מיני פרויקט במערכות חלונות תשפ"א

מערכת לניהול מערך היסעים

תוכן עניינים

מבוא

בפרויקט זה נכתוב מערכת (חלקית בלבד) לניהול מערך מידע על מערכת לתחבורה ציבורית

בבסיס הרעיון, הנוסע יכול להזמין נסיעה באוטובוס תחנות אוטובוס מתוך רשימה של תחנות מוגדרות.

למערכת ישנם 2 חלקים נפרדים :

- מבחינת החברה המפעילה :
  - ניהול המידע על תחזוקת כלי התחבורה (האוטובוסים) המשמשים לנסיעות (לבונוס)
  - **ניהול המידע על הקווים השונים שעל החברה להפעיל**
- מבחינת הנוסעים המשתמשים במערכת: (לבונוס)
  - מידע על אפשרויות הנסיעה.
  - שמירה של מידע אישי עבור הנוסע

המטרה: תרגול ויישום של

- עקרונות שפת #C
- תבניות עיצוב תכנה:
  - ארכיטקטוניות (מודל השכבות, Contract Design)
  - יצרניות (Singleton, Simple Factory)
  - התנהגותיות (Observer, Iterator)
- יצירת ממשק משתמש באמצעות תשתית הפתוח המודרנית WPF
- עבודת צוות
- חשיבה תכנותית ולמידה עצמית

הערה: ההכוונה במסמך זה היא כללית בלבד, שכן חלק מן הדרישה היא להפעיל חשיבה עצמאית יצירתית. כמו כן, חלק מהקריטריונים בבחינה הסופית של הפרויקט יתבסו על נושאים תיכנותיים שתתבקשו להתמודד איתם במהלך פיתוח הפרויקט. חשוב להדגיש שציון מתחת ל-85 בקורס זה בדומה לכל קורס מעשי איננו משמעותי בעיני המעסיקים בתעשייה. שימו לב שביצוע הפרויקט לפי דרישות מינימליות בלבד לא תאפשר לקבל ציון מעל 80.

חובה לקרוא את כל פרק של מבנה הפרויקט ודרישות כלליות לפני תחילת העבודה על הפרויקט!

דגשים:

- העבודה תתבצע אך ורק בזוגות (במקרים מיוחדים, ניתן לקבל אישור מהמרצה לעבודה יחידנית, אך ודאי לא בשלישיות). הזוגות יהיו קבועים לכל אורך הסמסטר (לא ניתן לעבור במהלך הסמסטר מזוג לזוג). לא ניתן להיות שותף עם חבר שרשום בקבוצה שונה. בכל אחד מן המקרים האלה הפרויקט ייפסל לשני השותפים.
- כל אחד מהשותפים לזוג חייב להיות שותף מלא בכל אחד מן השלבים. (במידה ויתברר כי שותף אחד עשה שלב מסוים ואלו שותף אחר עשה את החלק האחר, הציון יהיה פרופורציונלי למספר השלבים שנעשו ע"י כל אחד מן השותפים, כמו כן - ציון כל שותף יוכפל במקדם הערכה של מידת ההשתתפות השותף בפרויקט שייקבע בזמן ההגנה על הפרויקט כציון הגנה).
- נשקלת אפשרות שלאחר סוף הסמסטר, ייבחר פרויקט מצטיין מכל קבוצת תרגול ומבין אלו, ייבחרו הפרויקטים המצטיינים, והם יזכו את יוצריהם בפרס. המדד העיקרי למצוינות הוא מקוריות, למידה עצמית, וכמובן תכנון ותכנות נכונים. שימו לב שהקריטריונים לבחינת מצוינות הפרויקט הם אחידים לכל הקמפוסים ולכל הסטודנטים ללא קשר למה לימד או לא לימד אותם המרצה שלהם.
- אמנם הרושם ואיכות חווית המשתמש של הממשק הגרפי בפרויקט אינו מהווה חלק מהשיקולים למתן ציון הפרויקט, אך קחו בחשבון שבאופן טבעי ברמת תת-מודע הרושם שנעשה על הבוחן בחוויה הזו עשוי להשפיע על שיקול דעתו של הבוחן לטובה או לרעה. זאת אומרת - אל תשקיעו יותר מדי זמן ביצירת חווית משתמש איכותית במיוחד, אבל תשתדלו לא להביא משהו מכוער ומסורבל.

תהליך ההגשה:

- הפרויקט מחולק לשני שלבים המבוססים זה על קודמו.
- בסיום כל אחד מן השלבים חובה להגיש את הפרויקט בתיבת ההגשה המתאימה
- בדיקת הפרויקט תתבצע לאחר סיום הפרויקט כולו, אך תכולת ההגשות של השלבים יוכלו להעיד על רמת ומידת ההשקעה של הסטודנטים בפרויקט ויכולות להיבדק ע"י בוחן הפרויקט
- בכל שלב שמוגש המערכת שאתם כותבים חייבת ליצור הרצה תקינה וביצוע הפעולות בממשק המערכת לפי הנדרש בכל שלב
- עם הגשת השלב האחרון והמסכם תתבצע בדיקה יסודית של העבודה שתכלול גם הגנה (מעין בחינה בע"פ ע"י המרצה ~~בוחן חיצוני~~) של שני השותפים על העבודה, כ"א בנפרד. המרצה שלכם ~~אותו בוחן חיצוני~~ יקבל את ההגנה וגם יבדוק את איכות הפרויקט ויתן ציון סופי
- הציון הסופי על הפרויקט יתייחס לכל רכיביו, וכן - לאיכות ההצגה ולעמידה במועדי ההגשה
- ההגנה על השלב האחרון תתבצע בזום ~~בזמן השיעור או (אם בוחן הפרויקטים של הקבוצה אינו פנוי בזמן השיעור) בזמן שיתואם בין הבוחן לבין מרצה הקבוצה~~ במועד שייקבע ע"י המרצה של הקבוצה ע"פ החלטתו ובלי קשר לקבוצות ומרצים אחרים, ובכל מקרה ההגנה תיקבע לכל המאוחר עד סוף מועדי א' של סמסטר זה.
- שימו לב שמי שהיה שותף מלא בפרויקט שלו לכל חלקיו - רוכש הבנה מעשית ומעמיקה בחומרים הנלמדים ובעצם מגיע למבחן הרבה יותר מוכן מאשר סטודנט שלקח קטעי קוד מחבריו, קיבל הנחיה מחבריו - תוך חוסר הבנה מלאה במה שהוא עושה.
- הגשת כל שלבי הפרויקט תתבצע בעזרת טאגים של שחרורים (release tags) בדומה להגשות התרגילים בקורס, כאשר שם הטאג הינו: PR01, PR02~~, PR03~~ בהתאם למספר השלב המוגש.
- בהגשה הסופית של הפרויקט יש לצרף מסמך המפרט תוספות, וכן קודי גישה (סיסמאות) במידה וישנם.
- ציון הפרויקט יורכב מדרישות בסיס (80% מהציון) בתוספת פונקציונליות ע"פ בחירת הסטודנטים (20% מהציון). בדרישות הפרויקט כלולים "בונוסים" שהם הצעותינו לאפשרויות ההרחבה שאנחנו מציעים. אין שום צורך לבצע את כל הבונוסים על מנת להגיע לציון מירבי בקורס. הערכת רמת וכמות הבונוסים ומתן ציון עליהם בסמכות של המרצה של קבוצה.

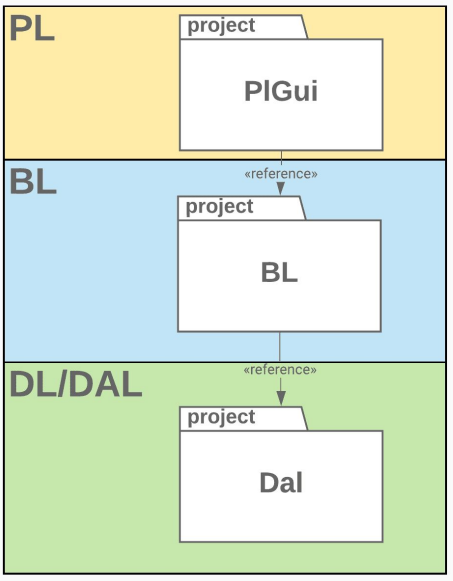
נ.ב.: יש לוודא שתכני הפרויקט (כולל תמונות וכדומה) יתאימו לרוח ההלכה – גם לפי הדעות המחמירות ביותר.

מבנה הפרויקט ודרישות כלליות

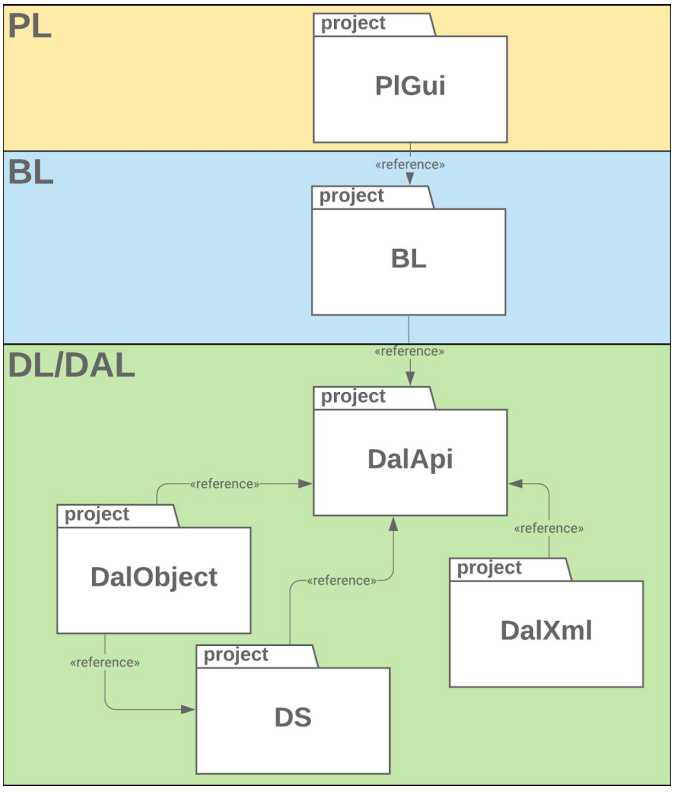
שימו לב שהנחיות ודוגמאות לבניית מודל שלושת השכבות במסמכי OSF - אינם מעודכנים ואינם מדויקים. פרויקט שהמבנה שלו יהיה ע"פ הדוגמאות מ-OSF יפסיד הרבה מאוד נקודות מהציון הסופי.
הפרויקט ייבנה באותו המאגר git ו-github ותחת אותו Solution שבו הסטודנטים עובדים מתחילת הסמסטר.

פרויקטים ב-VS ייפתחו ע"פ התבניות הבאות.

- ~~עבור ציון מירבי 80 (לפני בונוסים במוכן),~~ תבנית בסיסית של מודל השכבות כדלקמן (להלן - מבנה 1):



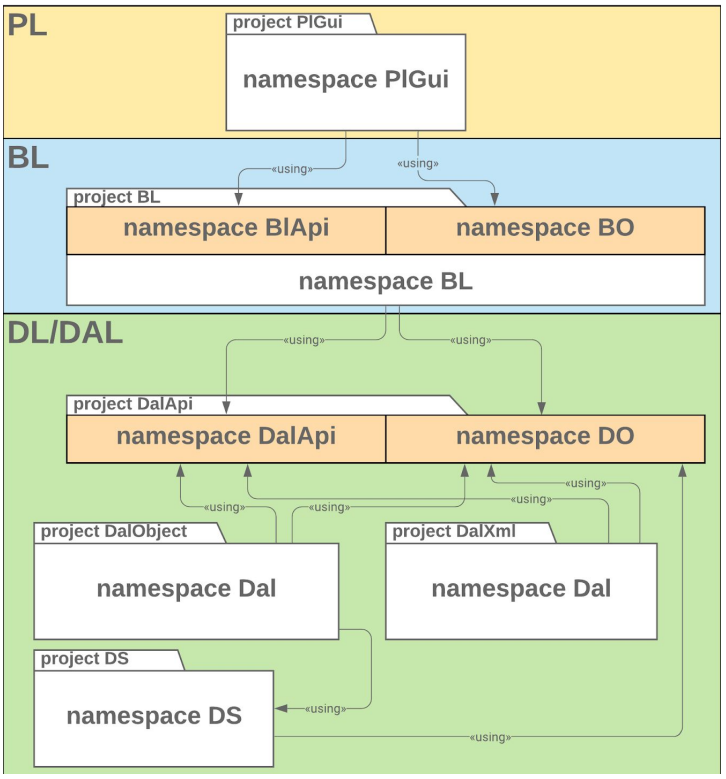
- הצעה לבונוס, תבנית של מודל השכבות כדלקמן (להלן - מבנה 2):



הפרויקט שיוגדר להרצה ב-Solution - הינו פרויקט PlGui.

כל חלקי השכבות יוגדרו במרחב שמות (namespace) מתאים.

תרשים כללי של מרחבי השמות כדלקמן (במבנה 2):



במבנה 1 - כל מרחבי השמות המופיעות בשכבת DAL יופיעו בפרויקט Dal.

דרישות כתיבת קוד כלליות:

- כל טיפוס יכלול תיעוד ע"י ///
- כל הגדרת פעולה (ז"א לא כולל מימושים של פעולות ממשק) תכלול תיעוד ע"י ///
- בכל פעולה לא טריויאלית (ורק לא טריויאלית) - תיעוד פנימי ע"י הערות בהתאם
- יש להשתמש בפרגמה region# על מנת להקל ניווט בתוך הקוד
- חובה לעצב את הקוד בהתאם לדרישות:
  - הזחות
  - ריווח שורות וריווח בתוך שורה
  - מוסכמות שמות - CamelCase\camelCase לפי הדרישות של מיקרוסופט עבור C#
  - שמות משמעותיים לפי הגישה ש הקוד צריך לתעד את עצמו

דרישות בעניין שימוש וטיפול בחריגות:

- כל שכבה תזרוק חריגה במקרה של תקלה או חוסר התאמה לוגית כלשהיא
- אסור בהחלט לזרוק חריגה מטיפוס Exception!
- כל שכבה תגדיר טיפוסים עבור החריגות שהיא זורקת במסגרת מרחב שמות מתאים של אותה השכבה (ב-DO או ב-BO) כך שהן תוכרנה בשכבה שמעליה
  - החריגות תכלולנה שדות נוספים ובנאים לפי הצורך
- כל שכבה תתפוס את החריגות מהשכבה שמתחתיה ותטפל בהן בהתאם - ייתכן שע"י זריקת חריגה משלה (במקרה של BL)
- שם טיפוס החריגה הוא המתאר את התקלה - שדות והטקסט שניתן בחריגה יכלול פרטים נוספים על התקלה אך אינם המזהה של סוג התקלה
- בשכבת הממשק (PL) תטופלנה כל החריגות בהתאם לממשק ולא תקרוס התוכנית בשום מצב בגלל חריגה שלא טופלה!
- בשכבת PL טיפול בחריגות יהיה בהתאם לפונקציונליות הנדרשת - אסור להקפיץ חלון הודעה שרק ידפיס את תוכן החריגה - בחלון הודעה הקופץ ה-PL צריך לתאר את הבעיה בצורה המובנת למשתמש (לא למתכנת) על בסיס תוכן (נתונים) באובייקט החריגה

דרישות כלליות של שכבת PL:

- אסור לבצע כל לוגיקה של הפרויקט למעט בדיקת תקינות קלט בסיסית (פורמט, תווים חוקיים וכו')
- בכל פעולה המבוצעת מ-GUI אסור לשלוח יותר מבקשה אחת ל-BL (אסור להפעיל יותר מפעולה אחת של BL)
- חובה להשתמש בכל מה שנלמד על WPF בקורס לפי הסילבוס:
  - פקדים בסיסיים
  - פקדים מרובי נתונים
  - תסדירים
  - שימוש במשאבים (Resources)
  - קישור בין תכונות תצוגה שונות וקישור של תכונות תצוגה לנתונים בקוד האחורי - בכמה דרכים עם עדכון חד כיווני ודו-כיווני
  - שימוש ב-DataContext בצורה היררכית - עבור קישור לנתונים
  - שימוש ב-ObservableCollection
- שימוש פונקציונליות נוספת של WPF שאיננה חובה בסילבוס של הקורס יזכה את הסטודנטים בניקוד בונוס, למשל כמה אפשרויות:
  - טריגרים (trigger - שלושה סוגים - טריגר תכונה, טריגר נתונים, וטריגר אירוע)
  - תבניות עיצוב (DataTemplate) בכללי ולתבניות אלמנט בפקד מרובה נתונים בפרט
  - המרות בקישור נתונים (IConverter)
  - סגנון (Style), ערכות נושא (Theme)
  - תבניות פקד (ControlTemplate)
  - טרנספורמציות של פקדים (Transform)
  - תכונות מצורפות (attached properties)
  - גרפיקה (drawing, shapes)
  - פקודות והתנהגויות (commands and behaviors)
  - ארכיטקטורה MVVM
  - ועוד ועוד ועוד - לפי הזמן והרצון של הסטודנטים
- אם סטודנטים מסוימים רוצים להשתמש ב-data binding בצורה מלאה ולכן נדרשת הוספת קוד בישויות המתקבלות מ-BO - יש ליצור ישויות PO מתאימות - אסור להוסיף כל קוד לישויות של BO - כמובן ששימוש בישויות PO יוכר בחלק הבונוס של הציון
  - בישויות PO - עבור אוספי אובייקטים ניתן (וכדאי\מומלץ) להשתמש ב-ObservableCollection
- **לבונוס**: כל בקשה ל-BL תתבצע בפועל רקע (Background Worker), כאשר מתקבלת תוצאה מבקשה כזו - יעודכן מידע באובייקטים של BO\PO השמורים באובייקט המחלקה של החלון המתאים
- לא יהיה שיתוף מידע בין החלונות - העברת המידע תתבצע בעזרת העברת ארגומנטים עבור הפרמטרים של בנאי החלון, כולל אובייקט של BL, או העברת פונקציה לבנאי בפרמטר מסוג דלגט

דרישות כלליות של שכבת BL:

- רק הטיפוסים המוגדרים במרחבי השמות BlApi (מחלקות IBL ו-BlFactory) ו-BO (ישויות לוגיות) יהיו עם הרשאת public
- כל הבקשות שעשויות להגיע משכבת PL לשכבת BL חייבות להיות מוגדרות כפעולות בממשק השכבה IBL.
- כל הטיפוסים המוגדרים במרחב השמות BL יהיו עם הרשאת ברירת מחדל (ז"א ללא הרשאה כלל שזה אומר - internal)
- בשלב הראשון בשכבה BL אסור להחזיק אובייקטים או אוספי אובייקטים של ישויות מכל סוג (ישויות נתונים או ישויות לוגיות) מחוץ לפעולות
- כל הישויות של BO:
  - יהיו PDS חלקי (PDS - Passive Data Structure):
    - **לא יכללו כל קוד** למעט העמסת פעולת ToString לצורכי debug
    - יכולים לכלול אוספים גנריים (IEnumerable<>)
    - יכולים לכלול תכונות מטיפוס של ישות BO אחרות
    - יכולים לרשת מישות BO אחרת (יש להיזהר מאד עם שימוש בירושה ב-BO!)
  - חייבות להכיל את כל המידע הנדרש עבור בקשת PL אפשרית (פונקציה של IBL), עבור פונקציונליות המיוצגת בחלון מתאים של PL
- מחלקת BlFactory הינה חלק מתבנית עיצוב Simple Factory ותהיה מחלקה סטטית שתכיל פעולה GetBl המחזירה אובייקט מטיפוס שמימש את הממשק IBL
- הפעולות המוגדרות בממשק IBL:
  - יכולות לקבל פרמטרים מטיפוסים:
    - של דוט-נט
    - של BO
    - של דלגטים בהתאם
      - על טיפוס הדלגט להיות מוגדר ב-BO או להיות אחד הטיפוסים של דלגט המוגדרים בדוט-נט
  - יכולות להחזיר ערך מטיפוס:
    - של דוט-נט
    - של BO
    - אוסף גנרי של איברים מטיפוסים כנ"ל (IEnumerable<T>) - אסור להגדיר סוג ערך מוחזר אוסף מסוג ספציפי (למשל List)
- יכולות להחזיר ערכים מטיפוסים של דוט-נט, של BO, ואוספים של BO (רק ע"י IEnumerable גנרי)
- במימוש של IBL (ז"א במחלקה הממשת את IBL):
  - בגרסה הסופית אין להשתמש בלולאת foreach במקום שניתן להשתמש ב-LINQ
  - חובה לכלול לפחות 4 שאילתות LINQtoObject
  - חובה לכלול לפחות 4 ביטויי למבדה
  - בשאילתות LINQ חובה להשתמש לפחות פעם אחת
    - ב-let
    - ב-select new
    - בקיבוץ (grouping)
    - במיון

דרישות כלליות של שכבת DAL:

- רק הטיפוסים המוגדרים במרחבי השמות DalApi (מחלקות IDAL ו-DalFactory) ו-DO (ישויות נתונים) יהיו עם הרשאת public
- כל הבקשות שעשויות להגיע משכבת BL לשכבת DAL חייבות להיות מוגדרות כפעולות בממשק השכבה IDAL
- כל הטיפוסים המוגדרים במרחב השמות Dal יהיו עם הרשאת ברירת מחדל (ז"א ללא הרשאה כלל שזה אומר - internal)
- כל ישויות ה-DO:
  - יהיו PDS (לא יכללו כל קוד למעט העמסת פעולת ToString לצורכי debug)
  - יכללו תכונות מטיפוסים שהם ValueType, string או enum **בלבד**
    - זה כולל גם (לדוגמה): DateTime, TimeSpan, כמו כן סוגי enum שאולי תגדירו גם ב-DO
  - לא יכללו תכונות מטיפוס שאינו טיפוס דוט-נט (גם לא טיפוס DO אחר, למעט enum)
  - לא יכללו שום סוג של אוסף (גם לא יכללו מערכים)
- מחלקת DalFactory הינה חלק מתבנית עיצוב Simple Factory ותהיה מחלקה סטטית שתכיל פעולה GetDal המחזירה אובייקט מטיפוס שמימש את הממשק IDal - ע"פ סוג המימוש הנדרש שהועבר בארגומנט לפרמטר מסוג מחרוזת או מספר (מבנה 1), או ע"פ ההגדרות בקובץ הקונפיגורציה (מבנה 2)
- הפעולות המוגדרות בממשק IDal:
  - יכולות לקבל פרמטרים מטיפוסים של דוט-נט, של DO, ושל דלגטים בהתאם.
  - יכולות להחזיר ערכים מטיפוסים של דוט-נט, של DO, ואוספים של DO (רק ע"י IEnumerable גנרי)
- הפעולות בממשק IDal ייבנו לפי הישויות של DO ובהתאם לסכימה של CRUD
  - סכימה CRUD או Create-Request-Update-Delete - פעולות הוספה, בקשה (אובייקט בודד, אוסף כולו, או אוסף לפי תנאי סינון), עדכון, מחיקה)
  - תנאי סינון יכול להיות ע"י פעולה נפרדת או ע"י פעולה שמועברת כארגומנט לפרמטר מטיפוס\פרדיקט
  - יתכן שחלק מהישויות לא זקוקות לחלק מפעולות CRUD לפי לוגיקת המערכת שהסטודנטים בונים - אין צורך לכתוב פונקציות שלא צריך אותן
- כל מחלקה המממשת את הממשק IDal חייבת לממש תבנית עיצוב Singleton
- במימושים של הממשק IDal אסור לבצע כל חישוב או בדיקה לוגית למעט בדיקת שלמות הנתונים (למשל בדיקת הימצאות אובייקט שנדרש למוחקו או לעדכנו, או אי הימצאות אובייקט שנדרש להוסיפו)
- במימושים של הממשק IDal באופן כללי:
  - שכבת DAL אחראית רק לגישה לנתונים, לכן השכבה לא תבצע שום בדיקות נתונים למעט המצאות או אי הימצאות של מופע הישות - בהתאם לפעולה - ע"פ מזהה או מפתח ייחודי של הישות. השכבה תזרוק חריגה מתאימה במקרה של תקלה כלשהי
  - **לבונוס**: במקרה של מחיקה אין למחוק את היישויות אלא יש לסמן ישות שלא פעילה (בישויות הנתונים המתאימות יש להוסיף שדה בהתאם) - זאת מכיוון שישויות אחרות עשויות להכיל מזהה של הישות הזו (למשל בהיסטורית נסיעות)
  - בהוספה של ישות עם מזהה טבעי יש לוודא שישות עם מזהה זהה לא קיימת באחסון הנתונים (או קיימת אך לא פעילה - **לבונוס של המחיקה כנ"ל**)
  - במחיקה או עדכון הנתונים יש לוודא שישות עם המזהה הנדרש קיימת (ופעילה - **לבונוס של המחיקה כנ"ל**)
  - על מנת לנהל מספר מזהה ייחודי עבור ישות מסוימת שאין לה שדה מזהה טבעי ייחודי (כמו קו, נסיעה, וכדומה) יש להשתמש במספר רץ המנוהל בישות של קונפיגורציה
  - אפשר שפונקציית הוספת ישות עם מספר רץ כנ"ל תחזיר את המספר המזהה הזה
  - בגרסה הסופית אין להשתמש בלולאת foreach במקום שניתן להשתמש ב-LINQ
  - **לבונוס של המחיקה כנ"ל**: אין לכלול אובייקטים לא פעילים בתשובה לבקשות רשימת\אוסף אובייקטים
- במימוש בזיכרון (עם DS) שנקרא DalObject:

- מחלקת DataSource תהיה במרחב שמות DS בפרויקט נפרד (במבנה 2) או בפרויקט Dal (במבנה 1)
  - במבנה 2 גם המחלקה DataSource תהיה עם הרשאה public על מנת לאפשר ל-DalObject גישה לאחסון הנתונים
  - המחלקה תכלול את אוספי הישויות של DO ואת נתוני הקונפיגורציה בהתאם
  - המחלקה תכלול אתחול האוספים עם כמות נכבדת (לפחות 50 תחנות, 10 קווים של לפחות 10 תחנות, 20 אוטובוסים, וכו') של נתונים התחלתיים על מנת לא להתחיל מאפס בכל הרצה
  - אפשר להשתמש בקבצי משרד התחבורה ולבחור משם נתוני תחנות וקוים אמיתיים (בפורמת csv שנתמך ע"י Excel)
  - ניתן בקלות ובמהירות להפוך שורות המידע הזה לשורות קוד ב#C או בהמשך לאלמנטים של XML ובכך לחסוך הרבה עבודה של הקלדת נתונים
- כל אובייקט המתקבל מ-BL לשמירה או עדכון חייב לעבור שיבוט (clone) לפני שמירה ב-DS
- כל אובייקט או אוסף מ-DS חייב לעבור שיבוט לפני שליחתו ל-BL
- נ.ב. "שיבוט" אוסף מתבצע בעצם ע"י שאילתת LINQ ולכן לא נדרשת כתיבת קוד מיוחד לזה
- שיבוט אובייקטים מטיפוסי DO יתבצע ע"י פונקציות הרחבה המוגדרות במחלקת עזר במרחב שמות Dal לצד המימוש המתאים של הממשק IDal

- במימוש בקבצי XML שנקרא DalXml:
  - לא יישמרו הנתונים בזיכרון אלא בתוך פעולות בלבד
  - מספרים רצים ומידע נוסף על קונפיגורציה ינוהלו בעזרת קובץ XML של קונפיגורציה המתעדכן אוטומטית בכל עדכון (יצירה של מספר רץ חדש)
  - כל פעולה תבצע קריאת הנתונים הרלוונטיים מקבצי XML בתחילתה
  - כל פעולה המבקשת לעדכן נתונים תבצע כתיבה לקבצים המתאימים לפני סיומה

נ.ב. ייתכן שבזמן ההצגה של הפרויקט תתבקשו להריץ את המערכת יותר מפעם אחת בו זמנית - ועדיין העבודה ביניהם צריכה להיות מתואמת!

תיאור שלבי הפרויקט בקצרה

- **שלב 1** (כשבועיים עד שלוש כולל חנוכה)

נבנה את המערכת על פי מודל השכבות ע"פ אחת התבניות לעיל, יש להגדיר פרויקטים מתאימים בסביבת העבודה.
השלב הראשון לא כולל תהליכונים בשכבת BL.
ממשק גרפי עבור מנהל בלבד שיאפשר הוספה, עריכה ומחיקה של ישויות - אך ללא שליחת אוטובוסים לנסיעות בפועל.
בחלק זה של הפרויקט נגדיר פרויקטים עבור כל אחת מהשכבות על חלקיהם שיהוו יחד את המיני-פרויקט שמאפיין את מערכת המידע והניהול עבור מערך ההיסעים. שלב זה כולל 4 חלקים כדלהלן:

  - חלק א' - הגדרת הממשקים של השכבות BL ו-DAL והמחלקות עבור היישויות
  - חלק ב' - הגדרת דרך העבודה מול מקור הנתונים, כלומר מימוש ממשק של שכבת ה-DAL ע"י DalObject\DalList - כאשר מקור הנתונים נמצא בזיכרון - DS/DataSource
  - חלק ג' – הגדרת לוגיקה בסיסית של המערכת, כלומר מימוש של שכבת ה-BL ע"י BIImp
  - חלק ד' – הגדרת ממשק משתמש של המיני-פרויקט, כלומר שכבת ה-PL\UI

- **שלב 2** (כשבועיים)
  - השלמת החסר מהשלב הראשון
  - בניית מימוש DAL עבור אחסון הנתונים בקבצי XML
  - סימולציה של נסיעת אוטובוסים **מדומים** בקוים והצגת "מסך" דינמי של תחנת אוטובוס שנבחרה ע"י משתמש המערכת - כולל "השלט הצהוב" והפאנל האלקטרוני של האוטובוסים המתקרבים והמגיעים לתחנה

~~הוספת ממשק גרפי עבור נוסע, שליחת אוטובוסים לנסיעות ומעקב הנסיעות האלה, כולל תהליכונים ב-BL~~

- ~~**שלב 3** (שבוע אחד)~~

~~יכלול בניית DAL עבור אחסון הנתונים בקבצי XML~~

שלב ראשון – מנהל המערכת

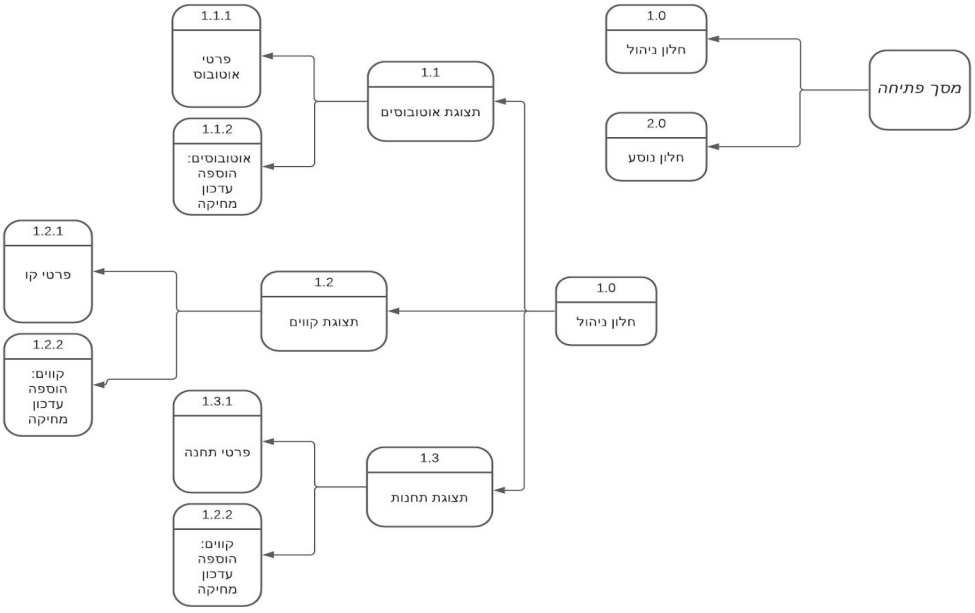
תצוגה

במערכת יש לספק אפשרות לשני סוגים של משתמשים, עובדי החברה המנהלים מידע במערך השליטה של החברה ונוסעים המשתמשים במערך ההיסעים.
התכנית תיפתח במסך ראשי המפנה לאפשרויות השונות. כאשר יהיו לפחות המסכים הבאים :
חלון כניסה ראשי - שם משתמש או אפשרות להרשמת נוסע חדש

- חלון מנהל / עובד בחברה
- *חלון נוסע (לבונוס)*

בשלב הראשון יש לממש רק את ניהול מרכז המידע של החברה:

- *תצוגה האוטובוסים הקיימים במערכת (לבונוס)*
  - *אפשרות לשלוח אוטובוס לטיפול או תדלוק*
  - *תצוגת פרטים של אוטובוס בודד*
- תצוגת הקווים הקיימים במערכת
  - בהצגת קו יש להציג מידע על התחנות במסלול הקו, כולל קוד, שם התחנה ומרחק וזמן ממוצע עד לתחנה הבאה (או מהתחנה הקודמת)
- תצוגת תחנות אוטובוס
  - כולל תחנות עוקבות (אם לא מימשתם אפשרות כזו בניהול קו כנ"ל)
  - תצוגת פרטים של תחנה בודדת
    - כולל קווי אוטובוס שעוברים בתחנה (מספר הקו + שם התחנה הסופית)
  - תצוגה לעדכון מרחק\זמן נסיעה בין שתי תחנות (אם לא מימשתם אפשרות כזו בניהול קו כנ"ל)
- אפשרויות הוספה, מחיקה ועדכון מידע
  - הוספה, גריעה משירות, עדכון פרטים של אוטובוסים (לבונוס)
  - הוספה עדכון וגריעה של קווים, כולל עדכון תחנות קו
    - ~~אם בעדכון נוספת ל~~בעדכון קו תחנה או בתצוגת תחנה ~~שאין מידע לגבי מרחק\זמן נסיעה מהתחנה הקודמת –~~ יש לאפשר תצוגה לעדכון מרחק\זמן נסיעה בין שתי תחנות (נ.ב. הוספה ו-\או עדכון של זוג של תחנות עוקבות וכמובן העדכון יתבצע דרך שכבת BL ~~בעדכון\הוספת קו~~)
  - הוספה, עדכון (עדכון מיקום תחנה הכולל עדכון מרחקים וזמנים מתחנות קודמות ואל תחנות עוקבות - לבונוס) וגריעה (גריעה - לבונוס) של תחנות אוטובוס

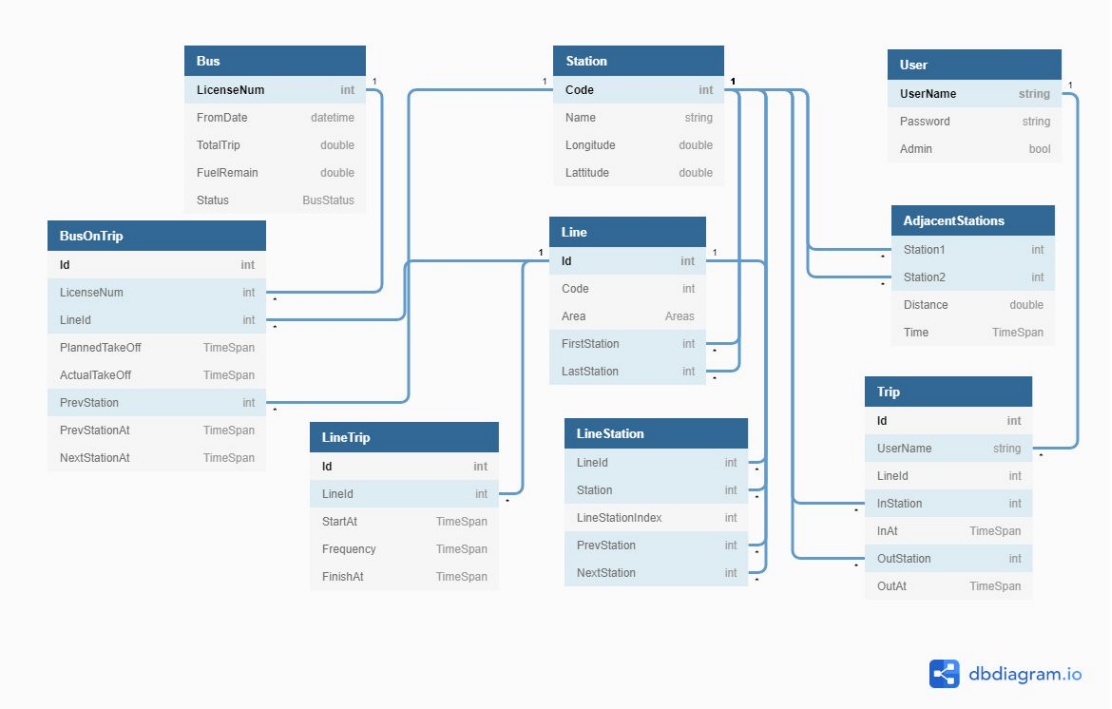


תיאור סכמתי אפשרי של מסכי השלב הראשון

ישויות נתונים במרחב שמות DO של שכבת DAL

נ.ב. אם במהלך העבודה בכל שלב תרגישו בצורך לשמור הפניה רשומת ישות כלשהי בתוך נתוני ישות אחרת (ישות נתונים או ישות לוגית) ואין לישות זו שדה מזהה ייחודי אחד - עליכם להוסיף לישות זו שדה מזהה מזהה - מספר רץ אוטומטי.

תרשים הישויות ב-DO עם קשרים ביניהם ע"י השדות המזהים (שמות התכונות והסוגים - הינם הצעה בלבד)



- אוטובוס (לבונוס)
  - מספר רישוי (התכונה המזהה של הישות, ייחודי)
  - תאריך רישוי
  - קילומטראז'
  - מיכל דלק
  - סטטוס האוטובוס: בנסיעה, זמין לנסיעה, בטיפול, בתדלוק
  - *מידע אופציונלי נוסף כרצונכם*

- אוטובוס בנסיעה (לבונוס - אבל יופיע חלקית כישות לוגית בשלב 2 בסימולציה)
  - מזהה אוטובוס בנסיעה (התכונה המזהה של הישות - מספר רץ אוטומטי)
  - מספר רישוי (מפתח הישות - חלק 1)
  - מזהה קו שבביצוע (מפתח הישות - חלק 2)
  - שעת יציאה לקו הפורמלי (מפתח הישות - חלק 3)
  - זמן יציאה בפועל
  - מס' תחנה אחרונה בקו שהאוטובוס עבר
  - זמן מעבר בתחנה האחרונה הנ"ל
  - זמן הגעה לתחנה הבאה
  - *מידע אופציונלי נוסף כרצונכם, למשל:*
    - *מס' זהות הנהג או שם הנהג*
    - *ועד לפי היצירתיות שלכם*

נ.ב. המפתח של הישות הייב להיות מפתח ייחודי!

- תחנות אוטובוס (יישות חובה)
  - קוד תחנה (התכונה המזהה של הישות, ייחודי)
  - מיקום – קווי אורך ורוחב
  - שם תחנה (למ' רח' ויצמן- רח' טשרניחובסקי - כמקובל בארץ)
  - *כתובת (אופציונלי)*

- *מידע אופציונלי נוסף כרצונכם (גישה לנכים, גגון, פנל דיגיטלי, וכו')*

- קו אוטובוס (ישות חובה)
  - מזהה קו אוטובוס (התכונה המזהה של הישות - מספר רץ אוטומטי)
  - מספר קו
  - אזור בארץ (אופציונלי - אין שימוש בדרישות חובה)
  - מס' תחנה ראשונה
  - מס' תחנה אחרונה
  - *מידע אופציונלי נוסף כרצונכם, למשל:*

- יציאת קו (ישות חובה)
  - מזהה קו אוטובוס (מפתח הישות - חלק 1)
  - זמן יציאה [או זמן התחלת תווך זמנים - לבונוס] (מפתח הישות - חלק 2)
  - תדירות (אם 0 - יציאה בודדת) - כל כמה זמן יוצא אוטובוס לקו - [לבונוס]
  - זמן סיום (רק אם תדירות גדולה מ-0) - שימו לב שעבור קו עירוני ייתכנו מספר מופעים כי התדירות משתנה בפרקי זמן היממה השונים - [לבונוס]
  - *מידע אופציונלי נוסף כרצונכם*

נ.ב.1 המפתח של הישות הייב להיות מפתח ייחודי!
נ.ב.2 יהיו יציאות מרובות לכל קו

- מידע על "זוג תחנות עוקבות" (ישות חובה)
  - קוד תחנה 1 (מפתח הישות - חלק 1)
  - קוד תחנה 2 (מפתח הישות - חלק 2)
  - מרחק
  - זמן נסיעה ממוצע
  - *מידע אופציונלי נוסף כרצונכם*

נ.ב. המפתח של הישות הייב להיות מפתח ייחודי!

- תחנת הקו (ישות חובה)
  - מזהה קו (מפתח הישות - חלק 1)
  - קוד תחנה (מפתח הישות - חלק 2)
  - מספר תחנה בקו
  - *מידע אופציונלי נוסף כרצונכם*

- משתמש (לבונוס)
  - שם משתמש (התכונה המזהה של הישות, ייחודי)
  - סיסמה
  - הרשאת ניהול

- נסיעת משתמש ~~(עבור שלב 2)~~ (לבונוס)
  - מזהה נסיעה (התכונה המזהה של הישות - מספר רץ אוטומטי)
  - שם משתמש
  - מזהה קו
  - מזהה תחנת עליה
  - זמן עליה
  - מזהה תחנת ירידה
  - זמן ירידה

הערה: *לעושי בונוסים עם נוסעים: לחברה (משתמש שנכנס כמנהל) אין גישה למידע על נוסעים אלא רק תחנות\קווים\ואטובוסים וכיוצא בזה!*

ממשק ומימוש שכבת גישה לנתונים DAL

בהתאם לדרישות כלליות לשכבה כפי שמופיע לעיל

ישויות לוגיות BO של BL

השכבה הלוגית מספקת מידע רלוונטי לפי דרישות שכבת התצוגה. ולכן מבנה הישויות הלוגיות מותאם לדרישות האלה. למשל:

- מידע על **קו אוטובוס** : כולל אוסף התחנות במסלול הקו (על פי סדר ההגעה לתחנות) עם מידע רלוונטי (בלבד) לתצוגת קו אוטובוס (קוד ושם תחנה - אך אין צורך במיקום לפי קו אורך וקו רוחב ופרטים נוספים על התחנה), ובתוספת מידע על מרחקים וזמני נסיעה בין התחנות. לכך בניתם בתרגיל ישות של **תחנת קו** - נדרשת עבודה על הישות להתאים אותה למה שכתוב כאן.
- מידע על **תחנת אוטובוס**: כולל רשימת הקווים שעוברים בתחנה - מזהה קו, מספר קו, תחנת סיום. לבונוס - זמני הגעה לתחנה ומידע רלוונטי נוסף שאינו מיותר לתצוגת מידע על תחנת אוטובוס. תחנת אוטובוס תכלול אוסף של "קו-תחנה" עם מידע מצומצם על כל קו העובר בתחנה - מספר קו ושם תחנה סופית שלו - כנ"ל.

כפי שאתם יכולים לראות - לישויות לוגיות יכול להיות מבנה מורכב - הכלה של ישויות אחרות של BO ואוספים של ישויות אחרות של BO וכדומה.

המידע יכול להגיע בתשובה למספר בקשות משכבת ה-DAL ובנוסף מידע שלא מופיע בצורה גולמית ביישויות DO.

עליכם להשלים את תמונת\רשימת הישויות הלוגיות כולל כל התכונות הפנימיות - ע"פ הנדרש בממשק GUI שאתם מפתחים.

ממשק ומימוש שכבה לוגית BL

בהתאם לדרישות כלליות לשכבה כפי שמופיע לעיל.

במימוש של BL ניתן וכדאי לבנות מחלקות עזר מתאימות, כמו כן ליצור ישויות פנימיות (internal) שעוטפות את ישויות BO לצורך תיאור והצמדה של פעולה\ות שמתבצעת\ות ע"י ישות מסוימת למחלקה שעוטפת את הישות. עליכם לכלול את הפונקציונליות שפיתחתם בתרגילים 2 ו-3ב בפרויקט - כמובן עם התאמות למודל השכבות ובהתאם לשלבי הפרויקט (למשל אין צורך בסימלוציה שבתרגיל 3ב בשלב 1 של הפרויקט).

במקרה של בקשות הוספת מופע חדש של ישות או עדכון של מופע קיים של ישות - שכבת BL היא אחראית בלעדית על בדיקת תקינות נתונים מבחינת ערך ספציפי ומבחינת תיאום הערכים עם ערכי מופעים אחרים של הישות ומידע במופעים של ישויות אחרות.

בכל מקרה של תקלה או טעות בטיפול בבקשה מ-PL, תיזרק ממימוש השכבה חריגה מתאימה, שבמקרה הצורך תוגדר ב-BO. אם התקלה קרתה בעקבות תקלה (חריגה) בשכבת DAL - מימוש שכבת BL יזרוק חריגה חדשה כאשר החריגה המקורית תיכלל כחריגה פנימית (InnerException) בחריגה החדשה - ע"י הפעלת בנאי מתאים של החריגה הנזרקת.

*שימו לב שציון הפרויקט תלוי במידה רבה מתכולת ופונקציונליות של השכבה הזו! השכבה הזו **<u>איננה</u>** בשום פנים ואופן מתווך פשוט להפעלת פעולות של שכבת DAL! וזוהי בין היתר המשמעות של הפרדת ישויות DO ו-BO - הישויות ב-BO אינן זהות בתוכנם לישויות DO - למעט מקרים כאשר הדמיון ביניהם מתבקש בגלל הדרישות שבאות מכיוון שכבת ה-PL.*

שלב שני
שמירת נתונים בקבצים
סימולטור נסיעות בקווים

השלמות שלב 1

לכאורה נראה שחלק מהסטודנטים לא שמו לב שיש ישות DO של יציאות קו (אפשר לקרוא לה גם נסיעות קו). בשינויים האחרונים היישות הזו עברה תהליך הרזיה - כל מופע של הישות יכיל מזהה קו וזמן יציאה מהתחנה הראשונה. הישות מייצגת בפועל את לו"ז נסיעות האוטובוסים בקווים. הישות הזו חשובה מאוד לשלב של סימולטור הנסיעות בהמשך.
לכן חשוב לוודא שבשכבת BL ביישות נפרדת או במסגרת של ישויות קיימות וגם בשכבת הממשק PL קיימת תמיכה בהצגת לו"ז יציאות לקווים ואפשרות להוריד או להוסיף יציאה בלו"ז של הקו. אין צורך לתמוך בשינוי זמן יציאה לקו - אפשר למחוק ולהוסיף.
כל יתר התכונות או ההערות בהקשר של היישות הזו - כולם בגדר הצעות לבונוסים.

כמו כן לאחר הבהרות רבות, כל מי שלא עשה - עליו להשלים את הישויות שבוטלו ע"י הסטודנטים למרות היותם חובה. כמו למשל ישות DO של תחנות עוקבות.

בנוסף - לקראת החלק של הסימולטור עליכם להפוך את מחלקת המימוש של שכבת ה-BL (מימוש ממשק IBL - המחלקה BllImp) ל**סינגלטון**. וזאת מכיוון שסימולטור יחזיק וינהל משאבים אשר על מנת להבטיח תקינות בגישה אליהם צריך להבטיח נקודת כניסה (אובייקט המימוש) אחת ויחידה לשכבה.

שמירת נתונים בקבצים - DalXml

החלק הזה של השלב השני של הפרויקט כולל הוספת פרויקט DalXml (בארכיטקטורה המורחבת) או מחלקה DalXml ומחלקות עזר שלה בפרויקט DAL (בארכיטקטורה הפשוטה יותר) - המחלקה תוגדר כסינגלטון ותממש את הממשק IDal באמצעות Linq-To-XML ובעזרת Xml Serializer.

יש להכין קבצי xml שיחליפו את האוספים שיצרנו כבר (כלומר יחליפו את מקור הנתונים) - קובץ אחד לכל אחת מהישויות של DO, אשר יכתבו בפורמט המתאים למבנה הישות אותה הוא מייצג. כדאי להוסיף תת-תיקיה בשם xml בתיקית ה-solution או תחת תיקיה bin (עבור מי שפתח אותה עבור ארכיטקטורה מהמורחבת).

המודולים הקשורים ל-DalXml יוגדרו במרחב שמות DAL או DL בדומה ל-DalObject. מי שהשתמש כבר בארכיטקטורה המורחבת - אנא לשים לב להתאמות האפשריות שתידרשנה בקובץ config.xml עם שמות המחלקה, הפרויקט ומרחב השמות לא יהיו זהים לברירת המחדל. זה לעשות את זה רק צריך לזכור לשים לב. המחלקה (סינגלטון) בשם DalXml מממשת ממשק IDal כפי שכבר הוזכר.. על הפרויקט חלות כל הדרישות הכלליות והטכניות כמו ב-DalObject. ניתן ליצור מחלקות עזר לעבודה עם קבצי xml שבין היתר תהיינה אחראיות על כל האתחולים, אפשרות שמירה וטעינה של קובץ וכן אפשרות תשאול ע"י שאילתת Linq. לאחר מכן יש לממש את כל המתודות של הממשק של ה-IDal. לגבי עבודה עם קבצי XML מקומיים, עיין בנספח.

בשכבה הזו עליך להחזיק נתוני קונפיגורציה רלוונטיים בתוך קובץ config.xml (אפשר להוסיף אלמנטים בהתאם לקובץ הקיים. אלו הם בעצם ה"מספרים הרצים" הנדרשים לכמה מהישויות. הנתונים שיתווספו לקובץ הם מספר רץ של מפתחות ייחודיים של קו ושל ישויות נוספות עבור מי שהוסיף בהם מזהה שהוא מספר רץ.

מבחינת דרישות השכבה:

- לפחות קובץ אחד של אחת הישויות של DO, יש להשתמש ב- linq to xml עבור כל פעולות ההוספה עדכון ומחיקה ושליפה (להשתמש באובייקטים של XElement)
- חובה לטעון את הנתונים בתחילת של כל פעולת השכבה ולשמור אותם בקובץ בסוף כל פעולת הוספה\עדכון\מחיקה על מנת לאפשר עבודה מקבילה של מספר הרצות היישום
- לא תהיה שמירת נתונים בין בקשה לבקשה בשכבה הזו - כל האובייקטים והאוספים יווצרו מקומית בתוך הפעולות, בסיום הפעולות יגיעו בכח עצמם למאסף הזבל (GC).
- במימוש של שאר הישויות ניתן להשתמש ב-Serialize, כלומר לשמור את זה ל-XML באמצעות xmlSerialize
- ניתן להשתמש בכלים כמו גיליון אקסל על מנת להמיר נתוני משרד התחבור לפורמט XML לצורך יצירת מוסד נתונים התחלתי.

סימולטור נסיעות קווים

1. נדרשות שתי תוספות ברמת התצוגה:

א הפעלת סימולטור

קבוצת פקדים (או חלופה אחרת) יתמקמו בחלון הראשי או במקום שהסטודנטים יחליטו לפי גישת חוויית המשתמש בפרויקט שלהם.

- פקד קלט ותצוגה של שעון. הפקד יהיה חשוף לקלט בזמן שהסימולטור לא מופעל. בזמן שהסימולטור מופעל - הפקד יציג דינמית את הזמן המתעדכן ולא יתאפשר קלט בפקד. הזמן יוצג כולל שעות, דקות ושניות. הערך הקלט מייצג שעת המערכת (של הפרויקט) שממנה הסימולטור מתחיל. זאת אומרת, עם הקלדתם\בחרתם 13:40:00, כלומר בהפעלת הסימולטור שעון מערכת הפרויקט הוא 13:40. ניתן לעשות את הפקד באחת האפשרויות הבאות (או כל שיטה אחרת המתקבלת על הדעת):
  1. לחלק ל-3 שדות טקסט לקלט מספרים בלבד - של שעות, דקות ושניות
  2. TimePicker עבור מי שמשתמש בתוסף של Material Design,

- פקד קלט מהירות סימולציה - מספר שלם המייצג כמה מילישניות של מחשב שוות לשניה אחת של סימולציה. למשל, אם הערך של השדה הוא 20, זה אומר שכל שניה אמיתית שעון הסימולטור (שעון המערכת של הפרויקט) מתקדם ב-20 שניות.~~כל 20 מילישניות שעון הסימולטור (שהוא שעון המערכת של הפרויקט) יתקדם בשניה אחת - זאת אומרת הסימולטור יעבוד מהר פי 50 משעון אמיתי~~. יכולת קלט בפקד תיחסם עם הפעלת הסימולטור ותיפתח מחדש עם עצירת הסימולטור.
  נ.ב. אם מימשתם כבר לפי הכתיב המבוטל לעיל (ודרך אגב - בניגוד למה שכתוב בסעיף אחר בהמשך) - גם זה מתקבל ללא הורדת ניקוד. התיקון כאן נועד להסדרת הסתירה שהייתה בניסוח המקורי.

- כפתור הפעלת\עצירת סימולטור. בהפעלת המערכת הסימולטור לא מופעל. בכפתור יופיע "הפעל" או כל טקסט מתאים אחר לפי עיצוב הסטודנטים. הפקדים הנ"ל נגישים לעריכה. בלחיצה ראשונה לכפתור:
  - נחסמת אפשרות עריכה בפקדים של שעה ומהירות הסימולטור
  - טקסט הכפתור משתנה ל"עצור" או כל טקסט מתאים אחר בהתאם לעיצוב הסטודנטים
  - מופעל פועל רקע BackgroundWorker ונשלחת ממנו בקשה לשכבת BL להפעיל את הסימולטור (ב-IBL תתווסף פעולה בשם StartSimulator). בבקשה מועברים נתוני הפקדים - זמן התחלת סימולציה ומהירות הסימולטור, ובנוסף פעולה ללא פרמטרים המחזירה TimeSpan. בעצם אובייקט החלון עם הפונקציה הזו נהיים משקיף (Observer) של שעון המערכת של הסימולטור והפעולה תזמן ע"י הסימולטור בכל מחזור התקדמות שעון המערכת. בתוך הפעולה יועבר "עדכון התקדמות" שיאפשר בפעולה מתאימה של פועל הרקע לעדכן את תצוגת שעת המערכת בפקד השעה הנ"ל.

בלחיצה השניה לכפתור:
  - פועל הרקע הנ"ל מקבל הנחיה להפסיק (ומפסיק את פעולתו)
  - טקסט הכפתור משתנה בחזרה ל"הפעל" או כל טקסט מתאים אחר בהתאם לעיצוב הסטודנטים
  - עריכת הפקדים של שעה ומהירות הסימולטור מתאפשרת מחדש

ב הצגה דינמית של תחנת אוטובוס

יוצג מידע של תחנת אוטובוס בודדת ע"פ בחירת התחנה ע"י המשתמש באחד המסכים\החלונות הקיימים לפי מה שימצא כמתאים ביותר מבחינת חוויית משתמש ע"י הסטודנטים.

בתצוגה הזו יוצג כל המידע המופיע בד"כ בשלט הצהוב בתחנת האוטובוס (רשימת מספרי הקווים יחד עם תחנות הסיום שלהם) ובלוח האלקטרוני של התחנה (האוטובוסים המתקרבים - מס' קו, תחנת סיום, מספר דקות להגעה, והן מספר האוטובוס [האחרון] שהגיע לתחנה). עבור עדכון הדינמי של הנתונים מופעל פועל הרקע בהתאם.

1. האוטובוסים המתקרבים יופיעו לפי סדר הגעתם.
2. יוצג בנפרד הקו של האוטובוס האחרון שהגיע לתחנה.

3. החלון יהיה המשקיף של מפעיל הנסיעות - ראה להלן. לכן בחלון תוגדר פעולת המשקיף שתקבל פרמטר בוסג ישות BO בשם `LineTiming` (ראה להלן) ותעדכן או תוסיף אותו ברשימת האוטובוסים המתקרבים, או במקרה שהתקבלה שעת הגעה `TimeSpan.ZERO` - יימחק האוטובוס מהרשימה של המאוטובוסים המגיעים ויוצג האוטובוס כאחרון שהגיע לתחנה.

הצעת בונוס 1: יוצגו 5 אוטובוסים ראשונים מהרשימה

הצעת בונוס 2: מס' הקו האחרון שהגיע לתחנה יימחק אחרי כמה שניות (5 או 10? להחלטתכם) אם לא הוחלף ע"י האוטובוס הבא שהגיע לפני שעבר הזמן הזה

2. סימולציה

א <u>שעון סימולציה</u>

קבלת הבקשה מ-PL להפעיל את הסימולטור, בשכבת BL יופעל השעון חל משעה שמופיע בפקד קלט\תצוגה של השעון ובקצב המוגדר בפקד מהירות סימולציה. למשל מהירות סימולציה של 50 - משמעותה שעבור שנייה של שעון אמיתי, מתקדם שעון הסימולציה ב-50 שניות.

בנוסף בהפעלת שעון הסימולציה, יופעל **המפעיל** של נסיעות קווים שתפקידו לשלוח אוטובוסים לנסיעות (ראה להלן).

1. כדאי ליצור מחלקה (סינגלטון) עבור השעון
2. חתימת פונקציית IBL עבור הפעלת הסימולטור ועובר עצירתו:

```
void StartSimulator(TimeSpan startTime, int Rate, Action<TimeSpan> updateTime);
void StopSimulator();
```

   - פרמטר `startTime` מגדיר את שעת הפעלת שעון הסימולציה
   - פרמטר `Rate` מגדיר את מהירות הסימולציה
   - פרמטר `updateTime` מעביר פעולת PL של חלון הפעלת הסימולטור עבור עדכון תצוגת שעון הסימולציה
3. שעון הסימולציה הינו המושקף (Observable) של חלון הפעלת הסימולטור, לכן עליו לכלול שדה דלגט (event) על מנת לשמור בו את המשקיף, ולהפעילו בעת עדכון השעון.
4. לצד שדה הדלגט הנ"ל תוגדר תכונה שתאפשר "הוספת" ו"מחיקת" המשקיף ע"י מימוש הפעולה `StartSimulator`. מימוש התכונה מבצע "החלפה" (דריסה) של הפעולה בדלגט במקום הוספת דלגט - זאת אומרת המושקף הזה שלנו יאפשר רק משקיף אחד בו זמנית!
5. שעון הסימולציה יפעל בתהליכון נפרד המופעל בעקבות זימון הפעולה הנ"ל
6. שעון הסימולציה יכלול "עצירה רכה" כפי שנלמד בקורס, עבור עצירה רכה מוגדר שדה (לא תכונה!) בולאני נדיף (volatile - על מנת למנוע ייעול הקוד על חשבון קריאת השדה מחדש) כדלקמן:

```
internal volatile bool Cancel;
```

7. השדה הנ"ל מומלץ להגדיר עם הרשאה פנימית על מנת לאפשר מפעיל הנסיעות (ראה להלן) גם לעצור בעזרתו. אפשרות אחרת - לעשות את השדה עם הרשאה פרטית וליצור במפעיל שדה משלו ופעולת עצירת המפעיל - לשיקול דעת הסטודנטים.
8. על מנת לשמור על דיוק השעון, השעון משתמש בסטופר (מסוג `StopWatcher`) אשר מופעל אוטומטית בהפעלת הסימולציה ע"י פעולה `Restart()` של הסטופר.
9. בעצירת הסימולטור (בעקבות בקשה מ-PL), ייעצר שעון הסימולציה ומפעיל הקווים
10. תהליכון שעון הסימולציה ייעצר לפרקי זמן קצרים ע"י פעולת `Thread.Sleep`, ובהתעוררות ממנה מעודכן שעון הסימולציה ומיודע המשקיף על השינוי בשעון (כנ"ל).
11. לא כדאי לעדכן את "השעון" בצורה ישירה אלא ליצור שעון עם TimeSpan חדש ואז לעדכן אותו (אחרת ייתכנו שיבושים בעת כתיבה\קריאה של השעון). לכן כדאי ליצור מחלקה פרטית פנימית עבור השעון שתכלול שדה ציבורי מסוג `TimeSpan`. דוגמה לגוף לולאת התהליכון של השעון:

```
simulatorClock = new Clock(simulatorStartTime + new TimeSpan(stopwatch.ElapsedTicks * simulatorRate));
clockObserver(new TimeSpan(simulatorClock.Time.Hours, simulatorClock.Time.Minutes, simulatorClock.Time.Seconds));
Thread.Sleep(your-sleep-time-in-msec);
```

נ.ב. על מנת לממש את שעון הסימולציה עליכם להבין את תפקוד המחלקות `TimeSpan` ו-`StopWatch` ואת הפונקציות שלהן שהשתמשתן בהן.

ב <u>מפעיל הנסיעות</u>

בקבלת בקשת הפעלת הסימולטור, יופעל המפעיל ע"י שעון סימולציה כנ"ל.

1. מפעיל הנסיעות מוגדר במחלקה נפרדת (סינגלטון).
2. בעקבות הפעלת הסימולטור כנ"ל, יופעל תהליכון המפעיל, עם עצירה רכה. על מנת לאפשר עצירה רכה אפשר להתבסס על שדה משעון הסימולציה או ליצור שדה עצירה משלו עם פעולת עצירה.

3. מפעיל הנסיעות הינו מושקף של חלון התצוגה הדינמית של תחנת אוטובוס. על מנת לאפשר את הוספת המשקיף של החלון:
   a. ב-IBL נוספת פעולת עדכון המעקב עם החתימה הבאה:

```
void SetStationPanel(int station, Action<LineTiming> updateBus);
```

      i. פרמטר `station` מגדיר את קוד התחנה שבמעקב
      ii. פרמטר `updateBus` מעביר פעולת PL של חלון התצוגה הדינמית של תחנה עבור עדכון תצוגת אוטובוסים מתקרבים או מגיעים לתחנה
      iii. במקרה של ביטול מעקב (סגירת לוח אלקטרוני של התחנה) יש לשלוח עדכון עם קוד תחנה מיוחד (שאינו חוקי - למשל 1-)
   b. ב-BO נוספת ישות `LineTiming` המייצגת עדכון אוטובוס מתקרב\מגיע. הישות הינה שטוחה (מבנה נתונים סביל) המכיל לפחות את התכונות הבאות:
      i. מזהה ייחודי של הקו (יכול להתגלות כשימושי)
      ii. מספר הקו (המוצג בלוח אלקטרוני של התחנה)
      iii. זמן תחילת הנסיעה (מהתחנה הראשונה של הקו) (יכול להתגלות כשימושי)
      iv. **שם** התחנה האחרונה בקו (המוצג בלוח אלקטרוני של התחנה)
      v. שעת הגעה לתחנה שבמעקב (המוצגת בלוח אלקטרוני של התחנה)
   c. כאשר האוטובוס מגיע לתחנה - יישלח עדכון למשקיף עם זמן `TimeSpan.ZERO`
   d. בדרישות הבסיס המפעיל יאפשר משקיף אחד לתחנה אחת ~~לקו אחד~~, ולכן ישמור המפעיל לצד הדלגט של המשקיף את קוד התחנה שבה מופעל "הלוח האלקטרוני".
4. בתהליכון המפעיל
   a. המפעיל יבקש מ-DAL את רשימת הנסיעות (יציאות הקווים)
   b. המפעיל יסדר את הרשימה של היציאות שלו לפי זמן היציאה
   c. המפעיל יעקוב אחרי יציאות הקווים בצורה מעגלית (זאת אומרת - לאחר הגעה לסוף הרשימה המפעיל יחזור לתחילתה
   d. המפעיל ילך לישון בין שליחת נסיעה לשליחת נסיעה לזמן המחושב
   e. עם עצירת הסימולטור - יפסיק המפעיל את שליחת הנסיעות
   f. עבור כל נסיעה "חדשה" יפעיל המפעיל תהליכון הנסיעה
5. בתהליכון הנסיעה
   a. יווצר מופע `LineTiming` עבור הנסיעה
   b. לצורכי דיבאג כדאי (אך לא חובה) ליצור שם לתהליכון שיכלול את מזהה הקו, מספר הקו, זמן יציאת הקו מהתחנה הראשונה
   c. יעבור על מסלול הנסיעה וימשוך את הזמנים הממוצעים של הגעה מתחנה לתחנה
   d. יעבור על תחנות המסלול
   e. בכל תחנה שהאוטובוס מגיעה אליה, מחושב מחדש זמן הגעה לתחנות הבאות עד התחנה האחרונה, ואם פוגשים את התחנה שבמעקב - נשלח עדכון זמן הגעה אליה ללוח האלקטרוני של התחנה
   f. יופעל שעון שינה של תהליכון כאשר הזמן יוגרל על בסיס הזמן הממוצע בתווך של איחור (עד 200%) או הקדמה (לא יותר מ-10%)
   g. כמובן בהגעה לתחנה שבמעקב - יישלח עדכון עם השעה `TimeSpan.ZERO`
   h. בהגעה לתחנה האחרונה יסתיים התהליכון, כמו כן יסתיים התהליכון בעצירת הסימולטור - ולא יישלח עדכון ללוח האלקטרוני לאחר עצירת הסימולטור

הצעה לבונוס 1: אפשרות הפעלת מספר תחנות עם לוח אלקטרוני
הצעה לבונוס 2: עבור מי שמימש ישויות וניהול אוטובוסים - שליחת אוטובוסים "אמיתיים" לנסיעה בקו (ניתן להציג מספר לוחית זיהוי של האוטובוס המתקרב - לא שעושים את זה באמת - אבל מתחיל להידמות לאפשרות לראות את המונית המתקרבת ביישומן של GETT)
הצעה לבונוס 3: יתבצע עדכון זמן הגעה שבמעקב לתחנה לא רק בהגעה לאחת התחנות הקודמות אלא לפחות פעם בשניה (האמיתית) תוך הערכת זמן ההגעה ע"פ התקדמות בין שתי תחנות נוכחיות + זמן ממוצע ביתר התחנות עד לתחנה שבמעקב. **נ.ב. - הבונוס הזה הוא משמעותי מבחינת חווית המשתמש**

ג **<u>שליחת בקשות מ-PL ל-BL עבור הסימולציה</u>**

כפי שאתה מבינים, העדכונים מהסימולור יתבצעו מתוך הליכונים נפרדים. לכן הפעלת פעולות המשקיפים לא ניתן לעדכן את התצוגה בצורה ישירה. לכן:

1. עם הפעלת הסימולטור יופעל בחלון הפעלת הסימולטור פועל רקע:
   a. בהפעלת פועל הרקע תישלח ל-BL בקשת הפעלת הסימולטור
   b. בלחיצה על כפתור עצירת סימולטור או בסיגרת חלון הפעלת הסימולטור תועבר לפועל הרקע "פקודת" עצירה
   c. בסיום פועל הרקע - ייחשפו פקדי הקלט ויוחלף טקסט כפתור הפעלת הסימולטור - כפי שמתואר לעיל
   d. בין הפעלת הסימולטור לעצירתו ימתין פועל הרקע לעצירה בלולאה תוך התעוררות פעם בשניה
   e. פונקצית המשקיף תפעיל עדכון התקדמות של פועל הרקע
   f. בעדכון ההתקדמות, תעודכן השעה
   g. <u>אם ברצונכם</u> לכפות עצירת הסימולטור גם באמצע שניית השינה של פועל הרקע - עליכם להפסיק את השינה ע"י הפעלת פעול ()Interrupt של תהליכון פועל הרקע. על מנת לעשות זאת - עליכם בתחילת פועל הרקע (לפני הלולאה) יהיה לשמור בצד את ההפניה לאובייקט התהליכון.
2. עם הפעלת חלון או תצוגת לוח אלקטרוני של התחנה
   a. בבחירת תחנה יופעל פועל רקע של החלון, אם הייתה תחנה במעקב - ייעצר פועל הרקע ויופעל מחדש
   b. בהפעלת פועל הרקע תישלח ל-BL בקשת עדכון תחנת מעקב
   c. בשינוי תחנת המעקב\סגירת החלון ייעצר פועל הרקע (כנ"ל)
   d. בסיום פועל הרקע התצוגה תתנקה ותישלח בקשת הפסקת מעקב (עם קוד תחנה 1- כנ"ל) ל-BL
   e. בין הפעלת המעקב לעצירתו ימתין פועל הרקע לעצירה בלולאה תוך התעוררות פעם בשניה
   f. פונקצית המשקיף תפעיל עדכון התקדמות של פועל הרקע
   g. בעדכון ההתקדמות, תעודכן רשימת האוטובוסים המתקרבים ו-\או האוטובוס האחרון שהגיע לתחנה
   h. <u>אם ברצונכם</u> לכפות עצירת הסימולטור גם באמצע שניית השינה של פועל הרקע - עליכם להפסיק את השינה ע"י הפעלת פעול ()Interrupt של תהליכון פועל הרקע. על מנת לעשות זאת - עליכם בתחילת פועל הרקע (לפני הלולאה) יהיה לשמור בצד את ההפניה לאובייקט התהליכון.