SaveUsersData(AuthenticatedOperation<IEnumerable<UserInfo>> usersData)

public OperationResult<UserInfo> GetUserInfo(AuthenticationToken authToken)