# מסמך עיצוב Common

ספריית Common היא הבסיס למודלים באפליקציה. אין לה reference לאף ספרייה אחרת, אך כל הספריות, המייצגות את כל השכבות באפליקציה, מכירות אותה. אובייקטים נשלפים ממאגר הנתונים בעזרת ה-DAL ומוסבים לאוייבקטים שמועברים לשכבת ה-BL ומשם לשכבת התצוגה. שלושת השכבות צריכות לדבר באותה שפה, כלומר להכיר את אותם האובייקטים.

אובייקטים אלו מוגדרים בספריית Common.

מסמך זה יתאר את המחלקות המרכיבות אותם ואת הקשרים והממשקים בינהם.

## ממשק Ipost

ממשק זה משמש כממשק המשותף לשאלות ולתשובות במערכת. מטרתו לוודא כי המחלקות המתארות שאלות ותשובות חושפות את ה-member-ים החיוניים לפעילות המערכת.

string Content { get; }

תוכן ה-post (שאלה או תשובה) – התוכן לקריאה בלבד, שכן תוכן ה-post נקבע עם יצירתו ולא משתנה יותר.

int Ranking { get; set; }

דירוג ה-post אשר מושפע ע"י המשתמשים במערכת

int AuthorID { get; }

מזהה כותב ה-post. נקבע עם יצירת ה-post ואינו משתנה.

int ID { get; }

מזהה ה-post אשר נקבע עם יצירת ה-post. זה המפתח הייחודי של השאלות/התשובות במערכת, אשר ישמש לכל אינטראקציה עם ה-post.

## מחלקת Question : IPost

מחלקה זו מממשת את ממשק Ipost. המחלקה מייצגת את אובייקט ה"שאלה".

בנוסף ל-properties הנדרשים מעצם מימוש הממשק כוללת את התכונות הנוספות הבאות

public string Title { get; set; }

כותרת השאלה. מחרוזת המכילה את נושא השאלה. משפט או שניים שמתארים את השאלה. כאשר השאלה מוצגת ברשימת שאלות, למשל, רק כותרת השאלה תוצג (ללא גוף השאלה).

public IEnumerable<string> Tags { get; set; }

התגיות המזהות את הקטגוריות אליהן השאלה משתייכת. כל שאלה יכולה להשתייך למספר קטגוריות. הסיווג לקטגוריות מאפשר לבצע חיפושים מותאים כדי לאתר שאלות הקשורות לנושא ספציפי, למשל.

ממומש כ-Ienumerable כדי לאפשר למתכנת לבחור איך לממש את הרשימה.

public bool IsRecommended { get; set; }

האם השאלה מומלצת. משתמשים במערכת יכולים להצביע על שאלה כשאלה מומלצת, כדי לבדלה מהשאלות האחרות שפורסמו ע"י משתתפי הקורס.

public int? RightAnswerId { get; set; }

מזהה התשובה הנכונה. לכל שאלה תשובה אחת שתוגדר כתשובה הנכונה, וזאת כדי להקל על משתתפי הקורס המשתמשים במערכת ולמנוע מצב בו סטודנט השתמש בתשובה לא נכונה שניתנה ע"י משתתף אחר בקורס.

## מחלקת Answer : IPost

מחלקה זו מייצגת את אובייקט ה"תשובה" ומממשת את Ipost. בנוסף למאפיינים הנדרשים ממימוש הממשק, ממשת המחלקה את מאפיין נוסף:

public int RelatedQuestionId { get; set; }

הקישור בין התשובה לשאלה עליה היא מנסה לענות.

## מחלקת DiscussionThread

מחלקה זו מאגדת את השאלה וכל התשובות שניתנו לה לכדי אובייקט אחד. כך ניתן להתייחס לאוסף זה כאובייקט לוגי אחד ולהקל בתפעולו (למשל כדי למחוק אותו).

המחלקה כוללת את השאלה, IEnumerable של התשובות, ומספר גרסה.

מספר הגרסה משמש לצרכי עדכון תצוגה וכדי שהמערכת תדע להבדיל בין גרסה עדכנית לגרסה לא עדכנית של ה-Discussion.

public Question Question { get; set; }

השאלה אשר פותחת כל Discussion.

public IEnumerable<Answer> Answers { get; set; }

Ienumrable התשובות (מאפס ומעלה) הקיימות עבור השאלה הנוכחית

public int Version { get; set; }

מזהה גרסת ה-Discussion, אשר גדל באחד עבור כל שינוי שנעשה בתשובות או בשאלה.

## מחלקת UserInfo

פרטי המשתמשמחלקה זו מכילה את כל המאפיינים המתארים את חשבון המשתמש במערכת, למעט הסיסמא, אשר נמצאת במחלקת UserCredentials המתוארת בהמשך.

public int Ranking { get; set; }

דירוג המשתמש. מאפיין מספרי הבא לציין את "דרגתו" של המשתמש במערכת. פעילות המשתמש במערכת מזכה אותו בדירוג מסויים, אשר מאפשר לו לבצע פעולות מסויימות, כפי שנקבע בשכבת ה-Business Logic.

public string Username { get; set; }

שם המשתמש במערכת

public string EmailAddress { get; set; }

כתובת המייל של המשתמש במערכת. הכתובת משמשת גם לזיהוי חד חד ערכי של המשתמש, וגם להרחבות עתידיות – כדי לשלוח למשתמש התראות בזמן אמת, או בדו"חות תקופתיים לגבי פעילותו או פעילות אחרים במערכת הקשורה לשאלות ששאל או לתשובות שענה לשאלות אחרים.

public bool IsAdmin { get; set; }

האם המשתמש הוא מנהל בקורס, כלומר מרצה/רכז אשר לו הרשאות נרחבות יותר מאשר של משתמש סטודנט (כגון מחיקת שאלות).

public bool IsMuted { get; set; }

האם חשבון המשתמש מושתק? למרצה בקורס מותר להשתיק סטודנט אשר עבר על כללי ההתנהגות במערכת, שאל שאלות או ענה תשובות בצורה שאינה מקובלת על צוות הקורס. פעולותיו של סטודנט שהושתק יהיו מוגבלות ע"י המערכת, עד אותו מועד בו יבחר המרצה לבטל את ההשתקה.

public int ID { get; set; }

מזהה חד חד ערכי של המשתמש במערכת.

## מחלקת UserCredentials

מחלקה זו כוללת את שני המאפיינים שם משתמש וסיסמא. במחלקה זו נעשה שימוש כדי לשמור את פרטי המשתמש שכבר הזדהה מול המערכת, כדי לוודא שלכל אורך פעילותו במערכת ניתנו לו הרשאות המתאימות לסוג המשתמש (סטודנט/מרצה/רכז) ולדירוגו.

שם המשתמש והסיסמא נקבעים בבונה של המחלקה, ולא ניתנים לשינוי לאחר מכן (מבחוץ), אלא לקריאה בלבד

public string Username { get; private set; }

שם המשתמש

public string Password { get; private set; }

סיסמת המשתמש

שם המשתמש והסיסמא משמשים ליצירת ה-AuthenticationToken – המזהה שישמר בצד המערכת כדי לוודא מי הוא המשתמש עבור כל פעולה ב-BL.

## מחלקת AuthenticationToken

זהו ה-token שנוצר לאחר ביצוע login. המערכת שומרת אותו ומוודאה בכל פעילות של המשתמש כי קיים token וכי הוא זהה לזה השמור במערכת. כך נמנע מצב שמשתמש לא מורשה יבצע פעולות במערכת, או שמשתמש מורשה יבצע פעולות אשר אינו מורשה לבצע.

public string TokenString { get; private set; }

המחלקה כוללת מאפיין אחד – מחרוזת ה-token הנקבעת ע"י בונה המחלקה ולאחר מכן זמינה לקריאה בלבד ע"י משתמש המחלקה

## מחלקות OperationResult

מחלקה המשמשת כ-return type של פעולות במערכת.

המטרה היא לאפשר קבלת תשובה לגבי ביצוע פעולה. האם הביצוע הצליח, ואם לא למה, או במידה והביצוע הצליח, לכלול מידע שחוזר עם הביצוע.

לוגיקה זו אינה מאפשרת להחזיר ערך בוליאני, או להחזיר את המידע שפעולה מוצלחת מחזירה ובמקרה כישלון להחזיר null ולכן יש צורך בהחזרת complex data type כזה.

### מחלקת OperationResult הבסיסית

public bool IsSuccessful { get; private set; }

מאפיין בוליאני המתאר אם הפעולה הצליחה או נכשלה

public ICollection<string> ErrorMessages { get; private set; }

רשימת הודעות השגיאה שהתקבלו. יכולה להיות יותר משגיאה אחת. למשל, בעת יצירת משתמש, יכול להיות ששם המשתמש תפוס, וכן שכתובת ה-mail שסופקה אינה חוקית. במקום להצריך כמה הפעלות חוזרות, ולשלוח את המידע שוב ושוב אל צד השרת, ניתן לבצע ולידציה של כלל הרכיבים ולהחזיר הודעות שגיאה מרובות מתאימות.

### מחלקת OperationResult המורחבת

public class OperationResult<TPayload> : OperationResult

מחלקה זו, היורשת מ-OperationResult הבסיסית, כוללת גם Payload – מחזיק מקום גנרי שאמור לכלול את התוצר של הפעלת ה-operation הרלוונטי. למשל, במקרה של ביצוע Login מוצלח, ה-OperationResult שיוחזר יכלול גם את IsSuccessful=true, גם אפס הודעות שגיאה, וגם Payload שיכלול את ה-AuthneticationToken הרלוונטי לפעולה.

## מחלקת AuthenticatedOperation<TPayload>

כאשר משתמש במערכת מבצע פעולה, יש לוודא בכל פעם כי מדובר במשתמש מורשה אשר ביצע לוגין מסודר למערכת ואשר יש לו הרשאות לביצוע הפעולה.

במקום להעביר לכל מתודה את הפרמטרים שלה, יש להעביר אובייקט AuthenticatedOperation אשר יכלול את ה-AuthenticationToken ואת ה-payload כאשר ה-payload הוא generic placeholder שיכלול את הפרמטרים שיש להעביר.

הבנאי של המחלקה מקבל את ה-payload ואת ה-AuthenticationToken אשר זמינים לאחר מכן לקריאה בלבד לטובת ביצוע הפעולה

public AuthenticationToken Token { get; private set; }

public TPayload Payload { get; private set; }

# מרחב Services

מרחב זה מכיל את ממשקי השירותים הזמינים במערכת.

כל ממשק מכיל את המתודות שיש לממש במערכת, בחלוקה לשלוש מחלקות

## ממשק IUserServices

ממשק זה מכיל את כל הפעולות שניתן לבצע על משתמש במערכת.

ממשק זה משמש את השירותים במוגדרים במערכת ב-wcf כדי לאפשר למערכת לבצע פעולות מול מאגר הנתונים.

כל המתודות מוגדרות כ-OperationContract כדי להגדיר כי מתודות אלו מגדירות פעולות אשר הינן חלק מ"חוזה" המוגדר ב-wcf

### הוספת משתמש חדש

OperationResult AddNewUser(UserInfo newUser, UserCredentials userCredentials);

מתודה זו מאפשרת להוסיף משתמש חדש למערכת

### ביצוע Log In

OperationResult<AuthenticationToken> LogIn(UserCredentials userCredentials);

מתודה זו מאפשרת לבצע Log in למערכת

### קבלת פרטי חשבון משתמש

OperationResult<UserInfo> GetUserInfo(AuthenticationToken authToken);

מתודה זו מאפשרת למשוך את פרטי חשבון המשתמש ממאגר הנתונים

קבלת פרטי כל המשתמשים במערכת

IEnumerable<UserInfo> GetAllUsersData(AuthenticationToken authToken);

מתודה זו משמשמת לשליפת פרטי כל המשתמשים במערכת. מתודה זו משמשת את מסך ניהול המשתמשים

### שמירת פרטי משתמשים

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

מתודה זו משמשת לשמירה של אובייקטי משתמש אל מאגר הנתונים

## ממשק IPostServices

ממשק זה משמש לביצוע פעולות על Post, כלומר על שאלות או תשובות במערכת.

כל הפעולות מבוצעות לאחר בדיקת ההרשאות המתאימות במערכת, ולכן מקבלות כפרמטר אובייקט מסוג AuthenticatedOperation אשר כולל את ה-AuthenticationToken של המשתמש שמבקש לבצע את הפעולה ואת ה-payload – הפרמטרים לביצוע הפעולה.

### הוספת שאלה

void AddQuestion(AuthenticatedOperation<Question> question);

### מחיקת שאלה

void DeleteQuestion(AuthenticatedOperation<int> questionId);

### הוספת תשובה

void AddAnswer(AuthenticatedOperation<Answer> answer);

### מחיקת תשובה

void DeleteAnswer(AuthenticatedOperation<int> answerId);

### העלאת דרוג לשאלה

void VoteUpQuestion(AuthenticatedOperation<int> questionId);

### העלאת דרוג לתשובה

void VoteUpAnswer(AuthenticatedOperation<int> answerId);

### הורדת דרוג לשאלה

void VoteDownQuestion(AuthenticatedOperation<int> questionId);

### הורדת דרוג לתשובה

void VoteDownAnswer(AuthenticatedOperation<int> answerId);

### המלצה על שאלה

void RecommendQuestion(AuthenticatedOperation<int> questionId);

## ממשק IPostQueries

ממשק זה משמש להגדיר את הפעולות המוגדרות בחוזה ב-wcf ומשמשות לשליפות נתונים לטובת הצגה במסכים השונים במערכת

גם בממשק זה האובייקט המועבר כפרמטר לכל המתודות הוא אובייקט AuthenticatedOperation הכולל את ה-AuthenticationToken של המשתמש המבקש לבצע את הפעולה ואת ה-Payload המכיל את הפרמטרים לביצוע המתודה, וזאת כדי שב-BL נוכל לבדוק ולוודא שהמשתמש אכן מאושר לביצוע הפעולה המבוקשת.

### חיפוש חופשי

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

מתודה זו מחזירה רשימת שאלות התואמת את פרטי החיפוש שהועברו

### חיפוש לפי תגיות

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

שימוש במתודה זו מחזיר את כל השאלות במערכת שתויגו לפי רשימת התגיות שהועברה כפרמטר בשליפה

### שליפת השאלות שלי

IEnumerable<Question> GetMyQuestions(AuthenticationToken authToken);

מתודה זו משמשת לשליפת כל השאלות שהמשתמש שאל בעבר במערכת

### שליפת DiscussionThread ספציפי

DiscussionThread GetDiscussionThreadById(AuthenticatedOperation<int> id);

מתודה זו משמשת לשליפה Thread ספציפי לפי המזהה המספרי שלו

# מרחב Notifications

מרחב זה כולל את הממשקים והמחלקות המשמשים להעברת התראות למשתמש. בעוד המשמש מנווט במערכת וצופה בשאלות של חבריו לקורס, או לאחר ביצוע כניסה מחודשת למערכת, על המערכת להציג למשתמש פעולות שבוצעו בהיעדרו, כגון תשובות שנוספו לשאלות ששאל, שינויים בדירוג שלו, סימון תשובות שענה כנכונות וכד'.

## ממשק INotificationService

ממשק זה מכיל מתודה אחת המשמשת לשליפת התרעות במערכת עבור המשתמש הנוכחי

IEnumerable<Notification> GetNotifications(AuthenticationToken authToken);

## סוגי התרעות

המערכת כוללת סוגים שונים של התרעות. תשובה חדשה לשאלה, שינוי בדירוג שאלה/תשובה, תשובה סומנה כנכונה ועוד'.

כל סוגי ההתרעות הקיימים במערכת מוגדרים ב-Enum של סוגי ההתרעות

public enum NotificationType

{

NewAnswer,

QuestionRankedUp,

QuestionRankedDown,

AnswerRankedUp,

AnswerRankedDown,

AnswerMarkedAsRight,

QuestionMarkedAsRecommended,

PostWasDeleted,

YouWereMuted,

MarkedAsLecturer

}

בעת יצירת התרעה חדשה, מגדיר היוצר את סוג ההתרעה. כך ניתן לייצר רק התרעות מסוגים קיימים ומוגדרים במערכת, אשר עבורן מוגדרת וקיימת התנהגות מתאימה.

מחלקת Notification

מחלקה זו מגדירה את אובייקט ההתרעה במערכת.

האובייקט כולל כמה מאפיינים:

public string Message { get; set; }  
ההודעה עצמה

public bool IsNewNotification { get; set; }

האם ההודעה חדשה

public NotificationType Type { get; set; }  
סוג ההתרעה

# מרחב Logic

מרחב זה, הכלול בספרייה נפרדת (Common.Logic) כולל את חוקי הולידציה במערכת. חוקים אלו משמשים את הן את ה-BL והן את שכבת התצוגה כדי לבצע ולידציה לפעולות שהמשתמש מבקש לבצע.

הצורך בולידציה בשני המקומות ברור. פעם אחת בממשק המשתמש – גם כדי להציג למשתמש רק פעולות שמותר לו לבצע, וגם כדי להציג למשתמש הודעות שגיאה על ערכי Input שגויים עוד לפני ששולחים אותם לעיבוד בצד השרת. פעם שניה – בשכבת ה-business logic כדי לוודא שהפעולה שהגיעה לצד השרת אכן מותרת לביצוע למשתמש שביקש לבצעה.

# מרחב Validators

מרחב זה כולל את כל הולידציות שמוגדרות במערכת.

## ממשק IValidator

ממשק זה מגדיר את המתודה שכל אובייקט שיממש את ממשק ה-validator חייב לממש.

### המתודה IsValid

OperationResult IsValid(T data);

משמשת לבדיקת הפעולה אותה מבקשים לבצע. מקבלת את ה-payload הגנרי עליו מתבצעת הבדיקה, ומחזירה OperationResult (ראה הסבר למעלה).

## מחלקה אבסרקטית ValidatorDecorator

מחלקה זו מגדירה את ה-Decorator עבור ממשק ה-IValidator. השימוש ב-design pattern זה מאפשר לממשק את המתודה IsValid בצורה דינאמית בכל מימוש של הממשק. כך אין צורך לשמור את הקוד במקום מרכזי וניתן לבצע הרחבות בצורה דינאמית ולפי הצורך.

בנוסף מחלקה זו מממשת את ה-template method design pattern כך שהיא מגדירה את האלגוריתם הנכון לביצוע הולידציה (כלומר, לבדוק האם הביצוע עובר ולידציה, ואם לא להחזיר את הודעות השגיאה הרלוונטיות) אך משאירה את המימוש הספציפי למחלקות היורשות את המחלקה האבסטרקטית.

המחלקות היורשות יממשו את InternalValidator. כך ישמר האלגוריתם כולו, בלי לפגוע ביכולת של המחלקות היורשות לבצע את הבדיקה בהתאם לצרכיהן.

# מרחב Validations.User

מרחב זה מכיל את הולידציות שיש לבצע על מאפייני משתמש (בעת יצירתו או עדכון של פרטיו) וכן האם מותר לו לבצע פעולות מסויימות.

## מחלקת EmailAddressValidator : ValidatorDecorator<UserInfo>

מבצעת בדיקה של כתובת המייל של המשתמש

OperationResult IsValidInternal(UserInfo userInfo)

## מחלקת NameValidator : ValidatorDecorator<UserInfo>

מבצעת בדיקה של שם המשתמש

OperationResult IsValidInternal(UserInfo userInfo)

## מחלקת PasswordValidator : ValidatorDecorator<string>

מבצעת בדיקה של הסיסמא שנבחרה עבור המשתמש

OperationResult IsValidInternal(string userInfo)

## מחלקת RankDownValidator : ValidatorDecorator<UserInfo>

בודקת האם לשמתמש מותר להצביע "למטה" עבור שאלה/תשובה, כלומר לתת דרוג שלילי לשאלות/תשובות

IsValidInternal(UserInfo userInfo)

## מחלקת RecommendQuestionValidator : ValidatorDecorator<UserInfo>

מבצעת בדיקה האם למשתמש מותר להמליץ על שאלה

OperationResult IsValidInternal(UserInfo userInfo)

# מרחב Validations.Question

מרחב זה מכיל את הולידציות שיש לבצע על מאפייני שאלה (בעת יצירת שאלה או עדכון פרטי שאלה) וכן האם מותר לבצע פעולות מסויימות על השאלה

## מחלקת ContentValidator : ValidatorDecorator<IPost>

בודקת האם תוכן ה-post (שאלה או תשובה) תקין.

OperationResult IsValidInternal(IPost userInfo)

## מחלקת TitleValidator : ValidatorDecorator<Common.Question>

בודקת האם ה-Title, הנושא שניתן לשאלה תקין

OperationResult IsValidInternal(Common.Question userInfo)

## מחלקת MarkRightAnswerValidator : ValidatorDecorator<Answer>

בודקת האם מותר למשתמש לסמן תשובה זו כנכונה

OperationResult IsValidInternal(Answer userInfo)