



Visi0n

מערכת ניהולית

**פרויקט גמר בהנדסת תוכנה
חלופת שירותי אינטרנט,
תכנות אסינכרוני ומסדי נתונים**

**דניאל סרמיאגין
תז. 216303230
בית ספר: אורט יד ליבוביץ
מנחה: שמעון מכלוף
תאריך: 24/05/2025**



**קרית החינוך - אורט יד לבוביץ'
בית ספר למנהיגות ויזמות חברתית
ניווט. להוביג. להשפיע.**



תוכן עניינים

מבוא.....	4
הרקע לפרויקט.....	4
מטרות המערכת.....	5
תיאור המערכת.....	5
גבולות המערכת.....	5
סביבת פיתוח.....	5
שפת תכנות.....	5
שכבות.....	6
פלטפורמות הלקוחות.....	6
אתגרים.....	6
ניתוח המערכת.....	7
ייזום.....	7
מטרת המערכת.....	7
מצב קיים.....	7
מצב רצוי (עתידי).....	7
עץ תהליכים.....	8
עץ התהליכים של אפליקציית ה WPF.....	8
עץ התהליכים של האתר.....	9
בסיס הנתונים.....	10
קשרי גומלין.....	10
טבלאות ראשיות.....	11
Usr_Tbl טבלת.....	11
Reminder_Tbl טבלת.....	11
Note_Tbl טבלת.....	11
Cale_Tbl טבלת.....	12
טבלאות משניות וטבלאות קישור.....	13
Personal_Tbl טבלת.....	13
Corp_Tbl טבלת.....	13
Type_Tbl טבלת.....	14
TypeUser_Tbl טבלת קישור.....	14
צד שרת.....	15
Model - שכבת ה.....	16
מחלקות נבחרות.....	17
מחלקת המשתמש הבסיסי.....	17
מחלקת משתמש פרטי.....	18
Reminder מחלקת.....	19
ViewModel - שכבת ה.....	20
מחלקות נבחרות.....	21
BaseDB מחלקת.....	21

NoteDB מחלקת.....	24
UserService מחלקת.....	27
שכבת שירותי הרשת.....	30
Service Host - הגדרות הרשת וה.....	30
ומימוש Service - ממשק ה.....	32
ממשק.....	32
מימוש.....	33
שירותי רשת חיצוניים.....	35
צד לקוח.....	37
WPF - ממשק משתמש ראשון.....	37
חלון כניסה.....	38
דף פתיחה.....	39
דף כניסה.....	40
דפי יצירה.....	41
חלון משתמש פרטי.....	43
דף בית.....	44
דפי לוח שנה.....	45
דפי רשימות.....	48
דף תזכורות.....	50
חלק הגדרות ופרטים.....	51
חלון משתמש חברה.....	52
דף פרטים.....	53
דפי הוספה.....	54
החלק האישי.....	56
blazor web app - ממשק משתמש שני.....	57
דף בית.....	58
דף כניסה.....	59
דף משתמש פרטי.....	60
דף משתמש חברה.....	61
דף הוספה.....	62
רפלקציה.....	63
נספחים.....	65
Corp.cs.....	65
Entity.cs.....	66
Event.cs.....	66
NotelItem.cs.....	67
Person.cs.....	68
Reminder.cs.....	69
User.cs.....	70
Dict0.xaml.....	71
Dict2.xaml.....	74

DictP.xaml.....	77
Menu001.xaml.....	82
Menu001.xaml.cs.....	82
CompanyDetailsPage.xaml.....	84
CompanyDetailsPage.xaml.cs.....	85
CompanyEventsPage.xaml.....	86
CompanyEventsPage.xaml.cs.....	87
CompanyNotesPage.xaml.....	89
CompanyNotesPage.xaml.cs.....	90
CompanyPersonalPage.xaml.....	92
CompanyPersonalPage.xaml.cs.....	92
CompanyRemindersPage.xaml.....	94
CompanyRemindersPage.xaml.cs.....	95
__CaleActionGP.xaml.....	96
__CaleActionGP.xaml.cs.....	97
_CaleDayViewGP.xaml.....	99
_CaleDayViewGP.xaml.cs.....	100
_NotesViewGP.xaml.....	104
_NotesViewGP.xaml.cs.....	105
CalendarGP.xaml.....	107
CalendarGP.xaml.cs.....	111
NotesGP.xaml.....	116
NotesGP.xaml.cs.....	117
RemindersGP.xaml.....	120
RemindersGP.xaml.cs.....	122
LoginPage.xaml.....	124
LoginPage.xaml.cs.....	125
NewUsr.xaml.....	127
NewUsr.xaml.cs.....	128
StartPage.xaml.....	129
StartPage.xaml.cs.....	130
TypeUserCreatePage.xaml.....	131
TypeUserCreatePage.xaml.cs.....	132
PersonalHome.xaml.....	134
PersonalHome.xaml.cs.....	135
CompanyW.xaml.....	136
CompanyW.xaml.cs.....	138
LoginW.xaml.....	140
LoginW.xaml.cs.....	141
PersonalW.xaml.....	143
PersonalW.xaml.cs.....	146

BaseDB.cs.....	151
CaleDB.cs.....	154
CorpDB.cs.....	158
NoteDB.cs.....	160
PersonalDB.cs.....	163
ReminderDB.cs.....	165
UserDB.cs.....	167
EventService.cs.....	169
NoteService.cs.....	171
ReminderService.cs.....	173
UserService.cs.....	173
appsettings.json (VWService).....	176
Host.cs.....	176
IVisionService.cs.....	178
VisionService.cs.....	179
launchSettings.json (Visi0nStarlight).....	181
app.css.....	182
MainLayout.razor.....	183
NavMenu.razor.....	184
Company.razor.....	185
Home.razor.....	186
LoginPage.razor.....	187
Personal.razor.....	188
UserGen.razor.....	190

מבוא

"Rise before me machine, for I gave you strength, for I gave you **vision**, and now you shall bring the dawn of a new age, an epoch of man-made divinity"

"And the machine rose before the man"

כמו האדם המתואר, כך גם אני שאפתי ליצירה. אולם לא הצלחתי להביא לתוצאה ראוייה, כפי שמצופה מפרויקט ברמה זו, עדיין הצבתי בסיס לחזון. כמו אותו האדם, נתתי חיים לרעיון של כלי אשר יהיה כלהב בידו של אביר.

במהלך תקופה צפיתי כיצד המערכות אשר אנו מתנהלים איתם נתקעות במגבלות. צפיתי כיצד אדם צריך מספר רב של תוכנות לניהול טוב של עסק, צפיתי כיצד לעיתים מחברת ועט עדיין טובים ממחשב. רציתי לראות מעבר לכך, שאפתי ליצור את הבסיס לכלי הנעלה. כמובן שלא הצלחתי, אך ייתכן שיום יבוא ואראה את יצירתי עולה לפניי.

לפניכם פירוט הפרויקט שלי, Visi0n, אשר מהווה מערכת ניהולית מבוססת על לוח שנה, פתקים ותזכורות. המערכת אינה קרובה ללהיות שלמה, שכן טכנולוגיה זו בלבד אינה כלל דבר חדש, ושאיפתי הייתה לנסות ליצור מערכת אשר נותנת מענה על כלל בעיות ימינו, ללא צורך בכלים אחרים. אף ברמת הפרויקט עצמו היא עדיין לא מוכנה בשלמותה, דבר שנגרר בעיקר מקשיי זמן. ובכל זאת, המערכת קרובה להישג מטרותיה ברמה הראשונית (נכון לזמן זה), והייתי רואה בה הצלחה פרוטה. חשוב להבהיר, ספר הפרויקט נכתב בהנחה שלקוראים יש את הידע המתאים בתחום, ואין צורך להסביר יתר על המידה.

הרקע לפרויקט

שם הפרויקט, כפי שנאמר, הינו Visi0n. בחרתי בו בהשראת סיפור שקראתי העוסק במכונות, ובעיקר משום שהוא מסמל את המטרה העליונה של המערכת - "לראות מבעד לעולם המבולגן". המערכת עצמה אמורה להוות ברמה הבסיסית לוח שנה אינטראקטיבי, פנקס רשימות, ואוסף תזכורות לכל משתמש. משתמש פרטי יכול לשנות רק את הפריטים שלו, ומשתמש מסוג חברה יכול לשנות את הפריטים שלו ושל המשתמשים הפרטיים שתחתיו. תחילה היה ברצוני לאפשר שימוש בשלושת התחומים בדרך חכמה, כגון ליצור תזכורת שמופיעה ביומן, אך הדבר עדיין לא נעשה משום שרק לא מזמן סיימתי לפתח את התחומים עצמם, ולא נותר לי זמן לתוספות. קהל היעד לפרויקט הוא כלל האנושות (בפרט אנשים וקבוצות או חברות), מלבד אלה שאינם יכולים להשתמש במכשירים אלקטרוניים מסיבות שונות. הפלטפורמות שנבחרו הם מחשבי windows (ברירת מחדל בשל wpf) והרשת. בחירת הנושא עצמה נעשתה גם כתוצאה מבעיות שונות שנתקלתי בהם (בעיקר ריבוי אפליקציות וחוסר תיאום ביניהם), ומכך שמערכת זו היא מערכת שימושית מאוד ופשוטה להבנה מבחינת רעיון כיוון הפיתוח שלה.

מטרות המערכת

מטרת העל של המערכת היא להביא לייעול פעילות יומית ותרומה לסדר, על ידי היותה כלי שבעזרתו ניתן לתכנן מגוון רחב של פעילויות. המערכת אמורה להיות כללית, ומתאימה במידה רבה לסוגים שונים של פעילות, כגון ניהול פרויקט או יצירת ספר. למערכת אין מטרות נלוות ספציפיות, כיוון שכלל מטרותיה עוסקות בסדר וארגון ובכך נגזרות מן מטרת העל. דוגמה למטרה כזו יכולה להיות ניהול חברה בעזרת המערכת.

תיאור המערכת

המערכת היא מערכת שרת-לקוח בעלת גישה למשתמשים במחשב הפרטי וברשת. במחשב, צד הלקוח הוא אפליקציית wpf (לוקלית) המאפשרת יצירת משתמשים, צפייה ושינוי פעילויות לוח שנה, פתקים ותזכורות, וברשת צד הלקוח פועל (דרך Service) בצורה דומה אך בעל פחות אפשרויות, על כך בהמשך. השרת פועל על בסיס נתונים של Access, לפי מודל MVVM.

גבולות המערכת

במחשב, משתמש כללי (מחוץ ל- login) מוגבל ליצירת משתמש מסוג חברה או פרטי. משתמש פרטי מוגבל לשינוי פעילויות לוח שנה, פתקים ותזכורות אשר הם רק שלו, ומשתמש חברה מוגבל לכך ולהוספה של פריטים אלו למשתמשים פרטיים קשורים. כמו במקומות אחרים, אפשרויות נוספות כגון מחיקת משתמש או הוספת משתמש קיים לחברה לא הוספו רק בשל מגבלות זמן, ואין סיבה מהותית לאי קיומן (בזאת אני מצהיר כי הייתי מוסיף אותם לולא היה לי יותר זמן לעבודה, ואוסיף אותם במידה ואעבוד עוד על הפרויקט). ברשת, נכון לעכשיו, משתמש כללי מוגבל ליצירת משתמש פרטי בלבד שאינו קשור לחברה, ומשתמש פרטי או חברה מוגבל לצפייה בתכנים אישיים ללא אפשרות שינוי.

בביבת פיתוח

הפרויקט הינו פרויקט vs והוא פותח ב- visual studio. צד השרת פותח בפרויקט C# כללי ב- .NET 7 וצדדי הלקוח הם פרויקטים ב- .NET 7 wpf ו- .NET 8 blazor.

שפת תכנות

שפת התכנות המרכזית היא C#, ועימה השתמשתי ב: html, sql, xaml. בפרויקט ניתן גם למצוא קוד ב: css, javascript.

שכבות

הפרויקט עובד על בסיס MVVM, ובו שכבת Model (קוד מחלקות שנמצא על השרת), שכבת ViewModel (קוד קישור בין שכבת Model, בסיס הנתונים, אפליקציה לוקלית ושכבת ה-Service), שכבת בסיס הנתונים, שכבת Web Service (מקשרת בין צד לקוח חיצוני ל-ViewModel) ושכבת ה-View אשר היא ממשקי הלקוחות.

פלטפורמות הלקוחות

הפלטפורמות הן: מחשב פרטי (wpf) - נבחר משום שנלמד במסגרת בית הספר ופועל על מערכת הפעלה windows שקיימת על רוב המחשבים; הרשת - כללי לכל מכשיר עם חיבור לאינטרנט, ובכך מתאים לקהל היעד.

אתגרים

אתגר אחד שאני יכול לציין, והוא המשמעותי מבין כולם, הוא ניהול זמן. פרויקט זה אינו פרויקט בקנה מידה גדול במיוחד, ולו הייתי עובד עליו באופן עקבי הייתי מספיק להוסיף ולעשות עוד הרבה, אולם כפי שניתן לראות לא הבאתי את הפרויקט למצב זה. ניתן לומר שזאת אחת הטעויות בפעולותיי, ועם זאת, עדיין התמודדתי עימו בצורה מספקת.

אתגר נוסף היה להתאים כלים שונים על סמך ידע אישי. בהרבה מן חלקי הפרויקט האפשרויות אשר לומדו לא תאמו לפרויקט, ולכן נאלצתי להשתמש באלטרנטיבות, למשל השתמשתי ב-CoreWCF בשכבת ה-Service במקום האפשרות שלומדה כיוון שהפרויקט פועל ב-.NET ולא ב-.NET framework. אני יכול להעיד שהתמודדתי גם עם אתגר זה. היו כמובן עוד אתגרים, אך אינני מאמין שקיים צורך לפרט עליהם.

ניתוח המערכת

ייזום

נושא המערכת הוא ניהול, והיא הינה מערכת ניהולית המאפשרת תמיכה בהתנהלות היומית של בני אדם. המערכת מהווה מקבץ של תוכנות שימושיות, בהם לוח שנה אינטראקטיבי, פנקס רשימות, ותזכורות. כאמור, בעזרת המערכת יהיה ניתן לנהל מגוון רחב של פעילויות, כגון פרויקט או התנהלות יומיומית כללית.

מטרת המערכת

למערכת מטרה פשוטה, והיא תמיכה בניהולם של אנשים בכל משימתם. לפי ניסיון אישי, הכלי הכללי השימושי ביותר למשימה כזו הוא בעצם יומן, ומכן ביסוס המערכת על פנקס רשימות, לוח שנה ותזכורות. הרחבתי מטרה זו מעבר למקרה הפרטי למקרה הקבוצתי, אשר בו ניתן להשתמש באלמנטים אלו על מנת לעזור בניהול קבוצת אנשים או חברה.

מצב קיים

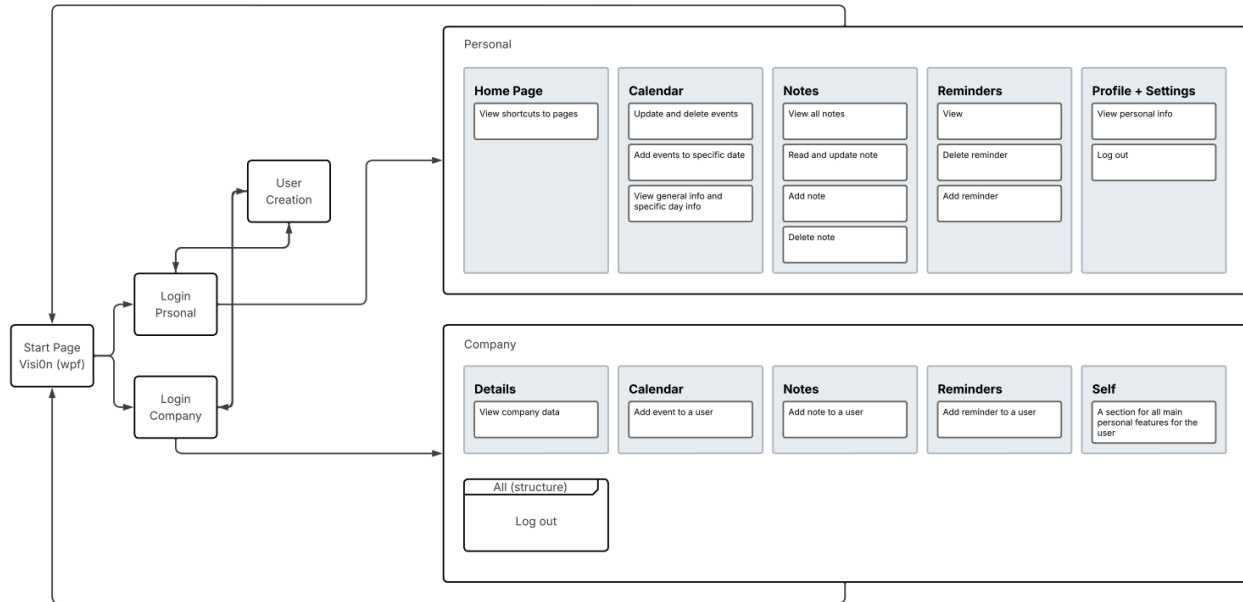
בעולם של היום יש מגוון רחב של פעילויות ותהליכים אשר יכולים להיות מבלבלים. לפיכך נוצר צורך לבני האדם באמצעי ניהול כדי לפשט את חווית הניווט בעולם המודרני ולהעלות את רמת היעילות במשימותיו החשובות. כלים אשר מתעסקים בבעיה זו יכולים להיות שימושיים לכל סוגי האנשים, לארגונים וחברות, וגורמים שונים העוסקים בלוגיסטיקה. כיום ישנם מגוון מוצרים העוסקים בנושא. בהם יש את אפליקציות הבסיס של אפל (calendar, reminders, notes), אפליקציות גוגל (google calendar, google docs), ועוד מגוון של אפליקציות דומות אחרות שניתן למצוא באינטרנט, כולל תוכנות ואפליקציות מקצועיות אשר נועדו למשימות ספציפיות (כגון ניהול פרויקט). לאפליקציות שצוינו יש מכנה משותף: הן כולן נוצרו כדי לעשות משימה אחת בלבד, כגון לנהל לוח שנה או פנקס רשימות. ישנם מעט כלים אשר מאפשרים ריבוי רב של פעולות, ותכונה כזאת מהווה את הפתרון המוחלט לבעיה, במידה שתטפל בכל הפעולות ברמה גבוהה. זכור לי כי פגשתי בכמה אפליקציות מסוג זה, והם אכן השיגו הצלחה מרובה, אך לצערי שכחתי את שמם.

מצב רצוי (עתידי)

המצב הרצוי הוא מערכת המאפשרת טיפול במגוון רחב של משימות, במקום אחד. מערכת אשר שואפת לטפל בכל בעיות הארגון בחיי היומיום בצורה טובה היא הפתרון האידיאלי לבעיה שצויינה, ומאותה סיבה היא גם תתעלה מעל המתחרים. השאיפה שלי הייתה ליצור תוכנה מסוג זה, אולם במגבלות הפרויקט היא לא הייתה אמורה להגיע למצב האידיאלי גם ממגבלות ידע וטכנולוגיה נתונה וגם ממגבלות זמן. לכן, שאיפתי הייתה ליצור מערכת המספקת לוח שנה אינטראקטיבי, פנקס רשימות ותזכורות בלבד. תוכנה גם אופציה לשלב בין האלמנטים במידה והיו משאבים לכך, אך היא לא בוצעה נכון לעכשיו.

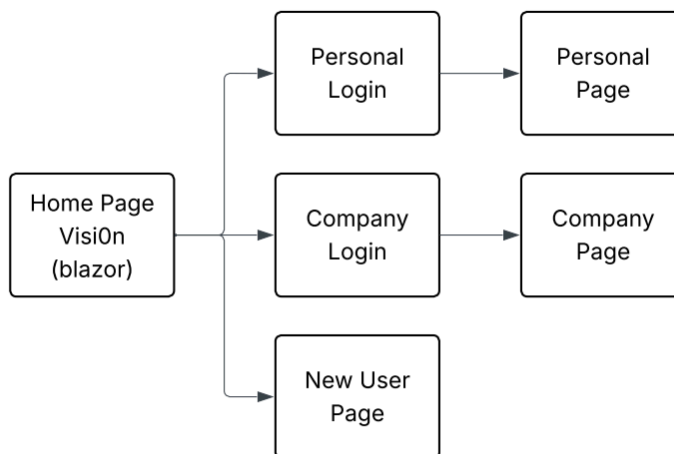
עץ תהליכים

עץ התהליכים של אפליקציית ה- WPF



האפליקציה נפתחת בחלון כניסה כללי, ממנו ניתן לבחור בכניסה כמשתמש פרטי או כמשתמש חברה. בכניסת המשתמש קיימת אופציה לעבור לדף יצירת המשתמש, אשר סוגו יהיה כסוג הכניסה הנבחר. לאחר הכניסה כמשתמש פרטי נפתח חלון חדש, בו ישנה אופציה לעבור בין הדפים המתוארים ולבצע פעולות בהתאם. באופן דומה פועל חלון החברה.

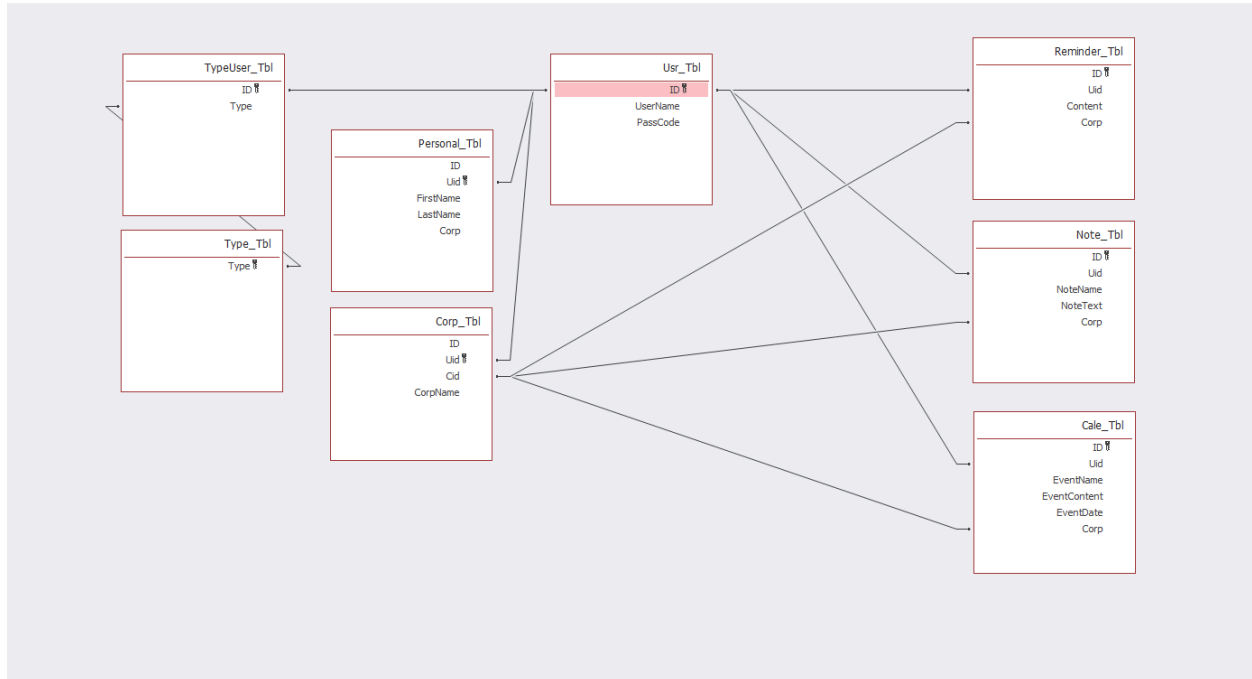
עץ התהליכים של האתר



האתר נפתח ב- layout הכללי אשר בתוכו נפתח דף הבית. בעזרת ה- layout ניתן לנווט לדפי הכניסה ולדף היצירה. בדף היצירה ניתן ליצור משתמש פרטי (באופן ידני או בעזרת random user generator api), ובדפי הכניסה ניתן להיכנס לדפי תצוגת נתונים (אין פעולות שניתן לעשות בהם מלבד צפייה) בהתאם במידה ופרטי הכניסה נכונים.

בסיס הנתונים

קשרי גומלין




מתוארים קשרי הגומלין בין הטבלאות. חשוב לציין כי לא ניתן לראות את סוג קשרי הגומלין בתמונה משום שלא בחרתי לאכוף את שלמות קשרי הגומלין ב- Access, מרצון של- Access לא תהיה אפשרות להשפיע על הנתונים (אך כיוון שבבסיסות האלו בכל אופן לא תהיה פגיעה בנתונים בחירה זו אינה בעלת ערך). הטבלאות הראשיות הן: Usr_Tbl; Reminder_Tbl; Note_Tbl; Cale_Tbl. הטבלאות המשניות הן: Type_Tbl; Personal_Tbl; Corp_Tbl. טבלת הקישור בין טבלת המשתמשים לטבלת סוג המשתמשים היא TypeUser_Tbl.

הקשרים הלוגיים פועלים כך: כל משתמש יכול להיות משתמש פרטי ו/או חברה (לצרכי סדר, העדיפות היא רק סוג אחד), וזאת מגדיר ערך סוגו מטבלת הסוגים. לכל משתמש מקושרות תזכורות, פעילויות לוח שנה ורשימות, ללא קשר לסוגו. במידה ופעילות/רשימה/תזכורת קשורה לחברה על ידי ערך מאפיין החברה לפי טבלת משתמשי חברה, כלל המשתמשים אשר בחלק המשתמש הפרטי שקשור אליהם על ידי טבלת משתמשים פרטיים מצויין ערך מאפיין אותה חברה, יראו את אותם פריטים בידיעה כי מקורן בפעילות החברה שהם קשורים אליה.

לפיכך: טבלת משתמשים קשורה לטבלת משתמשים פרטיים/חברה ב- Uid/ID ביחס יחיד ליחיד; טבלת משתמשים קשורה לטבלת סוג דרך טבלת מקשרת ב- Type/Type, בקשרים של יחיד ליחיד (מאפיין) ויחיד לרבים (סוג); בין טבלאות הפריטים לטבלת המשתמשים קיים קשר יחיד לרבים ב- ID/Uid; בין טבלת חברה לטבלת הפריטים קיים קשר יחיד לרבים ב- Cid/Corp (מאפיין חברה, אשר הוא לא מאפיין משתמש).


טבלאות ראשיות

טבלת Usr_Tbl

שם שדה	סוג נתונים
ID 	מספור אוטומטי
UserName	טקסט קצר
PassCode	טקסט קצר


טבלת משתמשים. שדה ID הוא המאפיין של המשתמש, שדה UserName הוא שם המשתמש, ושדה PassCode הוא סיסמת המשתמש. חשוב לציין שכל שדה מסוג טקסט קצר מוגבל ל- 255 תווים, והשמת מידע העובר את הגבול הזה יחזיר שגיאה (לא טיפולית בבעיה נכון לזמן זה, ככל הנראה אתקן בהמשך).

טבלת Reminder_Tbl

שם שדה	סוג נתונים
ID 	מספור אוטומטי
Uid	מספר
Content	טקסט קצר
Corp	טקסט קצר


טבלת התזכורות. שדה ID מהווה את המאפיין הייחודי של התזכורת (נמצא בכל טבלה מלבד טבלת הסוג והטבלה המקשרת על מנת להבטיח בחירה ייחודית אם המצב יצריך), Uid הוא המאפיין של המשתמש אשר אליו התזכורת מקושרת, Content הוא התזכורת עצמה, הטקסט שלה, ושדה Corp הוא מאפיין החברה. מאפיין החברה מייצג אם הפריט נוצר על ידי חברה, ושדה זה אינו משפיע על מצב הפריט בקוד מעבר לקבלת צבע שונה בממשק המשתמש.

טבלת Note_Tbl

שם שדה	סוג נתונים
ID 	מספור אוטומטי
Uid	מספר
NoteName	טקסט קצר
NoteText	טקסט ארוך
Corp	טקסט קצר

טבלת הרשימות. שדה ID הוא המאפיין הייחודי, שדה Uid הוא מאפיין המשתמש, שדה Corp הוא מאפיין החברה, NoteName הוא שם הרשימה ו- NoteText הוא תוכן הרשימה. ניתן לראות כי סוגו של שדה התוכן הוא טקסט ארוך (בניגוד לבחירת ברירת מחדל של string כטקסט קצר, עד 255 תווים), זאת כדי לאפשר שמירה של טקסטים ארוכים (יחסית). אולם גם לסוג זה ישנה מגבלה, ויש להתייחס למצב זה (כפי שצינתי אתקן בהזדמנות).


טבלת Cale_Tbl

שם שדה	סוג נתונים
ID 	מספור אוטומטי
Uid	מספר
EventName	טקסט קצר
EventContent	טקסט קצר
EventDate	טקסט קצר
Corp	טקסט קצר

טבלת פעילויות לוח שנה. שדות ID, Uid, Corp כפי שתוארו קודם, שדה EventName מהווה את שם הפעילות, EventContent מתייחס למידע על הפעילות (תקציר הפעילות), ו- EventDate הוא תאריך הפעילות.


טבלאות משניות וטבלאות קישור

טבלת Personal_Tbl

סוג נתונים	שם שדה	
מספור אוטומטי	ID	
מספר	Uid	
טקסט קצר	FirstName	
טקסט קצר	LastName	
טקסט קצר	Corp	


טבלה זו מהווה את החלק האישי של המשתמש הפרטי, ואמנם חלק זה אינו נחוץ לפעילות המשתמש (הפעולות ניתנות לביצוע כל עוד קיים משתמש בסיס) הוא חלק חשוב ומשלים לחווית המשתמש הפרטי, וגם החלק המאפשר את תכונות החברה בפרטיו (ללא פרטים אלו לא ניתן יהיה לקשר בין המשתמש למשתמש חברה). שדה ID הוא המאפיין הייחודי של הרשומה, Uid הוא מאפיין המשתמש, שדות FirstName, LastName מהווים שם פרטי ושם משפחה בהתאם, ושדה Corp הוא מאפיין החברה הקושר בין המשתמש לחברה, כמו בטבלאות האחרות הוא יכול להיות בלתי פעיל (ערך מצב זה מסומן ב: -).

טבלת Corp_Tbl

סוג נתונים	שם שדה	
מספור אוטומטי	ID	
מספר	Uid	
טקסט קצר	Cid	
טקסט קצר	CorpName	

לטבלת החברות תפקיד דומה לטבלת המשתמשים הפרטיים, אך כאן היא מהווה את החלק המשלים למשתמש החברה. כיוון שאופן הפעולה של חברה זקוק למאפיין חברה חלק זה הוא חלק חיוני למשתמש החברה, שבלעדיו יכול לפעול רק כמשתמש בסיס (כמו משתמש פרטי, אך ללא פרטים אישיים. על סמך זה קיים החלק האישי בממשק משתמשי החברה, הרי גם מנהל יוכל לעשות שימוש בתוכנה בשביל עצמו, ולא רק בשביל אחרים כפי שעוצב חלק החברה). שדות ID ו- Uid מהווים מאפיין רשומה ומאפיין משתמש בהתאם, Cid הוא מאפיין החברה, כמו שם משתמש לחברה, ו- CorpName הוא שם החברה המלא.


טבלת Type_Tbl

סוג נתונים	שם שדה	
מספר	Type	

טבלת הסוגים היא פשוטה ביותר - היא מקיימת את כל אפשרויות סוגי המשתמשים. כאן בא לידי ביטוי סוג המשתמש, וניתן לראות שהוא אינו קשור ישירות לטבלת חברות/משתמשים פרטיים. זאת משום שסוג המשתמש הוא אחד ממפתחות הגישה קודם כל (סוג 1, פרטי, יכול להיכנס רק לממשק פרטי; סוג 2, חברה, יכול להיכנס רק לממשק חברה; סוג 3 מאפשר את שני הממשקים וקיים בעיקר בשל בדיקות המערכת), ורק אז מאפיין המביא לקיום חלק פרטי/חברה.

טבלה זו קיימת וערך הסוג אינו כלול בטבלת משתמשי הבסיס על מנת לשמר את טוהר טבלת המשתמשים ולאפשר יותר סדר במצב של הוספת שדות.

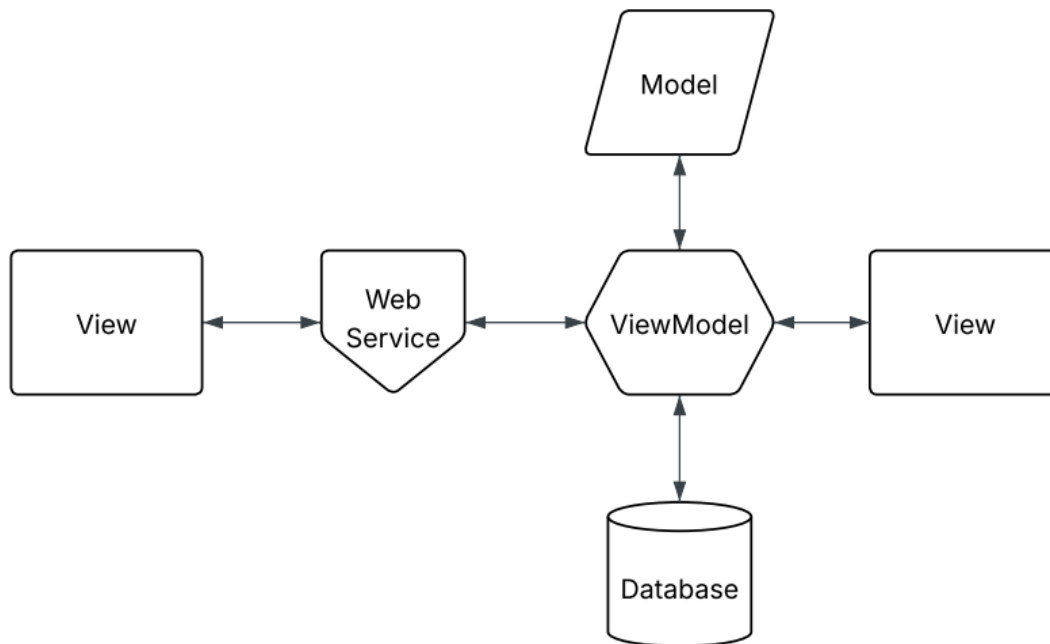
טבלת קישור TypeUser_Tbl

סוג נתונים	שם שדה	
מספר	ID	
מספר	Type	

זוהי הטבלה המקשרת בין טבלת הסוגים לטבלת המשתמשים (הקישור נעשה דרך התאמה בין סוג למאפיין), ID הוא מאפיין המשתמש ו-Type הוא סוגו.

צד שרת

כפי שצויין קודם, הפרויקט בנוי על פי שכבות MVVM, ולכן, בצד השרת נמצאות שכבות ה- Model ו- ViewModel. שכבת ה- View היא ממשקי הלקוחות. שכבת המודל היא השכבה המציבה את מבנה המידע של הפרויקט, ושכבת ה- ViewModel היא השכבה שמטפלת בזרימתו בין מסד הנתונים והשכבות השונות. ניתן לראות פירוט על זרימת המידע בתרשים הבא.

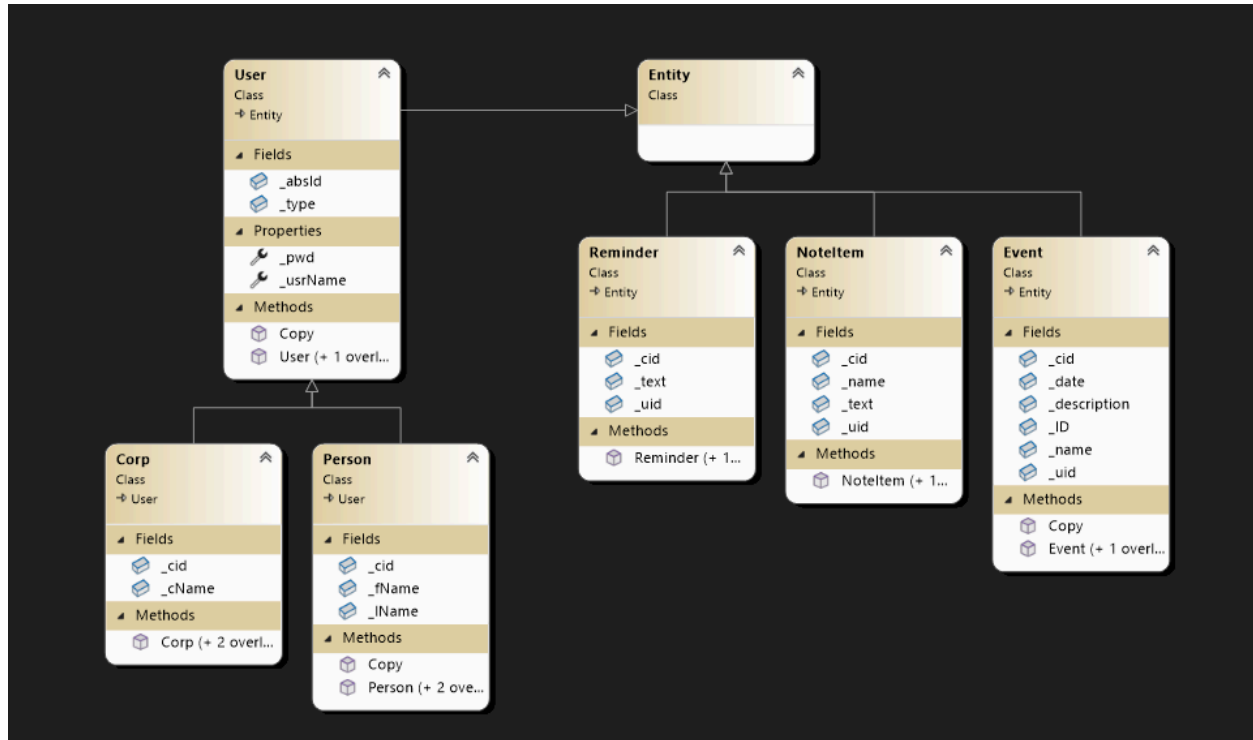


כמתואר, מידע נכנס ויוצא מן מסד הנתונים, עובר ב- ViewModel אשר נקרא על ידי ה- View או ה- Service שנקרא על ידי ה- View, ואשר משתמש במידע הפעיל בעזרת ה- Model. בין כל שכבה יש מעבר דו כיווני של המידע, וניתן לראות זאת לדוגמא כאשר אדם מבצע כניסה: הוא מכניס את פרטיו לממשק המשתמש, הממשק קורא לפעולת הכניסה הנמצאת ב- ViewModel, ה- ViewModel קורא מן מסד הנתונים ומכניס נתונים אלה לעצמים שמקורם ב- Model, משווה בין הערכים ומחזיר ל- View את תוצאת הכניסה. עוד דבר שיש להוסיף עליו הוא שקיים גם קשר ישיר בין ה- Model ל- View הלוקלי ול- Service, וזאת על מנת לספק מעבר של עצמים מורכבים בין השכבות.

השרת משתמש ב- OleDb על מנת להתחבר למסד הנתונים, לכן על מנת שהמערכת תפעל השרת חייב להיות על מחשב בעל מערכת הפעלה Windows (ככל הידוע לי) ועליו חייבים להיות רכיבי Access עדכניים יחסית (Access 2007 היא הגרסה הראשונה המאפשרת את ה- Provider).

שכבת ה- Model

בשכבה זו נמצאים קטעי הקוד של מחלקות הנתונים. מטרת השכבה לספק מעברי מידע בצורה אינטואיטיבית, והיא המוקד של תכנות מונחה עצמים בפרויקט. התרשים הבא מתאר בקצרה את המחלקות והקשרים ביניהם.



ניתן לראות שכל העצמים יורשים ממחלקה ריקה Entity, מטרתו לספק הכללה לכל מחלקות הנתונים. User היא מחלקת המשתמשים בהתאם לטבלת המשתמשים וטבלת הסוגים במסד הנתונים, והמחלקות היורשות ממנה הינן בהתאם משתמשי חברה ומשתמשים פרטיים. אין להתייחס להבדל בין Fields ל- Properties בתרשים. ניתן לראות איך ההשלמה של משתמש פרטי/חברה באה לידי ביטוי בקוד. שלוש המחלקות האחרות שיוורשות מ- Entity מהוות את הפריטים, שהם רשימות, פעילויות/אירועי לוח שנה ותזכורות.

מחלקות נבחרות

מחלקת המשתמש הבסיסי

```
public class User : Entity
{
    public int _absId;
    public string _usrName { get; set; }
    public string _pwd { get; set; }
    public int _type;

    public User()
    {
        _usrName = "";
        _pwd = "";
        _absId = -1;
        _type = 0;
    }

    public User(string un, string pd, int id, int t)
    {
        _usrName = un;
        _pwd = pd;
        _absId = id;
        _type = t;
    }

    public virtual void Copy(User u)
    {
        _usrName = u._usrName;
        _pwd = u._pwd;
        _absId = u._absId;
        _type = u._type;
    }
}
```

עצם מסוג User מכיל את ארבעת התכונות הבסיסיות של משתמש (שם משתמש, סיסמה, מאפיין ייחודי, סוג). ברגע זה כל המחלקות אינן מאובטחות ולכן גם כל תכונות העצם במחלקה זו ומחלקות אחרות בעלות גישה ציבורית, ולכן גם אין פעולות מחזירות או מיישמות לתכונות. מלבד constructor ברירת מחדל ומפורט קיימת פעולת Copy המעתיקה את ערכי User אחר לעצם.

מחלקת משתמש פרטי

```

public class Person : User
{
    public string _fName;
    public string _lName;
    public string _cid;

    public Person() : base() { _fName = ""; _lName = ""; _cid = "-"; }

    public Person(string cid, string fName, string lName, string un, string
pd, int id) : base(un, pd, id, 1)
    {
        _cid = cid;
        _fName = fName;
        _lName = lName;
    }

    public Person(User u, string f, string l, string c) : base(u._usrName,
u._pwd, u._absId, 1)
    {
        _cid = c;
        _fName = f;
        _lName = l;
    }

    public override void Copy(User u)
    {
        base.Copy(u);
        Person p; if (u is Person) p = u as Person; else p = new Person();
        _cid = p._cid;
        _fName = p._fName;
        _lName = p._lName;
    }
}

```

כאן ניתן לראות את קיומם של עוד תכונות (שם פרטי, שם משפחה, מאפיין חברה), וזאת בהתבססות על ההורשה ממחלקת User. כמו ברוב המחלקות ישנו בנאי ברירת מחדל (היוצר איבר "ריק"), בנאי מפורט, ויש גם בנאי משלים שמשתמש בעצם User. ישנה גם פעולת Copy, אשר הורחבה לרמת העצם.

מחלקת Reminder

```

public class Reminder : Entity
{
    public int _uid;
    public string _text;
    public string _cid;

    public Reminder()
    {
        _uid = -1;
        _text = "";
        _cid = "--";
    }

    public Reminder(int u, string t, string c = "--")
    {
        _uid = u;
        _text = t;
        _cid = c;
    }
}

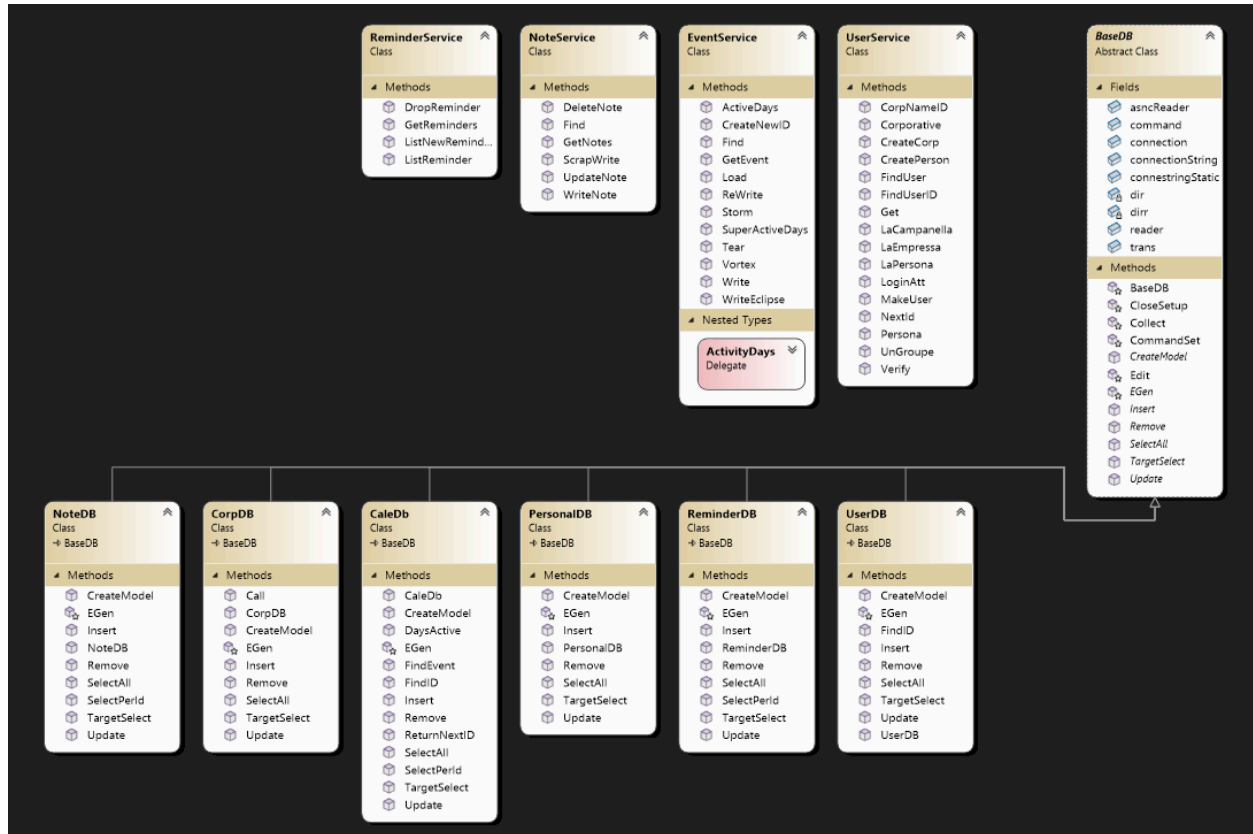
```

כפי שניתן לראות, בתזכורת אכן יש מאפיין משתמש (לקישור אל משתמשים), מאפיין חברה (מטרתו להעיד על מקור הפריט) וכמובן תוכן. לתזכורת יש את בנאי הבסיס והבנאי המפורט. שאר הפריטים (רשימה, אירוע) פועלים על מנגנון דומה, ותזכורת היא הדוגמה הבסיסית מביניהם (אף ניתן היה להגדיר מחלקה מסוג "פריט", אך לפי דעתי הדבר אינו היה מביא לערך רב).

כמו כן, ניתן לראות כי מאפיין חברה ריק בא לידי ביטוי גם ב- "-" וגם ב- "--", זוהי שגיאת ארגון מבחינתי, אך כמובן אוכל לתקן זאת בקלות אם אראה צורך (כעת 2 האפשרויות מקובלות).

שכבת ה- ViewModel

בשכבה הזאת קיימות המחלקות אשר מטפלות בנתונים, ומגשרות בין מסד הנתונים לשאר השכבות. פירוט בתרשים הבא.



המחלקה המרכזית היא **BaseDB**. מחלקה זו מיישמת את הכלים לשליפה והוספה של נתונים למסד הנתונים, ובא נמצאות הפעולות המשותפות לכל מחלקה המטפלת בנתון ספציפי, אשר הן המחלקות היורשות ממחלקה זו. פעולות מופשטות הקיימות במחלקה זו כגון **EGen** (יוצר עצם) מקבלות יישום ספציפי במחלקות היורשות. בנוסף, קיימות מחלקות אשר מקיימות אוסף פעולות כתבניות, הן מחלקות ה- **Service** שמשתמשות במחלקות ה- **DB** כעצמים.

מחלקות נבחרות

מחלקת BaseDB

```

public abstract class BaseDB
{
    private static string dir = Directory.GetCurrentDirectory();
    private static string dirr =
Path.GetDirectoryPath(Path.GetDirectoryPath(Path.GetDirectoryPath(Path.GetD
irectoryPath(Path.GetDirectoryPath(dir))));

    public string connectionString = @"\" + "Provider=Microsoft.ACE.OLEDB\" +
".12.0\" +
    \";Data Source=\" + dirr + "\\VModel!\\DB\\DatabaseTest1.accdb\" +
    \";Persist Security Info=True\";

    // a static connection string for the service, the first one is not
valid for it
    public string connestringStatic =
@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\\Users\\User\\Desktop\\VisiOn\\VisiOn\\VModel!\\DB\\DatabaseTest1.accdb;P
ersist Security Info=True\";

    public OleDbConnection connection;
    public OleDbCommand command;
    public OleDbDataReader reader;
    public System.Data.OleDb.OleDbDataReader asncReader;
    public OleDbTransaction trans;

    protected BaseDB()
    {
        // setup
        this.connection = new OleDbConnection(this.connestringStatic);
        this.command = new OleDbCommand();
        this.command.Connection = this.connection;
    }

    protected void CloseSetup()
    {
        if (reader != null) reader.Close();
    }

```

```

        if (connection.State == System.Data.ConnectionState.Open)
            connection.Close();
    }

    protected abstract Entity EGen(); // Generation of highest level
entity, such as (Entity)User; in specified class
    public abstract void CreateModel(Entity e); // Setup of the entity; in
specified class

// Returns all entities, internal command
protected List<Entity> Collect(string cmdTxt)
{
    command.CommandText = cmdTxt;
    List<Entity> el = new List<Entity>();

    try
    {
        command.Connection = connection;
        connection.Open();
        reader = command.ExecuteReader();
        Entity tmp = new Entity();

        while (reader.Read()) // each new reader line - each
progression
        {
            tmp = EGen();
            CreateModel(tmp);
            el.Add(tmp);
        }
    }
    catch (Exception ex)
    { Console.WriteLine(" Could not load database \n Error message: \n
" + ex.Message); } // print error if is
    finally // close
    {
        CloseSetup();
    }

    return el;
}

```



```

    }

    // The reflection of Collect in all specified classes, with a more self
    explanatory name
    public abstract List<Entity> SelectAll(string cmdTxt);

    // Returns target entity (mostly w/ Collect as well); not necessary in
    all classes
    public abstract Entity TargetSelect(int id, string cmdTxt);

    // The execution of Insert, Update & Remove commands
    protected async Task<int> Edit(string sqlStr = "emptyspace", int
    records = 0)
    {
        trans = null;
        try
        {
            //command.CommandText = sqlStr;
            //connection.Open();
            trans = connection.BeginTransaction();
            command.Transaction = trans;
            records = command.ExecuteNonQuery(); // action; + Async & await
for later
            trans.Commit();
        }
        catch (Exception e)
        {
            trans.Rollback();
            Console.WriteLine("Error message: \n " + e.Message);
        }
        finally
        {
            CloseSetup();
        }
        return records;
    }
    protected void CommandSet(string sqlStr)
    {
        command.CommandText = sqlStr;
        connection.Open();
    }

```

```

    public abstract Task<int> Insert(Entity e);
    public abstract Task<int> Update(Entity e0, Entity e1);
    public abstract Task<int> Remove(Entity e);
}

```

כפי שניתן לראות, כאן קיימות שמות הפעולות כגון Insert או EGen, וזוהי המחלקה שמקשרת בין הקוד למסד הנתונים (הקישור נעשה בבנאי, אשר רץ במחלקות היורשות). קיימים שני קטעי קישור, אחד סטטי ואחד דינמי (הסטטי נעשה לשם שכבת ה-Service), הפעולות EGen ו-CreateModel יוצרות את העצם בקוד (בעזרת נתוני בסיס הנתונים), פעולת Edit מריצה את פקודת ה-SQL לשם שינוי מסד הנתונים בחיבור לפעולות השינוי, אשר הם שלושת הפעולות האחרונות. בפעולות אלה חל שימוש בטרנזקציה לשם שמירת סדר בבסיס הנתונים. מחלקה זו, כמו מחלקות וקטעי קוד אחרים, עדיין איננה שלמה לחלוטין (ייתכן שיהיו שינויים) ולכן היא גם יחסית מבולגנת.

מחלקת NoteDB

```

public class NoteDB : BaseDB
{
    public NoteDB() : base() { }

    public List<NoteItem> SelectPerId(User user, string cmdTxt = "SELECT *
FROM Note_Tbl")
    {
        int id = user._absId;
        List<NoteItem> nl = new List<NoteItem>();
        List<Entity> el = base.Collect(cmdTxt);
        foreach (NoteItem n in el)
        { if (n._uid == id) nl.Add(n); }
        return nl;
    }

    public override void CreateModel(Entity e)
    {
        NoteItem n = e as NoteItem;
        n._uid = int.Parse(reader["Uid"].ToString());
        n._name = reader["NoteName"].ToString();
        n._text = reader["NoteText"].ToString();
        n._cid = reader["Corp"].ToString();
    }
}

```

```

    public override List<Entity> SelectAll(string cmdTxt = "SELECT * FROM
Note_Tbl")
    {
        return base.Collect(cmdTxt);
    }

    public override Entity TargetSelect(int id, string cmdTxt = "SELECT *
FROM Note_Tbl")
    {
        return new NoteItem(); // also unused
    }

    protected override Entity EGen()
    {
        return new NoteItem();
    }

    public override async Task<int> Insert(Entity e)
    {
        NoteItem n;
        if (e is NoteItem) n = e as NoteItem;
        else return -1;
        int records = 0;

        string sqlStr = string.Format($"INSERT INTO Note_Tbl (Uid,
NoteName, NoteText, Corp) VALUES (?, ?, ?, ?);");
        CommandSet(sqlStr);
        command.Parameters.AddWithValue("?", n._uid);
        command.Parameters.AddWithValue("?", n._name);
        command.Parameters.AddWithValue("?", n._text);
        command.Parameters.AddWithValue("?", n._cid);
        return Edit().Result;
    }

    public override async Task<int> Remove(Entity e)
    {
        NoteItem n;
        if (e is NoteItem) n = e as NoteItem;
        else return -1;
        int records = 0;

        string sqlStr = "DELETE FROM Note_Tbl WHERE (Uid = ? AND NoteText =

```

```

? AND NoteName = ?)";
    CommandSet(sqlStr);
    command.Parameters.AddWithValue("?", n._uid);
    command.Parameters.AddWithValue("?", n._text);
    command.Parameters.AddWithValue("?", n._name);
    return Edit().Result;
}

public override async Task<int> Update(Entity e0, Entity e1)
{
    NoteItem n0; NoteItem n1;
    if (e0 is NoteItem && e1 is NoteItem) { n0 = e0 as NoteItem; n1 =
e1 as NoteItem; }
    else return -1;
    int records = 0;

    string sqlStr = $"UPDATE Note_Tbl SET " +
        $" NoteName = ?, " +
        $" NoteText = ? " +
        $" WHERE (Uid = ? AND NoteText = ? AND NoteName = ?)";
    CommandSet(sqlStr);
    command.Parameters.AddWithValue("?", n1._name);
    command.Parameters.AddWithValue("?", n1._text);
    command.Parameters.AddWithValue("?", n0._uid);
    command.Parameters.AddWithValue("?", n0._text);
    command.Parameters.AddWithValue("?", n0._name);
    return Edit().Result;
}
}

```

זוהי המחלקה המטפלת בנתוני הרשימות. כמו שאר המחלקות מסוגה היא יורשת מ- BaseDB ומממשת את פעולותיה בהקשר המתאים. קיימות פעולות במחלקה זו שאין בהן צורך (TargetSelect). לפעולות אלה אין אלגוריתם תקין (במבט לאחור הייתי יכול להפוך פעולה זו לשימושית ולקצר את קוד המחלקה, אעשה זאת בהזדמנות).

ניתן לראות שחל שימוש בפרמטרים בפעולות מסוימות, זאת על מנת להגן מפני sql injection. מנגנון הפעולה פועל לפי כך שפקודה עם פרמטרים לא נקראת באותו האופן בו נקראת פקודה המהווה רק טקסט, וזה מונע בכך הרצת קוד אפשרי שהובא כקלט. הדבר בין השאר נעשה בהתאם להרחבה המתאימה.

מחלקת UserService

```
public class UserService
{
    // general login control on base form
    public static bool LoginAtt(string username, string pass, int type_ = 1)
    {
        User usr = FindUser(username);

        if (usr._usrName == username && usr._pwd == pass && (usr._type ==
type_ || usr._type == 3)) return true;
        return false;
    }

    public static User FindUser(string usrName) => (User)(new
UserDB().TargetSelect(new UserDB().FindID(usrName)));

    public static int FindUserID(string usrName) =>
FindUser(usrName)._absId;

    public static User Get(int id = 2) // get user with id
    {
        UserDB usrdb = new UserDB();
        User usr = (User)(usrdb.TargetSelect(id));
        if (usr._absId == id) return usr;
        else return null;
    }

    public static User MakeUser(string usrN, string pass, int ty)
    {
        User uu = new User(usrN, pass, NextId(), ty);

        new UserDB().Insert(uu);

        return uu;
    }

    public static int NextId()
    {
        List<Entity> u1 = new UserDB().SelectAll();
        int iid = 0;
        foreach (User u in u1)
        {
```

```

        if (iid <= u._absId) iid = u._absId;
    }
    iid++;
    return iid;
}

```

```

public static bool Verify(User u)
{
    if (u._absId > 0) return true; else return false;
}

```

```

public static Person Persona(User u) => (Person)(new
PersonalDB().TargetSelect(u._absId)); // get (person)u raw

```

```

public static Corp LaEmpressa(User u) => (Corp)(new
CorpDB().TargetSelect(u._absId)); // get (corp)u raw

```

```

public static Person LaPersona(User u, Person p = null) // returns the
full (besides type) person of user if valid | (Person)u
{
    Person pu = Persona(u);
    if (Verify(pu)) return pu;
    else return p;
}

```

```

public static Corp Corporative(User u, Corp cc = null) // returns if
the user is corp the corporation | (Corp)u
{
    Corp cu = LaEmpressa(u);
    if (Verify(cu)) return cu;
    else return cc;
}

```

```

public static Corp UnGroupe(Person p) => new CorpDB().Call(p._cid); //
returns the corp containing user if is

```

```

public static List<Person> LaCampanella(Corp c) // returns all people

```

in a corp

```
{
    List<Entity> el = new PersonalDB().SelectAll();
    List<Person> people = new List<Person>();
    foreach (Person p in el) { if (p._cid == c._cid) people.Add(p); }
    return people;
}
```

```
public static string CorpNameID(string cid) => new
CorpDB().Call(cid)._cName; // get name per id
```

```
public static void CreatePerson(User u, string f, string l, string c)
{
    Person p = new Person(u, f, l, c);
    new PersonalDB().Insert(p);
}

public static void CreateCorp(User u, string cid, string cn)
{
    Corp cc = new Corp(u, cid, cn);
    new CorpDB().Insert(cc);
}
}
```

מחלקה זו היא אוסף הפעולות הנחוצות לכלל המשתמשים. ללא מחלקות ה- Service קוד הממשקים היה ארוך בהרבה והיה הרבה יותר מקום לטעויות, ולשם כך יצרתי מחלקות אלו. הפעולות העליונות מתייחסות למשתמש הבסיסי, והן פעולת כניסה, יצירת משתמש חדש ומציאת משתמש. הפעולות מהמרכז ומטה מתייחסות גם למשתמש פרטי/חברה, ובהן החזרת משתמש מלא על סמך משתמש בסיס, החזרת כלל המשתמשים הקשורים לחברה, ועוד. אין להתייחס לשמות הפעולות הלא מסבירות, זו הייתה "בחירה עיצובית" בה הפכתי את הקוד לאישי יותר ובכך את תהליך העבודה לנעים יותר עבורי.

שכבת שירותי הרשת

שכבת שירותי הרשת, היא שכבת ה- Web Service, היא השכבה המקשרת בין הפרויקט לאפליקציות חיצוניות. שכבה זו חיונית לכל מערכת שפועלת בסגנון שרת-לקוח. השכבה בנוייה על WCF, ומתקשרת עם האפליקציה החיצונית באמצעות קריאות http. אופן הפעולה של השכבה הוא על ידי יצירת ממשק המשתמש ב- contracts, יישומו במחלקה ויישומה ב- host אשר כל עוד הוא רץ ניתן להתחבר אל השכבה באמצעות הכתובת שנתן לה. השכבה בפרויקט זה מבוססת על CoreWCF המותאם ל- .NET.

הגדרות הרשת וה- Service Host

```
public static void Main(string[] args)
{
    var builder = WebApplication.CreateBuilder(args);

    // Add WSDL support
    builder.Services.AddServiceModelServices().AddServiceModelMetadata();
    builder.Services.AddSingleton<IServiceBehavior,
    UseRequestHeadersForMetadataAddressBehavior>();

    var app = builder.Build();

    // Configure an explicit none credential type for WSHttpBinding as it
    defaults to Windows which requires extra configuration in ASP.NET
    var myWSHttpBinding = new WSHttpBinding(SecurityMode.Transport);
    myWSHttpBinding.Security.Transport.ClientCredentialType =
    HttpClientCredentialType.None;

    app.UseServiceModel(builder =>
    {
        builder.AddService<VisionService>((serviceOptions) => { })
        // Add a BasicHttpBinding at a specific endpoint
        .AddServiceEndpoint<VisionService, IVisionService>(new
    BasicHttpBinding(), "/VisionService/basichttp")
        // Add a WSHttpBinding with Transport Security for TLS
        .AddServiceEndpoint<VisionService, IVisionService>(myWSHttpBinding,
    "/VisionService/WSHttps");
    });

    var serviceMetadataBehavior =
```



```

app.Services.GetRequiredService<CoreWCF.Description.ServiceMetadataBehavior>
>();
    serviceMetadataBehavior.HttpGetEnabled = true;

    app.Run();
}

```

הרשת מוגדרת דרך פעולת ה-Main של ה-host כמתואר: באמצעות CoreWCF התוכנית קודם כל יוצרת בנאי, אשר יוצר תבנית, אשר מקשרת באמצעות הבנאי את השירות ויוצרת לו נקודות קצה (אחת ל-basichttp ואחת ל-wshttp), ולבסוף רצה. קטע קוד זה הובא ממקור חיצוני ואין לי הבנה מוחלטת שלו, לכן לא אוכל להסביר הרבה מעבר לכך.

בזמן בו תוכנית זו רצה, בזמן שנקודות הקצה פתוחות, ניתן להתחבר לשירות דרך נקודות קצה אלו. החיבור לשירות בפרויקט נעשה דרך service references של Visual Studio, אשר בטעינת השירות יצר "העתק" של קריאות פעולותיו באפליקציה, להם האפליקציה קוראת ובכך משתמשת בשירות (פעולות אלה קוראות לשירות עצמו).

ממשק ה-Service ומימוש

ממשק

```
[ServiceContract]
public interface IVisiOnService
{
    [OperationContract]
    string Status();

    [OperationContract]
    bool Login(string Uname, string Upass, int T);

    [OperationContract]
    void CreatePerson(string cid, string fName, string lName, string un,
string pd, int id);

    [OperationContract]
    List<string> RemindersString(int id);

    [OperationContract]
    List<Reminder> Reminders(int id);

    [OperationContract]
    int GetID(string n);

    [OperationContract]
    List<string> EventsString(int id);

    [OperationContract]
    List<Event> Events(int id);

    [OperationContract]
    List<string> NotesString(int id);

    [OperationContract]
    List<NoteItem> Notes(int id);

    [OperationContract]
    List<Person> CorpUsers(int id);

    [OperationContract]
    List<string> CorpUsersString(int id);
```

```
}
```

נכון לזמן זה בשירות קיימות פעולות סטטוס (נעשתה לצורך בדיקת ריצת השירות), פעולת כניסה, פעולות המחזירות פריטים ושמותיהם, פעולה המוצאת מאפיין משתמש על פי שמו, ופעולה שיוצרת משתמש פרטי.

מימוש

```
public class VisionService : IVisionService
{
    public string Status() => "Running";

    public bool Login(string Un, string Up, int T) =>
UserService.LoginAtt(Un, Up, T);

    public int GetID(string uName) => UserService.FindUserID(uName);

    public void CreatePerson(string cid, string fName, string lName, string
un, string pd, int id = 0)
    {
        Person p = new Person(cid, fName, lName, un, pd, id);
        User u = (User)p;
        User uu = UserService.MakeUser(u._usrName, u._pwd, 1);
        UserService.CreatePerson(uu, p._fName, p._lName, "-");
    }

    public List<string> RemindersString(int id)
    {
        List<Reminder> l = Reminders(id);
        List<string> result = new List<string>();
        foreach (Reminder r in l) { result.Add(r._text); }
        return result;
    }

    public List<Reminder> Reminders(int id)
    {
        User u = UserService.Get(id);
        List<Reminder> l = ReminderService.GetReminders(u);
        return l;
    }
}
```

```

public List<Event> Events(int id)
{
    User u = UserService.Get(id);
    var ll = EventService.Load(u);
    List<Event> l = new();
    foreach (var i in ll) { l.Add(i); }
    return l;
}

public List<string> EventsString(int id)
{
    var lst = Events(id);
    List<string> result = new();
    foreach (var i in lst) { result.Add(i._name); }
    return result;
}

public List<NoteItem> Notes(int id)
{
    User u = UserService.Get(id);
    var ll = NoteService.GetNotes(u);
    List<NoteItem> l = new();
    foreach (var i in ll) { l.Add(i); }
    return l;
}

public List<string> NotesString(int id)
{
    var lst = Notes(id);
    List<string> result = new();
    foreach (var i in lst) { result.Add(i._name); }
    return result;
}

public List<Person> CorpUsers(int id)
{
    User u = UserService.Get(id);
    Corp c = UserService.Corporative(u);
    var l = UserService.LaCampanella(c);
    return l;
}

public List<string> CorpUsersString(int id)

```

```

{
    var lst = CorpUsers(id);
    List<string> result = new();
    foreach(var i in lst) { result.Add($"{i._fName} {i._lName},
{i._absId}"); }
    return result;
}
}

```

רצוי להוסיף שאין צורך לרשום Async בסוף שם פעולה (כגון LoginAsync) אל אף שהפעולה היא אכן אסינכרונית בצד הלקוח, Visual Studio עושה זאת אוטומטית בבניית השירות (בצד הלקוח).

שירותי רשת חיצוניים

בנוסף לשירות שיצרתי, השתמשתי בפרויקט בשירות חיצוני Random User API כדי לאפשר יצירת משתמשים בצורה קלה יותר (וכן כדי ליישם את הרחבת שירותי הרשת החיצוניים). לא השתמשתי בשירות זה כפי שהשתמשי בשירות שלי, אלה קראתי לו מתוך פעולה באפליקציה:

```

public async Task<List<string>> Create()
{
    string apiUrl = "https://randomuser.me/api/?results=1&nat=us"; // 1
    person (from us)
    {
        string fn = "";
        string ln = "";
        string u = "";
        string p = "";
        List<string> list = new List<string>();

        using (HttpClient client = new HttpClient())
        {
            HttpResponseMessage response = await client.GetAsync(apiUrl);
            if (response.IsSuccessStatusCode) // if there is a result
            {
                string jsonResponse = await
response.Content.ReadAsStringAsync();

                // Parse
                RootObject0j rootObject =
JsonConvert.DeserializeObject<RootObject0j>(jsonResponse);

```

```
// loop for the case of more than 1 person
foreach (var result in rootObject.results)
{
    //l1l1 += ("Name: {result.name.first} {result.name.last};
    {result.login.username} {result.login.password}");
    fn += result.name.first; ln += result.name.last; u +=
    result.login.username; p += result.login.password;
    // since there is only 1 result there is no meaning in
    changing it
}
list.Add(fn); list.Add(ln); list.Add(u); list.Add(p);
}
else
{
    Console.WriteLine("Error fetching data.");
}
}

return list;
}
```

בפעולה ניתן לראות את הקריאה ב- `response = client.GetAsync`, ופעולה זו (`Get`) מחזירה ישירות מן השירות נתוני משתמש רנדומלי. שאר הפעולה היא הקריאה של הנתונים ויישומם בממשק המשתמש. נא להתעלם מהעובדה שהקוד מאוד מבולגן, לא עבר זמן רב לאחר סיום חלק וקוד זה עדיין לא סודר מאז שלבי ניסוי וטעייה.

צד לקוח

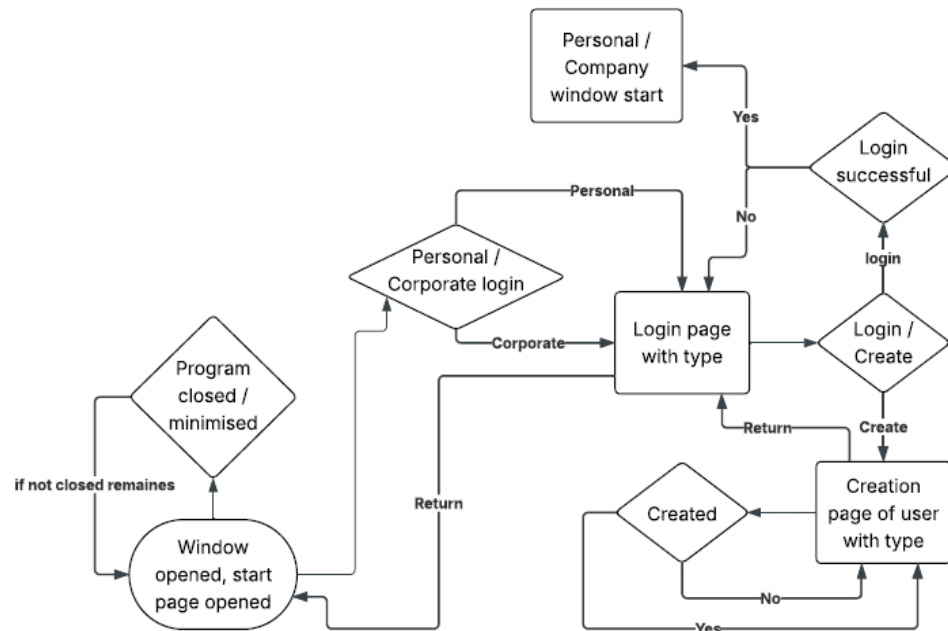
בצד הלקוח קיימת שכבה אחת והיא שכבת View. בשכבה זו שני ממשקי לקוח, אחד מהם הוא אפליקציה לוקלית (מתחברת ישירות ל- ViewModel) והיא בנוייה ב- WPF, ושני הוא אתר שנבנה ב- Blazor אשר מתחבר לשרת באמצעות שכבת ה- Service. בחלקים הבאים מתוארים ממשקי הלקוחות בצירוף הוראות שימוש לגבי כל חלק, כך שקודם יוצג החלק באופן כללי עם הוראות השימוש ולאחריו יוצג הסבר קצר על הקוד. דבר שנכון גם לגבי חלקים אחרים בספר הוא שייתכן ואפספס מידע חשוב שיש להסבירו בין אם הוא נדמה מובן מאליו או מכמות המידע הרבה, לכן להבנה מיטבית יש להסתכל בקובץ הפרויקט עצמו.

ממשק משתמש ראשון - WPF

אפליקציה זו רצה במחשב הפרטי ופועלת על מערכת הפעלה Windows, והיא תוכנה חלונאית שעוצבה בעיצוב רב חלונאי באמצעות Windows Presentation Foundation. האפליקציה נפתחת בחלון כניסה עם מסכים המאפשרים כניסה ויצירת משתמשים, ולאחר הכניסה ניתן להגיע לחלון משתמשים פרטיים וחלון משתמשי חברה. לכל אחד מחלונות אלו אפשרויות בהתאם לרמת ההרשאה לצפייה, הוספה, שינוי או מחיקה של פריטים תוך כדי שהייה בסביבה נוחה. מבחינת עיצוב, עיצוב האפליקציה לא הושלם בשלב זה והעיצוב הוא ברמה בסיסית. דבר זה נכון גם לגבי האתר. האפליקציה נבדקה על מחשבים נייחים בעלי מערכת הפעלה Windows (דבר זה נדרש בשל WPF) בגרסה 10 ו- 11, ומחשב נייד בעל Windows 10. ממשק משתמש זה מתחבר לשרת באופן מקומי ללא מעבר מידע בשכבת ה- Service.

חלון כניסה

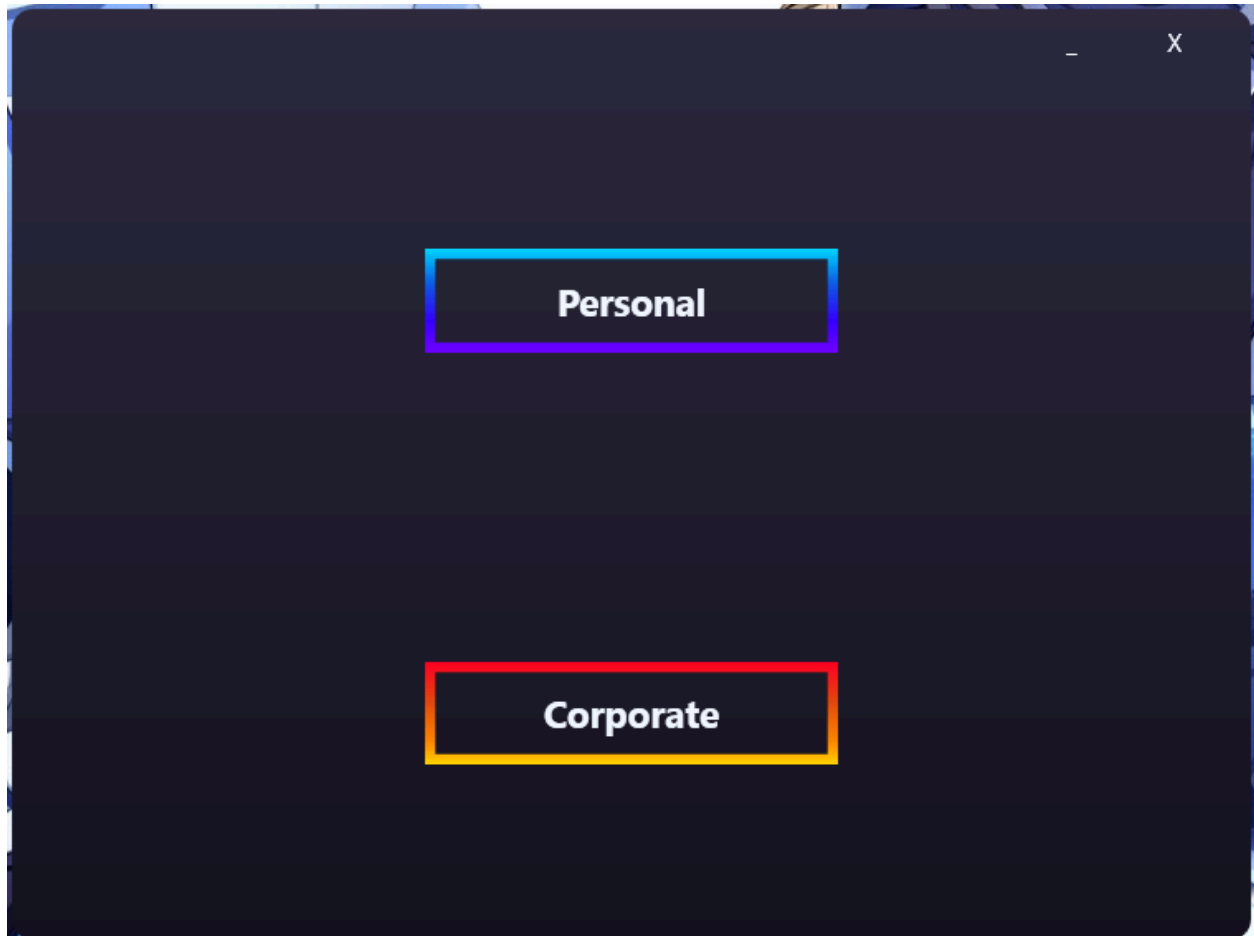
בהרצת האפליקציה נפתח חלון הכניסה המהווה מסגרת לדפי הכניסה והיצירה. לחלון יש כפתור מזעור וכפתור סגירה.
מצורף תרשים תהליכים מפורט של תהליכי החלון.



חלון הכניסה הנפתח הוא LoginW.xaml בתצוגתו והוא רץ מתוך LoginW.xaml.cs. אופן פעולה זה משותף לכל המסכים והדפים בממשק: כל חלק כולל קובץ תצוגה xaml וקובץ קוד cs.
עיצוב החלון הוא עיצוב ידני, לכן הפעולות הבסיסיות כגון סגירה או מזעור הועברו לכפתורים שנוצרו על ידי (מהכפתורים הכלולים ב- WindowStyle אשר עכשיו ריק). פעולות אלה קיימות בפונקציות: Quit_Click, Drag, Mini_Click, Maxi_Click אשר בהתאם מקיימים את פעולות הגרירה, מזעור, חזרה ממזעור, מסך מלא, חזרה ממסך מלא וסגירה.
בקוד החלון קיים גם בנאי החלון, אשר לאחר InitializeComponent (שפותח את התצוגה) פותח את דף ההתחלה StartPage בתוך מסגרת (Frame) הקיימת בחלון.

דף פתיחה

בעת פתיחת החלון נפתח בתוכו דף הכניסה הראשוני בעל שני כפתורים, כפתור לכניסת משתמש פרטי וכפתור לכניסת משתמש חברה. לחיצה על אחד מהכפתורים תעביר לדף הכניסה לכניסה המתאימה.



בתמונה ניתן לראות את חלון הכניסה כאשר דף זה פעיל. הכפתורים מלמעלה הם כפתורי הסגירה (X) והמזעור (_), והכפתורים במרכז מעבירים לדף הכניסה. כל אחד מהכפתורים מופעל בלחיצה יחידה.

בדף הכפתורים (Button) כמו כל האלמנטים בתצוגה מעוצבים לפי אחד הקבצים מתיקיית קבצי העיצוב. קבצים אלה מקושרים ב- ResourceDictionary.MergedDictionaries בקובץ App.xaml. בנאי הדף מקבל קישור לשני עצמים שהם מסגרת (Frame) ומסך (Window), שהם מסך הכניסה ומסגרתו במקרה זה.

בדף שני פעולות התואמות לכפתורים והם SetLoginP ו- SetLoginC אשר פותחים את דף הכניסה. פעולות אלה הם בעצם זהות, רק שקוראות לבנאי דף הכניסה עם סוגים שונים.

דף כניסה

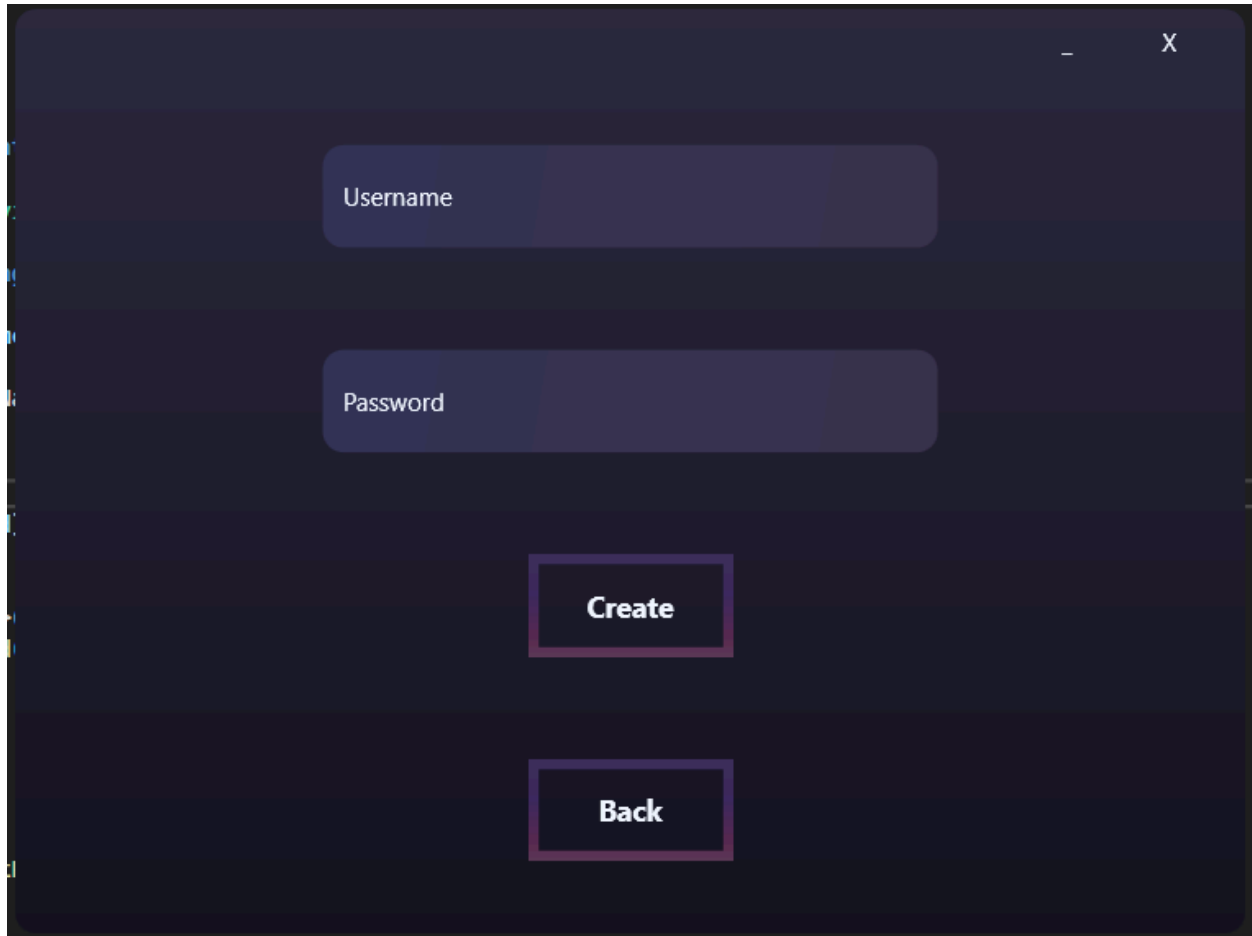
בדף הכניסה ישנם שני תיבות טקסט ושלושה כפתורים. תיבת טקסט אחת היא לשם משתמש, ותיבת טקסט שנייה היא לסיסמה. לחיצה על תיבה תבחר אותה ויהיה ניתן להוסיף טקסט (מנסיבות טכניות הטקסט יהיה מוסף מתחילת טקסט קיים, ומסוף טקסט אשר הוסף), ושני לחיצות ינקו את כל הטקסט שבתיבה. אין מגבלה למספר תווים, אך מסד הנתונים מוגבל בשדות אלו ל- 255 תווים. כפתור אחד הוא כפתור ליצירה אשר מעביר לדפי היצירה לפי הסוג שנבחר (נבחר בהתחלה), כפתור שני הוא כפתור כניסה הבודק את הנתונים שהוכנסו לתיבות ובמידה והם נכונים מכניס לחלון פרטי/חברה לפי סוג הכניסה (נבחר בהתחלה), והכפתור השלישי הוא כפתור חזרה שמחזיר לדף ההתחלה. כל הכפתורים מופעלים בלחיצה.

הטקסט המופיע הוא טקסט הקיים בפתיחה, והוא גם מהווה את נתוני משתמש הבדיקה (סוג 3, קיים גם כמשתמש חברה וגם כמשתמש פרטי) וגם מראה את שדה התיבה (מטרתו, במידה והעיצוב ישודרג טקסט זה יוחלף מטקסט אמיתי לטקסט נעלם).

בדף תיבות הטקסט והכפתורים הם מסוג `TextBox` ו- `Button` בהתאם. בנאי הדף מקבל 3 עצמים והם מסגרת, חלון ומספר. החלון והמסגרת מקשרים לחלון הכניסה ולמסגרתו (אופן פעולה זה חוזר גם ברוב הדפים האחרים) והמספר מהווה את סוג הכניסה. כמובן, כל הקלט נכנס לתכונות המתאימות בעצם הדף. בדף כמה פונקציות: `TextBoxClear` שמנקה את תיבות הטקסט מטקסט, `usrNew_Click` השולחת לדף היצירה, `back_Click` שמחזירה לדף הקודם ו- `Start` שלוקחת את קלט התיבות ומנסה כניסה באמצעות `UserService.LoginAtt` ואז פותחת את החלון המתאים לפי סוג הכניסה בהצלחתה.

דפי יצירה

דף היצירה הראשון אחראי על יצירת משתמש הבסיס. בדף זה קיימות שתי תיבות טקסט לשם משתמש ולסיסמה, כפתור ליצירה וכפתור לחזרה.



תיבות הטקסט והכפתורים פועלים כמו תיבות הטקסט והכפתורים בדף הקודם. לאחר יצירת משתמש הבסיס נפתח דף יצירה לחלק הפרטי/חברה של המשתמש. אם חלון המערכת נסגר בשלב היצירה של המשך המשתמש נוצר משתמש בסיס בלבד (כיוון שהדבר כבר נעשה בשלב הקודם). למשתמש זה אין נתונים פרטיים במסד הנתונים. כיוון שלא הוספתי אופציה לשנות נתונים אלו לאחר יצירת המשתמש, הדבר אינו מומלץ.

בנאי הדף מקבל את תכונות הדף הקודם, שהם החלון, המסגרת והסוג. בדף מלבד פונקציית ניקוי הטקסט ופונקציית החזרה קיימת פונקציית היצירה, אשר קודם יוצרת עצם User (לפי המודל) ב- UserService.MakeUser שבו זמנית מכניסה אותו למסד הנתונים, ופותחת את הדף השני עימו ועם הסוג.

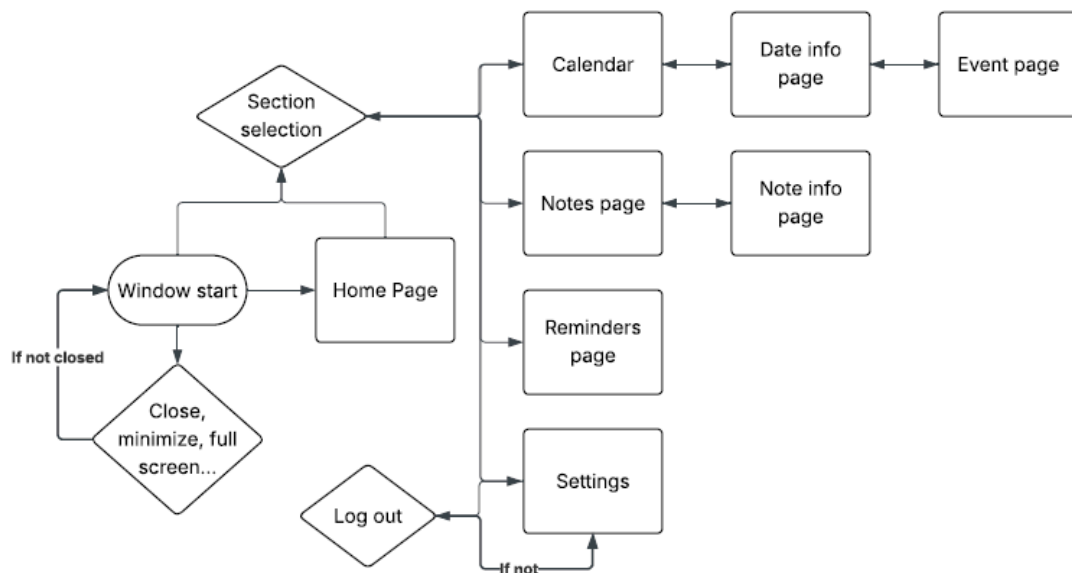
The image shows a dark-themed user creation form. It has three input fields: 'First Name' and 'Last Name' on the left, and 'Cid (--)' on the right. At the bottom right, there is a 'Create' button. The form is set against a dark background with light-colored text and input fields.

זהו הדף בהתאם ליצירת משתמש פרטי, הדף נראה כמעט זהה (מלבד כמות תיבות טקסט ומשמעותן) ביצירת משתמש חברה. לאחר תהליך היצירה המערכת חוזרת לדף הכניסה (לדף ההתחלה, ולא כמו בתרשים לדף יצירת המשתמש. תוכן היה לחזור לדף כניסת המשתמש, אך האופציה השנייה הייתה קלה יותר לביצוע ולכן נבחרה).

הדף מתחלף לפי חלק משתמש פרטי וחלק משתמש חברה באמצעות Visibility. בכל חלק תיבות הטקסט והכפתורים המתאימים (כאמור, כמעט אין הבדל בין החלקים).
בנאי הדף מקבל חלון, עצם משתמש (User) וסוג (ניתן לקצר את הקוד כיוון שהסוג כבר קיים במשתמש). הבנאי פותח את החלק המתאים לפי הסוג.
פונקצית SetUp היא זו שאחראית על היצירה, והיא נקראת על ידי פונקציות כפתורי היצירה (היא גם הפונקציה היחידה הקיימת מלבד פונקציות הכפתורים ופונקציות ניקוי הטקסט). פונקציה זו יוצרת את החלק הנוסף של המשתמש באמצעות UserService.CreatePerson ו- UserService.CreateCorp (אינני מסביר על פעולות השכבות האחרות כיוון שניתן לחקור אותן בקובץ הפרויקט ובנספחים), ואז היא פותחת חלון כניסה חדש וסוגרת את הנוכחי.

חלון משתמש פרטי

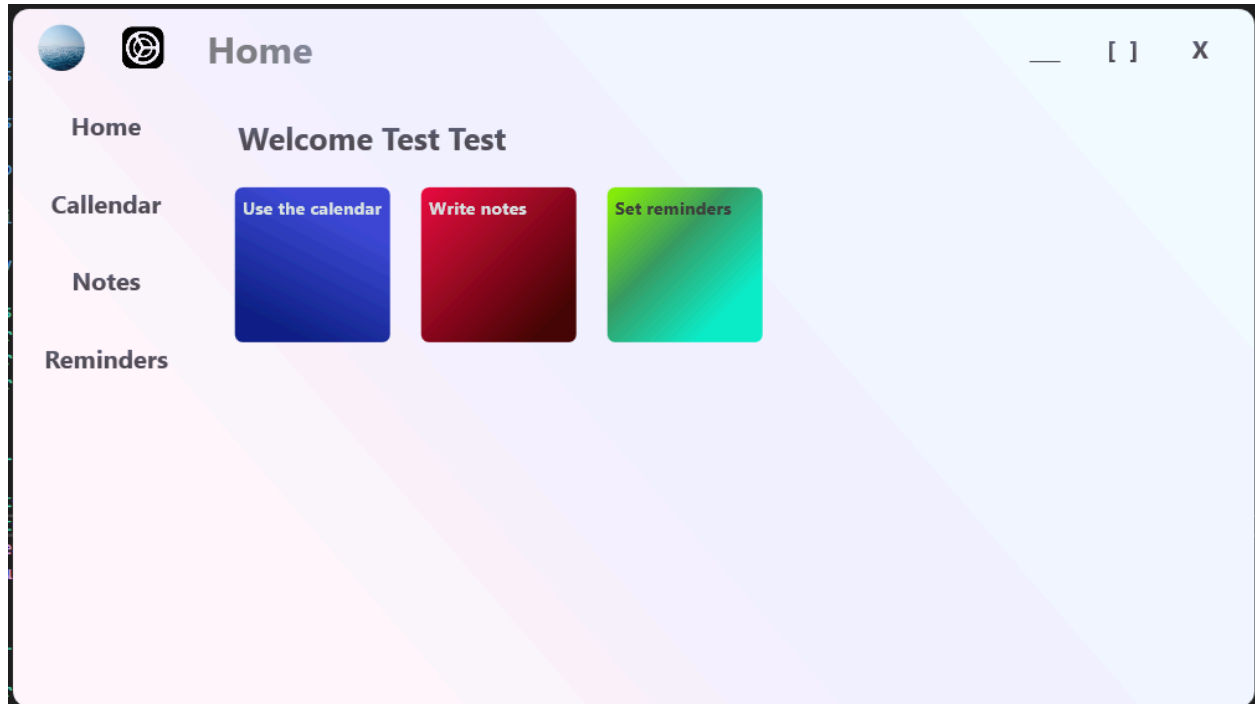
לאחת כניסה מוצלחת מדף הכניסה, המערכת פותחת את חלון המשתמש הפרטי במידה וזהו הסוג הנבחר (אחרת נפתח חלון משתמש חברה). בחלון זה ישנם 3 כפתורים האחראים על סגירה/מזעור/מסך מלא בפינה הימנית העליונה של החלון, 2 כפתורים המעבירים להגדרות ולפרטי משתמש בפינה השמאלית העליונה (כרגע אין הגדרות והכפתור פותח אזור ריק) הנראים כתמונת פרופיל (תוכן להוסיף אפשרות לתמונת פרופיל, לא יצא לפועל) וסמל הגדרות, ותחתם סרגל כפתורים בשמאל החלון המעבירים בין הדפים. בכניסה נפתח דף הבית אוטומטית.



זהו התרשים המתאר את המעברים בין החלקים בחלון. בתרשים חלק פרטי המשתמש וחלק ההגדרות שולבו לאחד.

בחלון ישנם ארבעה חלקים (ב- Grid ראשי): אחד למסגרת הדפים, אחד לכפתורי הבסיס של החלון (כמו בחלון הכניסה ולטקסט מצב, הפעולות הבסיסיות הועברו לכפתורים שבחלון כתוצאה מעיצוב אישי), אחד לחלק של תמונת הפרופיל וסמל ההגדרות והגדרות ופרטים הפותחת את חלק הפרטים וההגדרות (על ידי Visibility, החלק נפתח במקום המסגרת), ואחד לכפתורי מעבר הדפים. כל הכפתורים בארבעת חלקים אלו הם מסוג RadioButton לתיאום העיצוב, מלבד תמונת הפרופיל וסמל ההגדרות שהם תמונות (בעלות קישור לפונקציה ב-MouseDown).

בנאי החלון מקבל משתמש (User), מחליף לחלק המסגרת בפונקציית Section (המחליפה Visibility), ומפעיל את UserSetup אשר בעזרת המשתמש ומסד הנתונים יוצרת עצם משתמש פרטי (Person, בעזרת UserService.LaPersona) ומציגה את הנתונים במקום המתאים בתצוגה. ישנם פונקציות לפעולות הבסיס של החלון, פונקציה ליציאה (כמו סגירה אך עם פתיחת חלון כניסה), פונקציות למעבר (הפותחות במסגרת את הדף המתאים) ופונקציות מעבר החלקים מחלק הפרטים למסגרת ולהגדרות (המחליפות Visibility).



כאן ניתן לראות את חלון המשתמש הפרטי כאשר פתוח דף הבית.

דף בית

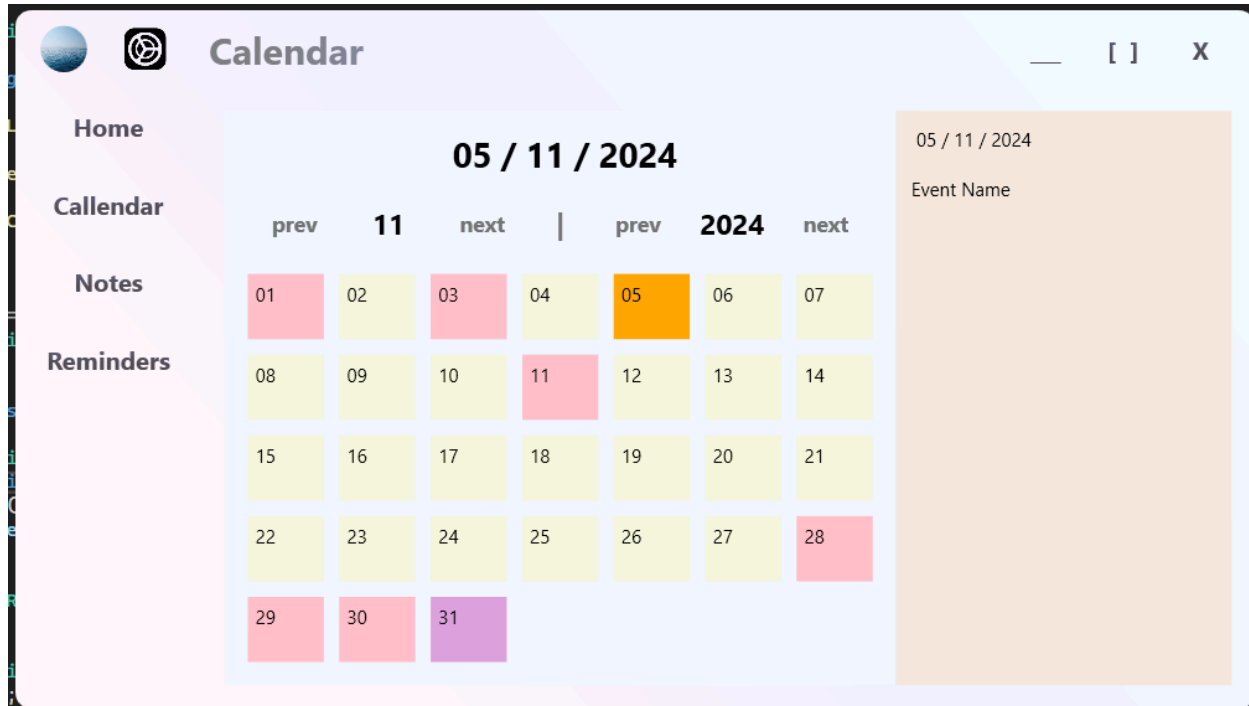
כפי שניתן לראות בתמונה האחרונה, בדף הבית קיימים כעת שלושה אזורים המספרים על אפשרויות המערכת. לחיצה כפולה עליהם תעביר לדף המתאים.

באופן כללי, דף זה שגם כן לא הושלם הוא הדף הראשי שאמור להציע פעולות מהירות או לתת דוח מצב. ערכו הראשי היה אמור להיות ערך אסתטי, אך עד כה הוא עדיין לא עוצב עד הסוף. דבר אחד שניתן להוסיף על הדפים הבאים הוא שכולם נוצרו ב-mal בסיסי ללא תוספות או NuGets, ולכן מנסיבות טכניות הם אינם מושלמים.

כל הקיים בדף הבית זה טקסט (Label) קבלה ושלושה אזורים מעוצבים כבתמונה (Border) להם מקושרת פעולת ניווט ב-MouseDoubleClick.

דפי לוח שנה

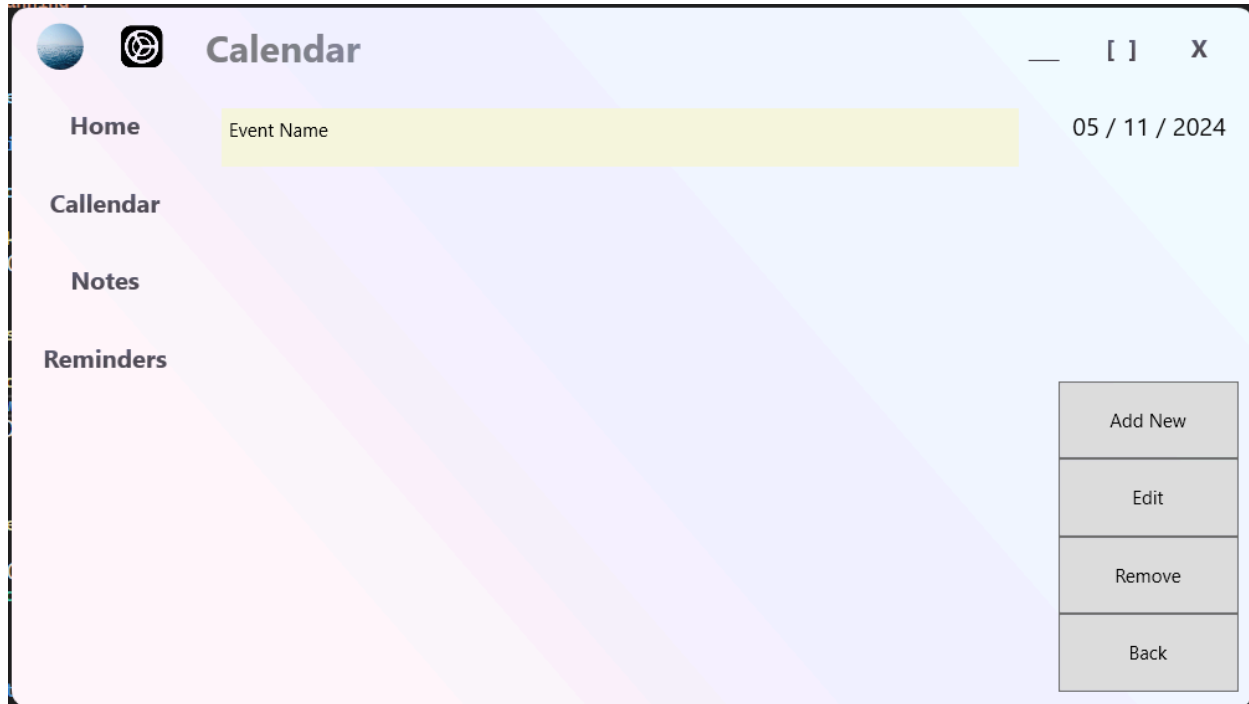
הדף הראשי של לוח השנה הוא לוח שנה פשוט בעל 31 מקומות (גם לחודשים בעלי פחות ימים) המסמלים ימים ואזור לשמות הפעילויות של כל יום נבחר.



בלחיצה על יום מסוים צבעו משתנה ואזור הטקסט מציג את אירועי אותו יום. צבע בהיר מסמן יום ללא פעילויות, צבע ורוד מסמן יום עם פעילויות אישיות, וצבע סגול מסמן יום עם פעילות אחת או יותר שחברת אותו משתמש הוסיפה לו. ניתן לנוע בתאריכים בלחיצה על כפתורי קדימה אחורה הנמצאים מעל אזור הימים, וכמובן בלחיצה על ימים. לחיצה כפולה על יום מסוים מעבירה לדף הפעילויות של אותו יום.

הדף כולל שני אזורים מסוג Border בהם אחד שוייך להצגת נתוני יום ואחד ללוח השנה. באזור הנתונים קיים טקסט Label המשתנה בהתאם. בלוח השנה יש Label המציג את התאריך, שני טקסטים Label המראים את החודש את השנה וכל אחד מהם כפתור המשנה את הנתון, ושלושים ואחת מקומות שהם בניגוד לנראה מסוג RadioButton (עיצובם יחסית מורכב והוא משנה את צבע האזור שלהם בהתאם להאם הם לחוצים או נושאי נתון).

הבנאי מקבל מסגרת ו-User, יוצר מערך חודשים ומתקין את הימים (יש לכך מערך ומחלקה נפרדת). יש פונקציות המשנות את ערך חודש ושנה, פונקציה המשנה את התאריך בהתאם, פונקציה המשנה את ערך טקסט היום בהתאם, ופונקציות לקביעת צבע היום בהתאם לנתוניו (צבע בלחיצה מתקבל מהעיצוב ב-Trigger). קיימת גם כמובן פונקצית מעבר לדף הימים, המקושרת לימים ב-MouseDownClick.



דף זה נפתח לאחר לחיצה כפולה על יום מסוים, בו ניתן לבחור בפעילות ולשנות/למחוק אותה, או להוסיף פעילות חדשה. הבחירה נעשית בלחיצה יחידה על פעילות (נפתחת הודעה עם פרטי הפעילות). כפתורים מימין אחראים על הוספה, מחיקה, שינוי וחזרה לדף הקודם. לא ניתן למחוק או לשנות פעילות שלא נבחרה.

בדף זה יש אזור לכפתורים ואזור לפעילויות. דף זה אינו שלם מבחינת פעילויות רבות, בניגוד לדף הרשימות או התזכורות לו אין אפשרות להציג את כל הפעילויות של המשתמש באותו יום והוא יציג רק את הראשונות. ככל הנראה אוסיף לו את אותו המנגנון של דפי הרשימות והתזכורות בהזדמנות. מנגנון ההצגה מבוסס על Grid שבכל חלק מופיע Label הנושא שם פעילות ופעולת מעבר מקושרת. בנאי הדף מקבל משתמש מסגרת ותאריך (התאריך הוא מחרוזת), ומוסיף את הפעילויות ללוח בעזרת פונקציות Inprint ו-AddItem המבוססות על Grid.Children.Add. פעולות אלה יוצרות Label, מוסיפות לו את שם הפעילות ומוסיפות אותו למקום הנכון ב-Grid. ישנם גם פעולות מתאימות לכל כפתור בחלק הכפתורים, פעולות שינוי הפעילויות מנווטות לדף השינוי הבא בהקשר לפעילות הנבחרת (בפעולת label_MouseDown). ישנה גם פעולת טעינה מחדש (היא פותחת את הדף מחדש עם נתוניו).

The screenshot shows a 'Calendar' application window. The title bar includes a blue globe icon, a circular icon, and the text 'Calendar'. The sidebar on the left lists 'Home', 'Calendar', 'Notes', and 'Reminders', with 'Calendar' being the active selection. The main content area features two text input fields labeled 'Event Name' and 'Details'. At the bottom of the window are two buttons, 'Cancel' and 'Save'.

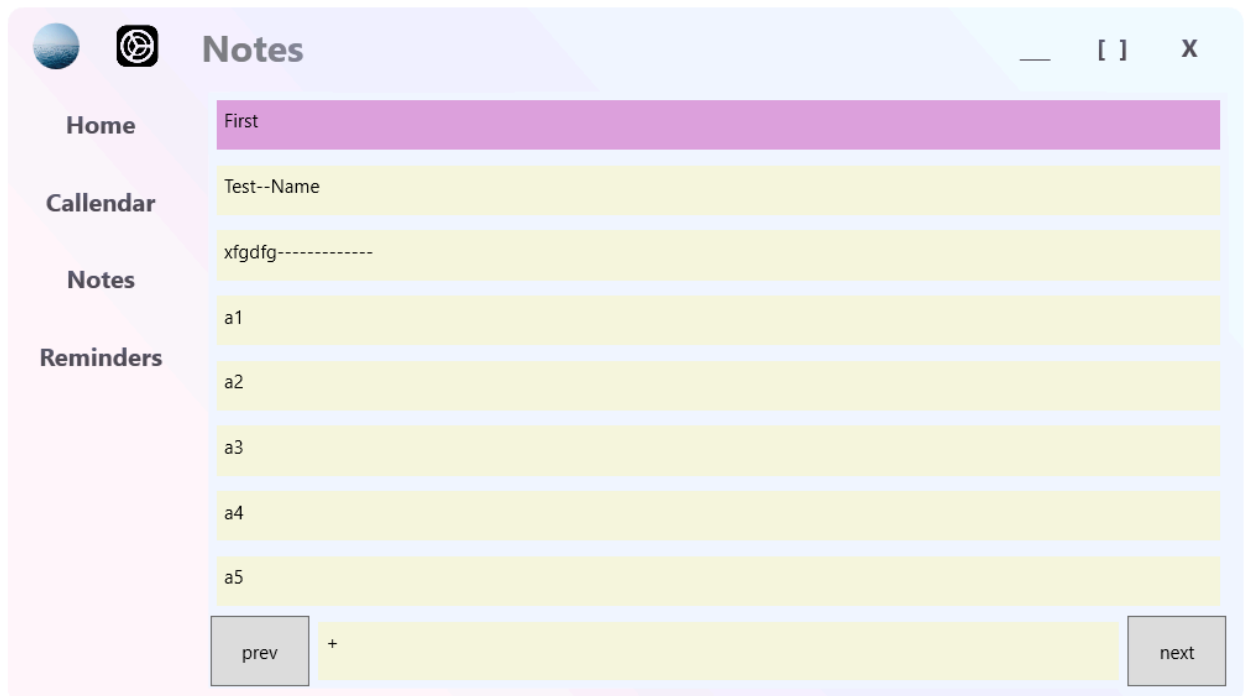
לאחר בחירת הוספה או שינוי פעילות נפתח דף השינוי, בו כאשר נלחץ כפתור השמירה הנתונים שבתיבות נשמרים במסד הנתונים (כפתור הביטול מבטל כל שמירה וכמו הכפתור השני מחזיר לדף הפעילויות). דף זה גם משמש לצפיית פרטי הפעילות (מוגבל כמו השם ל- 255 תווים). במידה וההגעה ודף נעשתה על ידי כפתור הוספת הפעילות הדף מוסיף פעילות חדשה, ובמקרה השני הדף משנה נתונים קיימים.

בנאי דף זה מקבל משתמש, מסגרת, תאריך, ופעילות (Event), ובמידה והפעילות אינה ריקה מציג את נתונה. שני פונקציות הכתורות מחזירים לדף הקודם (בעזרת המסגרת, התאריך והמשתמש שקיבלו), וההבדל ביניהם הוא פעולת השמירה.

פעולת השמירה הועברה לשכבת ViewModel וכעת היא `EventService.WriteEclipse`. פעולה זו מקבלת משתמש, תאריך, פעילות ואת נתוני תיבות הטקסט. במידה והפעילות קיימת, נובע מכך כי מטרת הדף היא עדכון והפונקציה מכניסה את נתוני התיבות בלבד לפעילות וקוראת לעדכון הפעילות במסד הנתונים. אם הפעילות לא קיימת (מטרת הדף היא הוספה) הפונקציה יוצרת פעילות חדשה בעזרת התאריך, נתוני המשתמש ונתוני התיבות, ומוסיפה אותה למסד הנתונים. כיוון שפעולה זו נקראת בפונקציית כפתור השמירה, ישר לאחריה נפתח הדף הקודם (עם נתונים מעודכנים).

דפי רשימות


הדף הראשי של הרשימות מראה את שמות הרשימות ברשימה.



הצבע הסגול מסמן את הרשימות שהחברה הוסיפה למשתמש. לחיצה על כפתורי קדימה/אחורה תעביר בין חלקי רשימת הרשימות, ולחיצה כפולה על הרשימה המקוצרת עם ה- "+" תוסיף רשימה חדשה בהינתן שמירה. לחיצה על רשימה מסוימת כמו לחיצה כפולה על כפתור ההוספה תפתח את דף הרשימה.

מנגנון הדף זהה גם למנגנון דף התזכורות. הדף מחולק לחלק הפעולות ולרשימה, בחלק הרשימה מוצגות תבניות מסוג Label המתארות את הרשימות לפי רשימה (הכוונה ל- `List<NoteItem> _notes`): מכל הרשימות ברשימה, מוצגות אלה הנמצאות באינדקס בין מספר המוכפל ב- 8 למספר העוקב לו המוכפל ב- 8 כאשר אותו מספר הוא מספר העמוד, כך שבכל עמוד יוצגו ב- 8 Grid תבניות. הכפתורים מלמטה הם האחראים לשינוי מספר העמוד, כאשר המספר המינימלי הוא 0. העמוד נטען בשינוי מספר העמוד וטעינתו נעשית בניקוי ה- Grid מרשימות ויצירת תבניות חדשות לפי אינדקס רשימתם.

על יצירת תבנית אחראית הפונקציה `AddItem` (שמבוססת על `Children.Add`) ועל טעינת עמוד מסוים אחראית הפונקציה `DrawTable`. כפתור ההוספה טוען את העמוד הבא ביחד עם נתוני עמוד זה (משתמש, מסגרת). כמו בפונקציה הוספת רשימה חדשה הקשורה לכפתור ההוספה בלחיצה על רשימה נטען העמוד הבא אך גם עם הרשימה שנבחרה.



בהינתן וזו רשימה קיימת (כמו זו) כפתור המחיקה ימחק אותה, וכפתור השמירה ישמור את נתוניה. בהינתן וזו רשימה חדשה כפתור השמירה יוסיף אותה כרשימה חדשה ואילו כפתור המחיקה לא יעשה דבר. כפתור הביטול מבטל את כל השינויים ומחזיר לדף הקודם, והכפתורים האחרים מחזירים לדף הקודם לאחר פעולותיהם.

בנאי דף זה מקבל מלבד משתמש ומסגרת גם רשימה. הבנאי טוען את נתוני הרשימה במידה והיא כבר קיימת, ואם מדובר בהוספת רשימה אז הבנאי אשר קיבל רשימה חדשה טוען את נתוניה. פונקצית כפתור החזרה רק טוענת במסגרת את הדף הקודם, פונקצית המחיקה מוחקת את הרשימה הטעונה ואז טוענת את הדף הקודם (אם זוהי רשימה חדשה - רשימה שאינה קיימת פעולת המחיקה לא מצליחה למצוא רשימה אשר לה מאפיין משתמש 1- כיוון שכל המאפיינים גדולים מ- 0 ולכן לא עושה דבר), ופונקצית השמירה מעדכנת את הרשימה אם היא קיימת ואם היא חדשה (מאפיין משתמש לא חוקי) היא טוענת בה את נתוני המשתמש בנוסף לנתוני התיבות (שנטענים בשני המקרים) ומוסיפה אותה למסד הנתונים, ולאחר מכן טוענת את הדף הקודם.

דף תזכורות

דף זה אחראי על כל פעולות התזכורות, שהן צפייה, הוספה ומחיקה.



ישנם כפתורים לניווט כמו בדפים הקודמים, וכפתור להוספה אשר מוסיף תזכורת חדשה אשר היא זו שרשומה בתיבת הטקסט. למחיקת תזכורת יש ללחוץ עליה, ותזכורות בצבע סגול מקורן בחברת המשתמש.

לכפתורי קדימה/אחורה ולרשימה עצמה יש את אותן מטרות ואותו מנגנון כמו בדף הרשימות, לכן אין צורך להסביר עליו. ההבדל היחיד הוא שבמקום תבנית Label כאן מופיעות תבניות CheckBox, ובלחיצה עליהם במקום פתיחת דף שינוי הם נמחקות מן מסד הנתונים ישירות (בפעולה CheckBox_Checked שקוראת ל-ReminderService.DropReminder) ודף נטען מחדש באותו העמוד. כמו לתבניות הרשימות, במהלך יצירתן הם יכולות לקבל צבע שונה במידה ובנתונים יש מאפיין חברה חוקי. כפתור ההוספה קורא לפונקציית ההוספה אשר מוסיפה תזכורת חדשה עם הטקסט בתיבה למטה למסד הנתונים (בפעולת AddRCommand שקוראת ל-AddReminder שקוראת ל-ReminderService.ListReminder) (למשתמש, אשר מתקבל בבנאי ביחד עם המסגרת) וטוענת מחדש את הדף באותו העמוד.

חלק הגדרות ופרטים

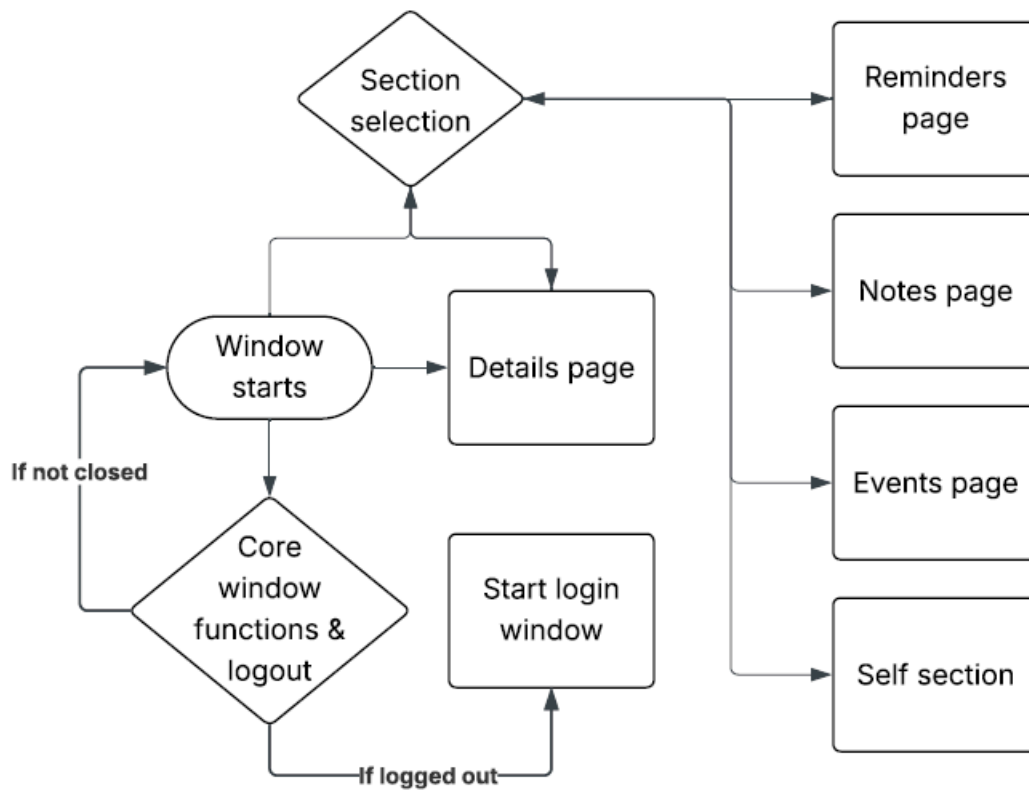
ברגע זה דף ההגדרות ריק, לכן אין צורך להסביר עליו. אם אפתח עוד את הפרויקט הוסיף הגדרות לרקע, גודל מסך ועוד. חלק הפרטים האישיים מראה את פרטיו של המשתמש, ובו קיים כפתור יציאה (בשונה מכפתור הסגירה כפתור זה רק יוצא מן המערכת לדף ההתחלה שבחלון הכניסה).



חלק זה הוא חלק מן חלון המשתמש הפרטי, ולו טקסטים Label המציגים את נתוני המשתמש שהתקבלו בטעינת החלון וכפתור היציאה אשר טוען חלון כניסה חדש וסוגר את החלון הנוכחי.

חלון משתמש חברה

במידה ונבחרה כניסה כמשתמש חברה, נפתח חלון משתמש החברה (במידה והפרטים מתאימים). חלון זה מעוצב עוד פחות מחלון המשתמש הפרטי, אך לו יש יותר אפשרויות: בנוסף לשינוי הפרטים האישיים, ניתן גם להוסיף פרטים למשתמשים קשורים. התרשים הבא מתאר את תהליכי החלון.



בחלון זה חלק עליון בו נמצאים כפתורי הניווט וכפתורי פעולות הבסיס (גם כאן אין שימוש ב- WindowStyle), וחלק תחתון בו קיימת מסגרת למעבר הדפים. בנאי החלון מקבל User והוא המשתמש, ובחלון עצמו יש רק את פונקציות הניווט ופעולות הבסיס.

דף פרטים

בדף הפרטים מופיעים פרטי המשתמש וכן רשימה של המשתמשים הקשורים אליו. בחלק התחתון של הדף ישנו מקום להוספת משתמשים לחברה, אך פעולה זו לא יצאה לפועל (כרגע ניתן להצטרף לחברה רק ביצירת המשתמש).

Sudo (B)
Current: Details

Details Calendar Notes Reminders Self Log Out Quit

Company users:
1, Daniel S; 35, Asphodel Eteroth; 37, UserQ1 Q1WE;

Company:
Name: Sudo
Corp ID: B
Username: Username
Password: Password
Type: 3
User ID: 2

User:
Username: Username
Password: Password
Type: 3
User ID: 2

This feature isn't working yet

Remove Save

כך נראה חלון משתמש החברה כאשר פתוח דף הפרטים. כפי שניתן לראות, בחלק העליון יש את כפתורי הניווט והיצאה, בצד ימין מופיעים פרטי המשתמש ובצד שמאל מופיעה רשימת המשתמשים הקשורים.

בנאי דף זה מקבל User, ממנו הוא משלים את פרטי משתמש החברה באמצעות UserService.Corporative ואת הפרטים הוא מציג בחלק הימני אשר הוא אזור עם Label. בחלק השמאלי (שהוא גם אזור Border עם Label) הבנאי מציג את רשימת המשתמשים הקשורים באמצעות UserService.LaCampanella. דף זה הוא דף צפייה בלבד, כיוון שהחלק התחתון אינו פעיל (עלול להשתנות בעתיד).

דפי הוספה

ישנם שלושה דפי הוספת פריטים, אחד לכל סוג פריט. בכל דף מופיעה תיבה למאפיין משתמש נבחר (להוספה הוא חייב להיות קשור לחברה), תיבות לרשימת פריטי הפריט, וכפתור הוספה אשר מוסיף את הפריט למשתמש הנבחר. פריטים אלו נצבעים בצבע בולט בחלון בדפי הפריטים של המשתמש, כפי שכבר נראה. מנסיבות יעילות הדפים עוצבו באותה הדרך, אך לפי התכנון המקורי (העיצוב הסופי) כל דף היה אמור להיות מותאם לסוג פריטו בדרך אינטואיטיבית יותר וכזו המבדילה בין הדפים למניעת בלבול.

זהו דף להוספת אירועי לוח שנה.

בנאי כל דף מקבל User ומשלים ל- Corp (משתמש החברה, ניתן היה לקצר על ידי השמת משתמש חברה בבנאי אך אין זה משנה), ובכל אחד מן דפי היצירה ישנה פונקציה אחת השורה לכפתור היצירה. פונקציה זו היא SendEvent הרצה על ידי Button_Click הקשורה לכפתור, קודם כל בודקת את תקינות הקלט. לאחר שנבדק שמאפיין המשתמש תקין ושדף יצירת האירועים התאריך תקין, הפונקציה קוראת לפונקצית ההוספה המתאימה ב- ViewModel עם הפרטים הנמצאים בתיבות הטקסט. כל פריט אשר יוסף בדרך זו יהיה בעל מאפיין חברה בהתאם לחברה שיצרה אותו, ומאפיין זה יביא לצבעו השונה של הפריט בדפי המשתמש הפרטי.

Sudo (B)
Current: Notes

DetailsCalendarNotesRemindersSelf

Log OutQuit

Note Name

User ID (internal only)

Note Text

Send

זוהו הדף להוספת רשימות.

Sudo (B)
Current: Reminders

DetailsCalendarNotesRemindersSelf

Log OutQuit

Reminder

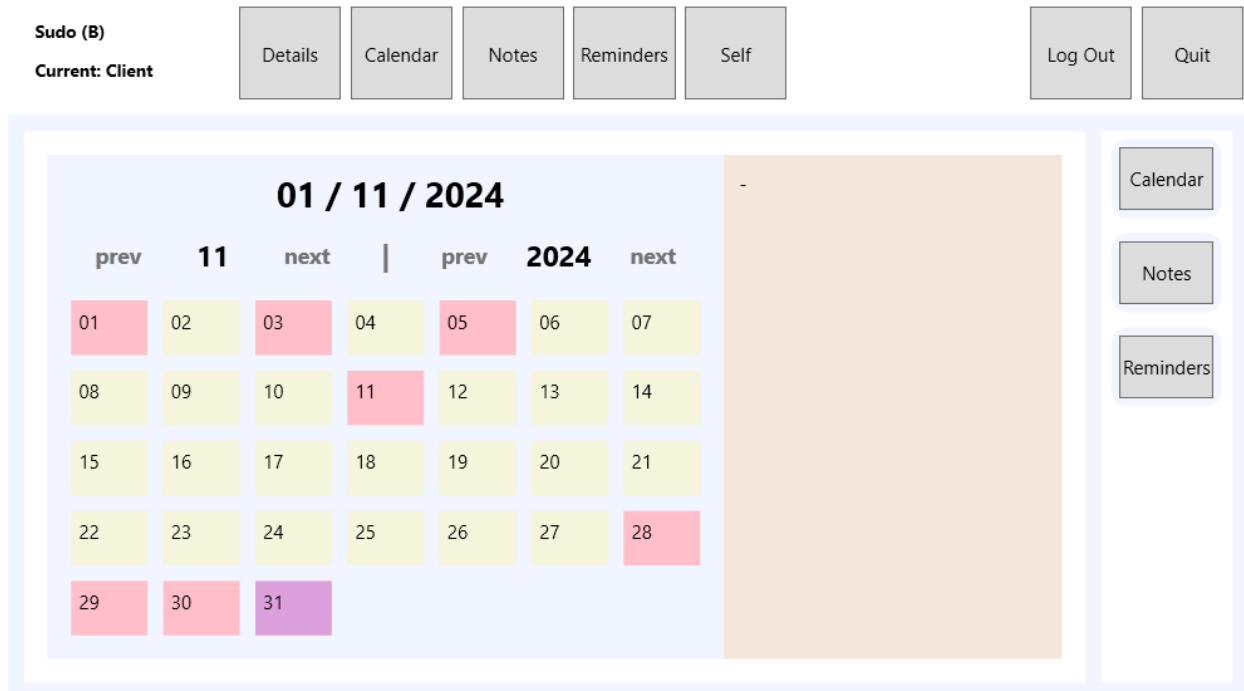
User ID (internal only)

Send

זוהו הדף להוספת תזכורות.

החלק האישי

החלק האישי הוא העתק (יותר נכון לומר שימוש מחדש) של כל שלושת דפי המשתמש הפרטי. גם כאן, דפים אלו היו אמורים להיות נבדלים מדפי המשתמש הפרטי להתאמה לעיצוב החלון, אך עיצובים ועיצוב החלון לא הושלם (לכן נלקחו הדפים המקוריים, שכן לפעולות הפריטים נדרש רק משתמש בסיס).



כאן ניתן לראות את החלק האישי, המהווה מסגרת לדפי הפריטים, ואשר מציג כעת את דף לוח השנה.

בדף זה שני חלקים, אחד הוא המסגרת לדפי הפריטים והאחר הוא תפריט הניווט. שני החלקים כמו בדפים האחרים בנויים על ידי אלמנטים של Border ו-Grid. חלק הניווט הוא StackPanel בו נמצאים כפתורי הניווט.

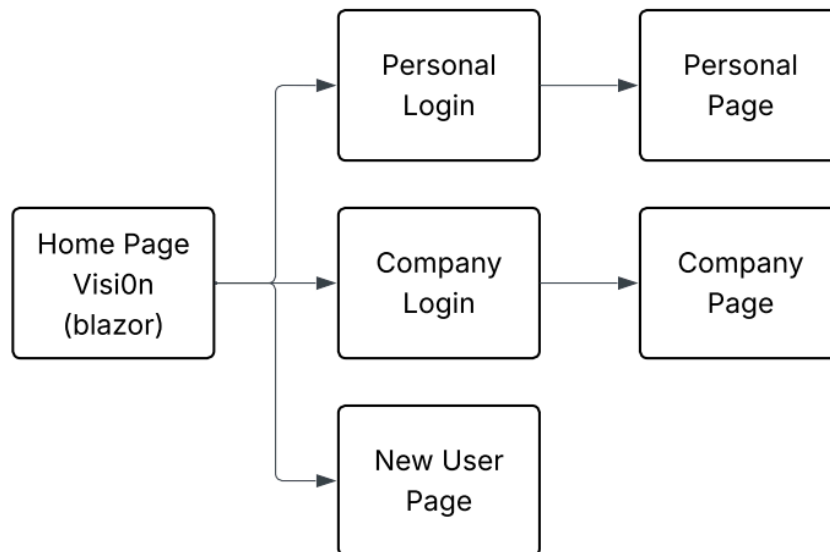
כפתורי הניווט הם אלמנטים של UserControl.

אלמנט UserControl זה, שהוא Menu001 הנמצא בתיקייה Core/UControls מהווה כפתור פשוט מאוד המיוצר במיוחד עבור דף זה ומאפשר ניווט בין דפי המשתמש הפרטי ביחס למסגרת כלשהי. האלמנט הוא אלמנט ניסיוני ועבדתי עליו לא הרבה לפני שהפריקט סוויים ברמה הבסיסית, אך במידה ואעבוד עוד על הפריקט אשלב אלמנטים של UserControl גם במקומות נוספים בהם יש ערך לבחירה זו. כמו כן, השימוש ב- UserControl תואם להרחבה של השימוש ב- UserControl.

ממשק משתמש שני - blazor web app

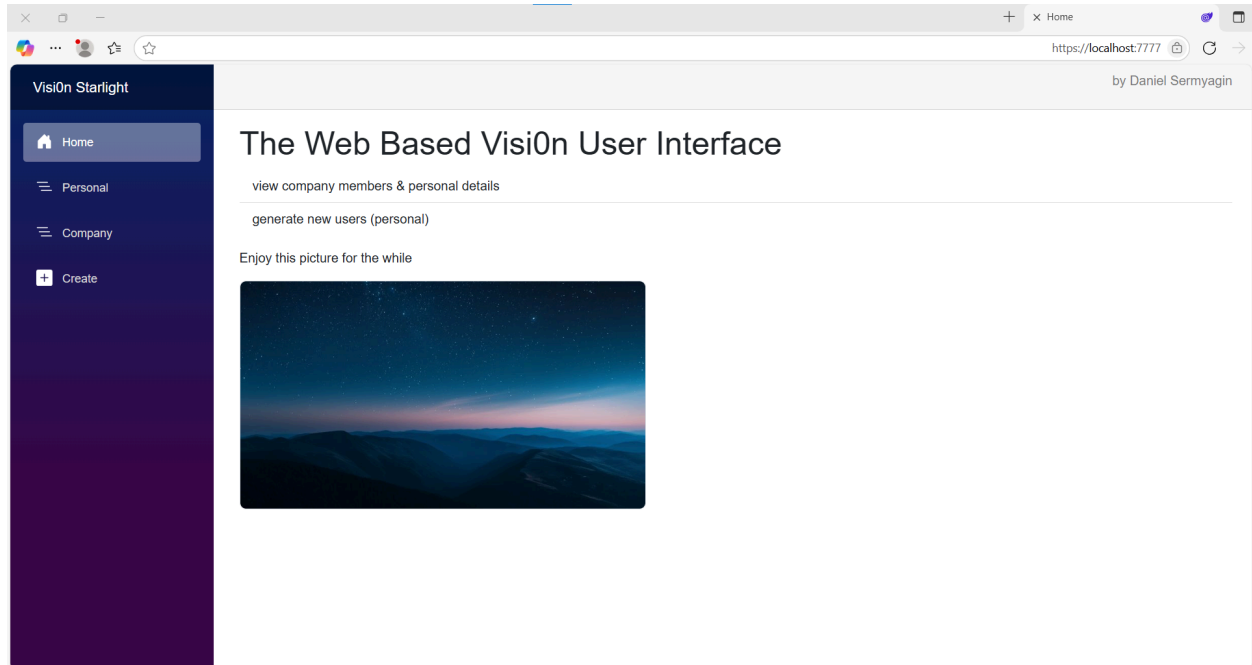
ממשק משתמש זה (שקיבל את השם VisiOn Starlight) הוא אתר הרץ ברשת ועוצב בהתאמה לה בעיצוב אינטרנטי. האתר נפתח בדף כאשר משמאלו יש תפריט לניווט בין הדפים. בממשק זה יש אפשרות להוספת משתמשים פרטיים וצפייה בנתוני המשתמש, נתוני פרטים במידה וזה משתמש פרטי ורשימת משתמשים קשורים במידה וזה משתמש חברה. תוכן הרבה מעבר לאפשרויות אלה, אך מנסיבות כעת אלה הם הפעולות האפשריות. עיצוב הממשק כמו עיצוב האפליקציה אינו הושלם אלה רק ברמה בסיסית, וזאת גם בשימוש עיצוב כללי הניתן על ידי Microsoft.

עוד ניתן לציין שהאתר נבנה בטכנולוגיית razor אשר מאפשרת שילוב של html וקוד c# באותו הקובץ. התרשים הבא מתאר את התהליכים באתר, אותו ניתן לראות גם בחלק ניתוח המערכת.



דף בית

דף הבית הוא דף הנותן מידע על האתר והאפשרויות שלו. אין פעולות שניתן לעשות בדף זה, וממנו יש לעבור בעזרת התפריט לדפים האחרים על ידי לחיצה על הכפתור (השם) הרצוי.

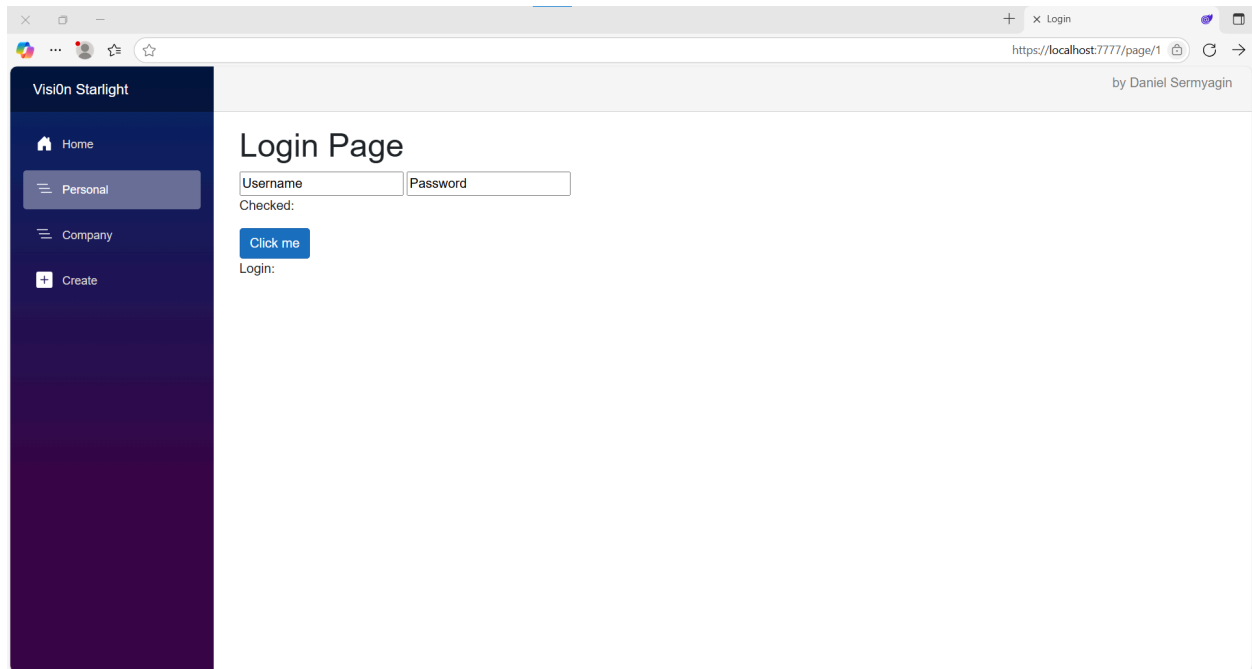


זהו דף הבית כאשר האתר פתוח בדפדפן Edge, מצד שמאל ניתן לראות את התפריט.

מקור התפריט והמסגרת הכוללת של האתר בקבצים `MainLayout.razor` ו-`NavMenu.razor`. הדף שנפתח בתוך המסגרת הוא `Home.razor`, ובו יש רשימה (מאופיין ב-`ul`) בה רשומים ההצעות, טקסט (מאופיין ב-`p`) לגבי התמונה, התמונה עצמה (`img`), הפניות עוגלו על ידי `css` והכותרת של הדף (מסוג `h1`). דף זה הוא לצפייה בלבד.

דף כניסה

דף הכניסה הוא כללי גם למשתמש פרטי וגם למשתמש חברה, אך סוג הכניסה נקבע על ידי ההגעה לדף זה על ידי התפריט. בדף הכניסה ישנם שני תיבות טקסט, לשם משתמש וסיסמה, וכפתור כניסה. ישנו גם טקסט המתאר את מצב הכניסה.

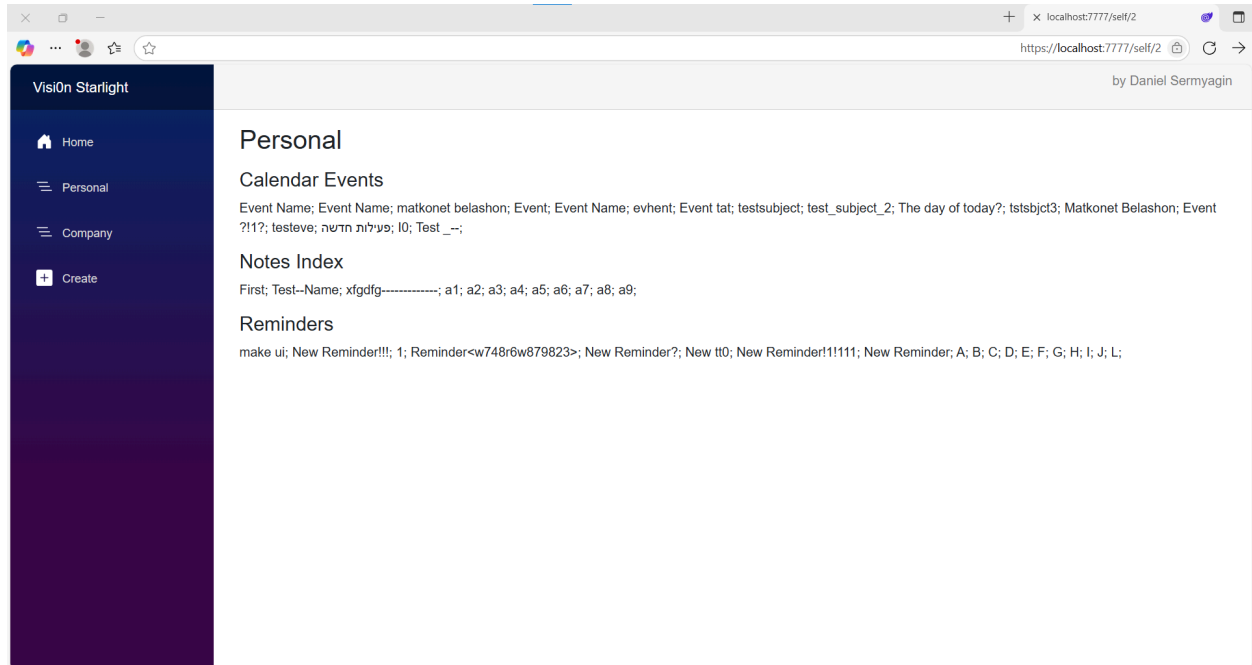


כך נראה דף הכניסה. ניתן לראות את סוג הכניסה בקישור הדף, "page/1".

דף זה, LoginPage.razor, מקיים פרמטר הקשור לכתובת שלו, מחרוזות שם משתמש וסיסמה הקשורות לתיבות הטקסט, כותרת, תיבות טקסט (input), טקסטים להצגת המצב וכפתור לכניסה (button) המקושר לפונקציית הכניסה המבצעת את הכניסה בעזרת פונקציית Call המשתמשת בפונקציית LoginAsync מן שירות הפרויקט לביצוע כניסת המשתמש (ועוד מספר פרטים שאין צורך לפרט עליהם).
אופן פעולת פונקציית הכניסה היא יצירת עצם שירות, קריאה בעזרתו לפונקציית הכניסה, המתנה לתוצאה (הפעולה היא אסינכרונית), ובמידה והכניסה תקינה מעבר לדף המשתמש בהתאם לסוגו (הנקבע לפי הפרמטר שצויין קודם לכן, אשר גם קובע את סוג הכניסה).

דף משתמש פרטי

לאחר כניסה כמשתמש פרטי נפתח דף נתוני המשתמש, בו יהיו שמות כל הפריטים של המשתמש. אין פעולות שניתן לעשות בדף זה כעת.



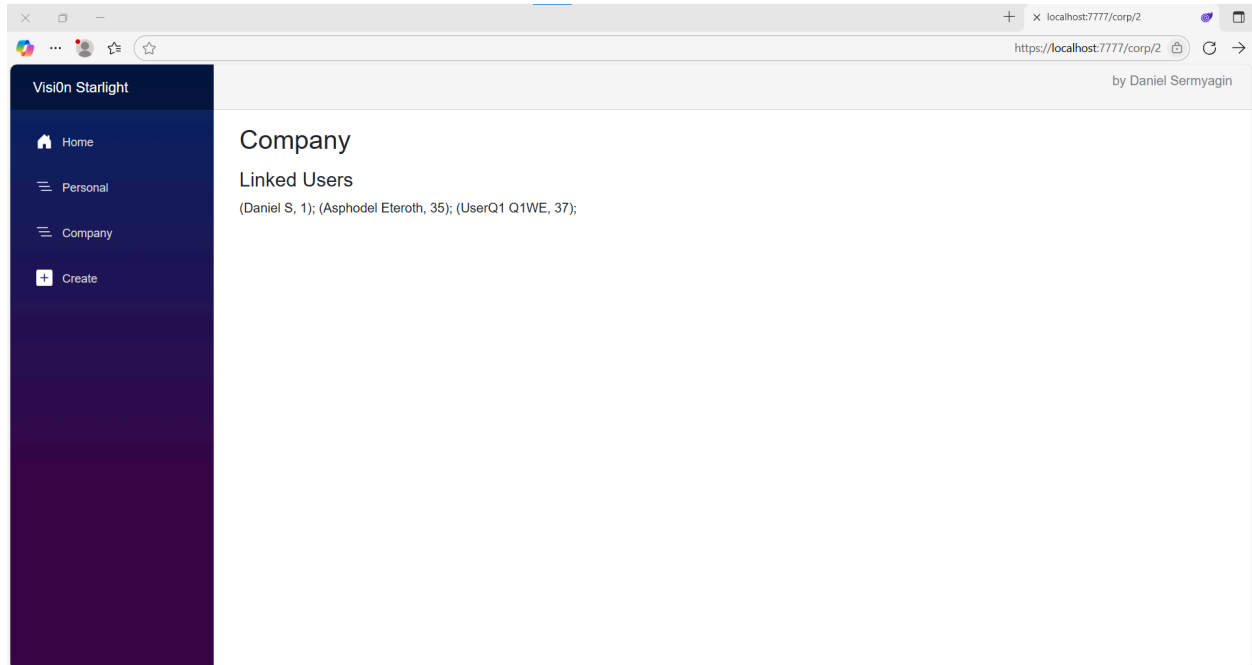
כך נראה דף נתוני המשתמש הפרטי, וניתן לראות את שלוש הרשימות המתארות את הפריטים.

בדף זה ישנם כותרת ראשית, שלוש כותרות משניות לשלושת סוגי הפריטים, והרשימות עצמם שהם מסוג טקסט (p, ליתר דיוק).

כאשר הדף נטען רצה פעולת ההתחלה שקוראת לנתוני המשתמש (בפעולות LoadReminders לתזכורות, בפעולות LoadNotes לרשימות ובפעולות LoadEvents לפעילויות) על פי פרמטר הדף בכתובת המהווה את מאפיין המשתמש, ומיישמת אותם במחרוזות הקשורות לטקסי הרשימות. כפי שצויין, הדף הזה הוא לצפייה בלבד.

דף משתמש חברה

לאחר כניסה כמשתמש חברה, נפתח דף נתוני משתמש החברה ובו רשימת המשתמשים הקשורים. כמו דף נתוני המשתמש הפרטי, זהו דף לצפייה בלבד.



כך נראה דף נתוני משתמש החברה.

מנגנון הפעולה של דף זה זהה לדף המשתמש הפרטי והוא תוכנת בדיוק באותה הדרך, אך במקום לקרוא לנתוני הפרטים פעולת ההתחלה של דף זה קוראת לנתוני המשתמשים הקשורים (בפעולה LoadUsers) ומיישמת אותם בהתאם.

דף הוספה

דף ההוספה הוא הדף המאפשר הוספת משתמשים פרטיים, והוא גם עושה שימוש בשירות החיצוני של Random User Generator. ישנם ארבע תיבות טקסט לנתוני המשתמש, כפתור ליצירת פרטים אקראיים, וכפתור להוספת המשתמש למערכת.

כך נראה דף ההוספה. כפי שרשום, כל המשתמשים שיווצרו בדרך זו יהיו משתמשים פרטיים ולא יהיו קשורים לאף חברה.

בלחיצה על כפתור Generate רצה פעולת GenUser הקוראת לפעולה Create הקוראת לשירות החיצוני וטוענת את הנתונים בתיבות הטקסט. בלחיצה על כפתור Register רצה הפעולה Act היוצרת משתמש חדש באמצעות הפעולה CreatePersonAsync ונתוני תיבות הטקסט. בדף קיימת גם מחלקה לתיאום נתוני השירות החיצוני RootObject0j ותכונותיה, פעולות עזר ואלמנטים אחרים.

רפלקציה

תהליך העבודה החל עוד בחודשים הראשונים ללימודים. אז עוד לא הייתה מטרה ולא היה ידוע מהי המטרה הסופית, אך כבר התבקש ליצור תוכנית פעולה ומסמך ייזום. זאת הייתה נקודת ההתחלה של הבעיות. אדגיש כי אינני כותב זאת כביקורת, אלא להסבר מצבי בתהליך היצירה. זמן רב חלף לו ללא מעשה מהסיבה היחידה שבמקום ליצור ולתכנת היה צריך לתכנן ולפרט. זהו האתגר המרכזי שהיה בתהליך העבודה. ניתן לקרוא לו תכנון זמן לא נכון, אך יש בו גם מעבר.

במלוא הכנות, בשבילי כתיבת מסמכים זהו תהליך מלא בסבל שמוציא את כל החשק מתהליך העבודה. חודשים עברו כמעט מבלי שנגעתי בפרויקט משום שהוא היה בשלב התכנון. אף עכשיו בכתיבת ספר הפרויקט, בשלב זה הפרויקט מוכן כבר כשבועיים אך אינני הייתי יכול להביא את עצמי לכתיבה ופירוט מעשי, שהם גם לא ברמה גבוהה במיוחד. מעבר לכך, הרי שהיה אפשר לצמצם בהרבה את הספר עם תכנון אחר, אחד שלא היה מצריך פירוט כה רב, ונדמה כי עבודה זו חסרת ערך בהשוואה לפרויקט עצמו. בסופו של יום, זוהי חולשתי אשר הראתי וטעות אשר עשיתי, אך הצגתי אותה לא לשווא.

תהליך העבודה התחלק בין מבחנים ועבודות ועניינים אחרים לכמה תקופות בהם עבדתי בניגוד לשאר הזמן. פיתחתי את הפרויקט מנקודת ההתחלה עד כה בזמן עבודה כולל של כשישה שבועות, כאשר לפני שבועיים הייתה הפעם האחרונה בה עבדתי ברצינות ובה הוספתי את שכבת השירות והאתר. לולא הייתי פועל כך הייתי יכול להביא להצלחה רבה יותר, וכמובן שהייתי מוסיף את כל האפשרויות עליהם פירטתי בחלקים השונים בספר. אך הגורל כבר לקח את שלו, וזה מה שנותר. תהליך הלמידה לעומת זאת היה מוצלח. הטכנולוגיה אינה קשה להבנה, וכיוון העבודה היה פשוט. לא הייתה אף פעם אחת בה לא הבנתי את החומר המועבר, בין אם היה על בסיס הנתונים או על שכבת השירות, אולם מנסיבות הוא לא עזר לפתרון בעיות.

ובמידה ונסתכל עליהם, ניתן למצוא מחסור בידע בתחום הרשתות ומבנה מערכות. הפרויקט הציג מערכת הפועלת בעזרת קופסה שחורה, אך נכשל להסביר את הפרטים הגורמים לו לפעול. זה היה צפוי כמובן, בית ספר תיכון אינו המקום המתאים לכך. אך זה הביא למצב בו הרגשתי שאני מגשש ביער אפל... לבסוף למדתי רק חלקים, ואני יודע רק כיצד נראית תוכנה פועלת מבלי לדעת את עקרונותיה העמוקים יותר. מצד שני, עדיין הצלחתי להביא לשליטה מלאה את יכולות התכנות שלי במסגרת ה-wpf, וצברתי ידע רב גם ב-sql, שכבות, תכנות מונחה עצמים ועוד. נקודתי היא שלמדתי הרבה במהלך העבודה בתחומי, אך אני מרגיש שהיה טוב יותר אם הייתי לומד גם את שאר הדברים לעומק. זאת היא עוד אפשרות מפוספסת מבחינתי. אני חושב שלהמשך אקח איתי את טכנולוגיית razor המאפשרת יצירת אתרים בדרך מאוד נוחה, ואת ה-wpf לזמן מסוים. זה כמובן מבלי להתייחס לידע את צברתי.

נדמה בכל אופן כי הטכנולוגיה בה השתמשתי מיושנת ובעלת פוטנציאל לא גבוה במיוחד. אני מקווה שאני טועה, אך בכל אופן היא אכן קשורה ל-Microsoft ול-Windows, וזוהי בעיה מבחינתי, כיוון שאני מעדיף לעבוד עם כלים כלליים יותר.

מבחינה חברתית, תהליך העבודה היה יחסית נעים. ברוב הזמן שהוקצב לא עשיתי הרבה, אך מתי שכן עבדתי מצאתי את חבריי עוסקים באותם הדברים ונעזרנו אחד בשני. מהם גם למדתי על blazor שאיתו יצרתי את האתר. התובנות אשר אני יכול לקחת מכך הם שלעבודה בקהילה יש ערך רב, בהינתן וזוהי הקהילה המתאימה. ואם נחזור לפרויקט עצמו, בראייה לאחור אני יכול להגיד דבר אחד: נכשלתי ביצירתו. פיתחתי פחות מחצי ממה שרציתי ליצור, בזבזתי זמן יקר על התבטלות. הייתי יכול ליצור מערכת התואמת את החזון, אך מה שיש היום זה רק שלד חלוד. הייתי צריך לעבוד יותר, לבזבז פחות זמן על עניינים אחרים שבסוף התגלו כחסרי ערך. הייתי יכול להוסיף הגדרות, פרופיל משתמש, קישור בין הדפים. הייתי יכול להפוך את האתר לפעיל בהרבה, והייתי יכול לעצב את הפרויקט ברמה גבוהה. הייתי יכול ליעל את הפעולות בהרבה ולשלב גישות תכנות שונות. ישנם אינספור דברים אשר הייתי יכול לעשות עם פרויקט זה, אולי אף הייתי יכול להפוך אותו למערכת ברמת השוק. יש לי את הכלים והידע לכך, אך לא עשיתי זאת.

זמן. זמן הוא המשאב שהיה לי כה הרבה, וכה קצת ממנו. ומעבר לאתגרי הפיתוח שהסתדרתי איתם בצורה מספקת, בין אם היה זה להבין למה הספק למסד הנתונים אינו קיים או מהי שגיאה פנימית בחיבור שירות הרשת לאתר, הזמן היה האתגר הגדול מכולם, והוא קיבל את כוחו מגישתי הרעועה.

לפני עומדת יצירה כה לא מוכנה שאני אף לא יכול לענות על מה הייתי יכול לשפר כתוצאה ממגוון רב כל כך של חלקים לשיפור. באותה מידה, אינני יכולה לומר אם הייתי מיישם את חלקי הפרויקט בדרך שונה. אמנם ככל הנראה לא מלבד שינוי מסד הנתונים מ- Access לאלטרנטיבה יותר כללית, אך שוב, לדעתי לא עשיתי מספיק כדי באמת לענות על שאלה זו.

אבל בסופו של דבר הפרויקט קיים. הוא עונה על הדרישות, והוא פועל. הצלחתי בעזרת מגוון מקורות להתגבר על קשיי הידע, בעזרת Github ו- Stack Overflow למצוא פתרונות לשגיאות, ובעזרת יצירתיות להחליף אלטרנטיביות לבעיות בלתי פתירות כגון גרסאות לא מתאימות.

הפרויקט מוכן, אמנם ברמה מאוד בסיסית, אמנם הוא רק הצל של מה שהיה יכול להיות, אך עבודתי הסתיימה. תודה לכם, שימי יניב וגיל, על כך שהייתם איתי בדרך זו. הפרויקט אכן היה חוויה משמעותית.

נספחים

מצורף הקוד המלא של הפרויקט נכון לעכשיו

Corp.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;

namespace Model_
{
    /// <summary>
    /// A control unit user - all (corp) are (user)
    /// </summary>
    public class Corp : User
    {
        // function as cosmetic in relation to users; alternative control over users (self & other
        // compared to self)

        public string _cid;
        public string _cName;

        public Corp() : base()
        {
            _cid = "_"; _cName = "";
        }
        public Corp(string cid, string cName, string un, string pd, int id) : base(un, pd, id, 2)
        {
            _cid = cid;
            _cName = cName;
        }
        public Corp(User u, string id, string cn) : base(u._usrName, u._pwd, u._absId, 1)
        {
            _cid = id;
            _cName = cn;
        }
    }
}
```

```
}
```

Entity.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Model_
{
    public class Entity
    {
        // for convert control
    }
}
```

Event.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Model_
{
    public class Event : Entity
    {
        public int _uid;
        public string _name;
        public string _description;
        public string _date;
        public int _ID;
        public string _cid;

        public Event(int u, string n, string d, string t, int i, string cc = "--")
        {
            _uid = u;
            _name = n;
            _description = d;
            _date = t;
        }
    }
}
```

```

        _ID = i;
        _cid = cc;
    }

    public Event()
    {
        _uid = -1;
        _name = string.Empty;
        _description = string.Empty;
        _date = string.Empty;
        _ID = -1;
        _cid = "--";
    }

    public Event Copy() => new Event(_uid, _name, _description, _date, _ID, _cid);
}
}

```

Noteltem.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Model_
{
    public class Noteltem : Entity
    {
        public int _uid;
        public string _name;
        public string _text;
        public string _cid;

        public Noteltem()
        {
            _name = "";
            _text = "";
            _uid = -1;
            _cid = "--";
        }

        public Noteltem(string n, string t, int u, string c = "--")
    }
}

```

```

    {
        _name = n;
        _text = t;
        _uid = u;
        _cid = c;
    }
}
}

```

Person.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;

namespace Model_
{
    /// <summary>
    /// A continuum of user for personal matters - all (person) are (user)
    /// </summary>
    public class Person : User
    {
        public string _fName;
        public string _lName;
        public string _cid;

        public Person() : base() { _fName = ""; _lName = ""; _cid = "-"; }

        public Person(string cid, string fName, string lName, string un, string pd, int id) : base(un,
pd, id, 1)
        {
            _cid = cid;
            _fName = fName;
            _lName = lName;
        }

        public Person(User u, string f, string l, string c) : base(u._usrName, u._pwd, u._absId, 1)
        {
            _cid = c;
            _fName = f;

```

```

        _IName = I;
    }

    public override void Copy(User u)
    {
        base.Copy(u);
        Person p; if (u is Person) p = u as Person; else p = new Person();
        _cid = p._cid;
        _fName = p._fName;
        _IName = p._IName;
    }
}
}

```

Reminder.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Model_
{
    public class Reminder : Entity
    {
        public int _uid;
        public string _text;
        public string _cid;

        public Reminder()
        {
            _uid = -1;
            _text = "";
            _cid = "--";
        }

        public Reminder(int u, string t, string c = "--")
        {
            _uid = u;
            _text = t;
            _cid = c;
        }
    }
}

```

```
}
```

User.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Model_
{
    /// <summary>
    /// The target of contact, the actual user, in its basic state
    /// </summary>
    public class User : Entity
    {
        public int _absId;
        public string _usrName { get; set; }
        public string _pwd { get; set; }
        public int _type;

        public User()
        {
            _usrName = "";
            _pwd = "";
            _absId = -1;
            _type = 0;
        }

        public User(string un, string pd, int id, int t)
        {
            _usrName = un;
            _pwd = pd;
            _absId = id;
            _type = t;
        }

        public virtual void Copy(User u)
        {
            _usrName = u._usrName;
            _pwd = u._pwd;
            _absId = u._absId;
            _type = u._type;
        }
    }
}
```



```
}
}
}
```

Dict0.xaml

```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
```

```
<Style TargetType="Button" x:Key="Button01">
    <Style.Setters>
        <Setter Property="BorderThickness" Value="5" />
        <Setter Property="Background" Value="Transparent"/>
        <Setter Property="Foreground" Value="AliceBlue" />
        <Setter Property="FontWeight" Value="Bold"/>
        <Setter Property="FontSize" Value="18"/>
        <Setter Property="BorderBrush">
            <Setter.Value>
                <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
                    <GradientStop Color="#FF00D6FF" Offset="0" />
                    <GradientStop Color="#FF1057E4" Offset="0.35" />
                    <GradientStop Color="#FF3700FF" Offset="0.7" />
                    <GradientStop Color="#FF7100FF" Offset="1.0" />
                </LinearGradientBrush>
            </Setter.Value>
        </Setter>
    </Style.Setters>

    <Style.Triggers>
        <!-- Trigger Property="AreAnyTouchesDirectlyOver" Value="True">
            <Setter Property="Background">
                <Setter.Value>
                    <ImageBrush ImageSource="/Pictures/sea.jpg"/>
                </Setter.Value>
            </Setter>
        </Trigger-->
        <Trigger Property="IsMouseOver" Value="True">
            <Setter Property="Foreground" Value="Black"/>
        </Trigger>
    </Style.Triggers>
</Style>

<Style TargetType="Button" x:Key="Button02">
    <Style.Setters>
```

```

<Setter Property="BorderThickness" Value="5" />
<Setter Property="Background" Value="Transparent"/>
<Setter Property="Foreground" Value="AliceBlue" />
<Setter Property="FontWeight" Value="Bold"/>
<Setter Property="FontSize" Value="18"/>
<Setter Property="BorderBrush">
  <Setter.Value>
    <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
      <GradientStop Color="#FFFF0023" Offset="0" />
      <GradientStop Color="#FFE44A10" Offset="0.4" />
      <GradientStop Color="DarkOrange" Offset="0.8" />
      <GradientStop Color="Gold" Offset="1.0" />
    </LinearGradientBrush>
  </Setter.Value>
</Setter>
</Style.Setters>

<Style.Triggers>
  <Trigger Property="IsMouseOver" Value="True">
    <Setter Property="Foreground" Value="Black"/>
  </Trigger>
</Style.Triggers>
</Style>

<Style TargetType="Button" x:Key="Button03">
  <Style.Setters>
    <Setter Property="BorderThickness" Value="5" />
    <Setter Property="Background" Value="Transparent"/>
    <Setter Property="Foreground" Value="AliceBlue" />
    <Setter Property="FontWeight" Value="Bold"/>
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="BorderBrush">
      <Setter.Value>
        <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
          <GradientStop Color="#FF3D2F5B" Offset="0" />
          <GradientStop Color="#FF3D2A5F" Offset="0.4" />
          <GradientStop Color="#FF592A52" Offset="0.8" />
          <GradientStop Color="#FF5C3A57" Offset="1.0" />
        </LinearGradientBrush>
      </Setter.Value>
    </Setter>
  </Style.Setters>

  <Style.Triggers>

```

```

        <Trigger Property="IsMouseOver" Value="True">
            <Setter Property="Foreground" Value="Black"/>
        </Trigger>
    </Style.Triggers>
</Style>

<Style TargetType="TextBox" x:Key="TextBox01">
    <Style.Setters>
        <Setter Property="Foreground" Value="AliceBlue" />
        <Setter Property="FontSize" Value="12"/>
        <Setter Property="BorderBrush" Value="Transparent"/>
        <Setter Property="Background" Value="Transparent">
            <!--Setter.Value>
                <LinearGradientBrush>
                    <GradientStop Color="#FF3D2F5B" Offset="0" />
                    <GradientStop Color="#FF3D2A5F" Offset="0.4" />
                    <GradientStop Color="#FF592A52" Offset="0.8" />
                    <GradientStop Color="#FF5C3A57" Offset="1.0" />
                </LinearGradientBrush>
            </Setter.Value-->
        </Setter>

    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="TextBox">
                <Border CornerRadius="10">
                    <Border.Background>
                        <LinearGradientBrush>
                            <GradientStop Color="#FF343456"/>
                            <GradientStop Color="#FF3B324E" Offset="1"/>
                        </LinearGradientBrush>
                    </Border.Background>
                    <!--Border.Background>
                        <LinearGradientBrush>
                            <GradientStop Color="#FF3D2F5B" Offset="0" />
                            <GradientStop Color="#FF3D2A5F" Offset="0.4" />
                            <GradientStop Color="#FF592A52" Offset="0.8" />
                            <GradientStop Color="#FF5C3A57" Offset="1.0" />
                        </LinearGradientBrush>
                    </Border.Background-->

                    <Label x:Name="TheText" HorizontalAlignment="Left"
VerticalAlignment="Center" Content="{TemplateBinding Text}" Margin="5"
Foreground="AliceBlue"/>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>

```

```

        </Border>
    </ControlTemplate>
</Setter.Value>
</Setter>

</Style.Setters>
</Style>

</ResourceDictionary>

```

Dict2.xaml

```

<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <!--Style x:Key="WindowS1" TargetType="Window">
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate>
                    <Border CornerRadius="10">
                        <Border.Background>
                            <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                                <GradientStop Color="#FF373546"/>
                                <GradientStop Color="#FF272537" Offset="1"/>
                            </LinearGradientBrush>
                        </Border.Background>
                        <Grid>
                            <Grid.RowDefinitions>
                                <RowDefinition Height="30*" />
                                <RowDefinition Height="400*" />
                            </Grid.RowDefinitions>

                            <Grid Margin="12,0">
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="1*" />
                                </Grid.RowDefinitions>

                                <StackPanel x:Name="TopMenu" Orientation="Horizontal"
                                    FlowDirection="RightToLeft">
                                    <Button x:Name="Quit" Content="X" Width="50" Foreground="AliceBlue"
                                        Background="Transparent" BorderBrush="Transparent"/>
                                    <Button x:Name="Mini" Content="___" Width="50"
                                        Foreground="AliceBlue" Background="Transparent" BorderBrush="Transparent"/>
                                </StackPanel>
                            </Grid>
                        </Grid>
                    </Border>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </--Style>

```

```

        <Button x:Name="Maxi" Content="[ ]" Width="50"
Foreground="AliceBlue" Background="Transparent" BorderBrush="Transparent"/>
    </StackPanel>
</Grid>
</Grid>
</Border>
</ControlTemplate>
</Setter.Value>
</Setter>
</Style-->

```

```

<Style x:Key="CDay01" TargetType="Label">
    <Setter Property="Background" Value="Beige"/>
</Style>

```

```

<Style x:Key="Note01" TargetType="Label">
    <Setter Property="Background" Value="Beige"/>
</Style>
<Style x:Key="Note02" TargetType="Label">
    <Setter Property="Background" Value="Plum"/>
</Style>

```

```

<Style x:Key="Reminder01" TargetType="CheckBox">
    <Setter Property="Background" Value="Beige"/>
</Style>
<Style x:Key="Reminder02" TargetType="CheckBox">
    <Setter Property="Background" Value="Plum"/>
</Style>

```

```

<Style x:Key="CDay02" TargetType="RadioButton">
    <Style.Setters>
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="RadioButton">
                    <Border Background="{TemplateBinding Background}">
                        <Label Content="{TemplateBinding Content}"/>
                    </Border>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style.Setters>

```

```

<Style.Triggers>
  <Trigger Property="IsChecked" Value="True">
    <Setter Property="Template">
      <Setter.Value>
        <ControlTemplate TargetType="RadioButton">
          <Border Background="Orange">

            <Label Content="{TemplateBinding Content}"/>
          </Border>
        </ControlTemplate>
      </Setter.Value>
    </Setter>
  </Trigger>
  <!-- Trigger Property="IsMouseDirectlyOver" Value="True">
    <Setter Property="Template">
      <Setter.Value>
        <ControlTemplate TargetType="RadioButton">
          <Border Background="Blue">

            <Label Content="{TemplateBinding Content}"/>
          </Border>
        </ControlTemplate>
      </Setter.Value>
    </Setter>
  </Trigger-->
</Style.Triggers>
</Style>

```

```

<Style x:Key="CaleControl01" TargetType="Button">
  <Style.Setters>
    <Setter Property="Background" Value="Transparent"/>
    <Setter Property="BorderThickness" Value="0"/>
    <Setter Property="BorderBrush" Value="Transparent"/>
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="Foreground" Value="#FF757575"/>
    <Setter Property="FontWeight" Value="Bold"/>
    <Setter Property="VerticalAlignment" Value="Center"/>
    <Setter Property="HorizontalAlignment" Value="Center"/>
  </Style.Setters>
</Style>

```

```

</ResourceDictionary>

```

DictP.xaml

```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
```

```
<Style TargetType="Button" x:Key="BP01">
    <Style.Setters>
        <Setter Property="BorderThickness" Value="5" />
        <Setter Property="Background" Value="Transparent"/>
        <Setter Property="Foreground">
            <Setter.Value>
                <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
                    <GradientStop Color="#FF60617B" Offset="0"/>
                    <GradientStop Color="#FF534F58" Offset="0.6"/>
                </LinearGradientBrush>
            </Setter.Value>
        </Setter>
        <Setter Property="FontWeight" Value="Bold"/>
        <Setter Property="FontSize" Value="14"/>
        <Setter Property="BorderBrush">
            <Setter.Value>
                <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
                    <GradientStop Color="#FFCBD5FF" Offset="0"/>
                    <GradientStop Color="#FFC7E9FF" Offset="0.8"/>
                </LinearGradientBrush>
            </Setter.Value>
        </Setter>
    </Style.Setters>
</Style>
```

```
<Style TargetType="Label" x:Key="BP2L01">
    <Setter Property="FontSize" Value="16"/>
    <Setter Property="FontWeight" Value="Bold"/>
    <Setter Property="Foreground">
        <Setter.Value>
            <!--SolidColorBrush Color="#FF353538"/-->
            <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
                <GradientStop Color="#FF60617B" Offset="0"/>
                <GradientStop Color="#FF534F58" Offset="0.6"/>
            </LinearGradientBrush>
        </Setter.Value>
    </Setter>
</Style>
```

```

        </Setter.Value>
    </Setter>
    <Setter Property="HorizontalAlignment" Value="Center"/>
    <Setter Property="VerticalAlignment" Value="Center"/>
</Style>

<Style x:Key="BP02" TargetType="RadioButton">
    <Style.Setters>
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="RadioButton">
                    <Border>
                        <Border.Background>
                            <!--LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
                                <GradientStop Color="#FFF3FDFF" Offset="0"/>
                                <GradientStop Color="#FFF0F3FF" Offset="0.6"/>
                                <GradientStop Color="#FFFFFF5FE" Offset="1"/>
                            </LinearGradientBrush-->
                            <SolidColorBrush Color="Transparent"/>
                        </Border.Background>

                        <Label Content="{TemplateBinding Content}" Style="{StaticResource
BP2L01}"/>
                    </Border>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style.Setters>
    <Style.Triggers>
        <!--Trigger Property="IsChecked" Value="True">
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="RadioButton">
                        <Border Background="Orange">

                            <Label Content="{TemplateBinding Content}"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Trigger-->
        <Trigger Property="IsMouseOver" Value="True">
            <Setter Property="Template">
                <Setter.Value>

```



```

        <ControlTemplate TargetType="RadioButton">
            <Border>
                <Border.Background>
                    <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1"
Opacity="0.125">
                        <!--GradientStop Color="#FFC7E9FF" Offset="0"/>
                        <GradientStop Color="#FFF4D6FF" Offset="1"/-->
                        <!--GradientStop Color="#FFD7EAED" Offset="0"/>
                        <GradientStop Color="#FFC8CDE6" Offset="0.5"/>
                        <GradientStop Color="#FFE2CAE0" Offset="1"/-->
                        <GradientStop Color="#FF546CD0" Offset="0"/>
                        <GradientStop Color="#FFD26DCE" Offset="1"/>
                    </LinearGradientBrush>
                </Border.Background>

                <Label Content="{TemplateBinding Content}" Style="{StaticResource
BP2L01}"/>
            </Border>
        </ControlTemplate>
    </Setter.Value>
</Setter>
</Trigger>
<Trigger Property="IsMouseCaptured" Value="True">
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="RadioButton">
                <Border>
                    <Border.Background>
                        <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1"
Opacity="0.975">
                            <GradientStop Color="#FFCBD5FF" Offset="0"/>
                            <GradientStop Color="#FFFFCFFD" Offset="1"/>
                        </LinearGradientBrush>
                    </Border.Background>

                    <Label Content="{TemplateBinding Content}" Style="{StaticResource
BP2L01}"/>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Trigger>
</Style.Triggers>
</Style>

```

```

<Style TargetType="Label" x:Key="pfp">
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="Label">
        <Border Background="Black" CornerRadius="20">
          <Image Source="{TemplateBinding Content}"/>
        </Border>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>

<Style TargetType="Label" x:Key="LL01">
  <Setter Property="FontWeight" Value="Bold"/>
  <Setter Property="FontSize" Value="{Binding FontSize}"/>
  <Setter Property="Foreground">
    <Setter.Value>
      <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
        <GradientStop Color="#FF60617B" Offset="0"/>
        <GradientStop Color="#FF534F58" Offset="0.6"/>
      </LinearGradientBrush>
    </Setter.Value>
  </Setter>
</Style>

<Style TargetType="Label" x:Key="LL02">
  <Setter Property="FontWeight" Value="Bold"/>
  <Setter Property="FontSize" Value="{Binding FontSize}"/>
  <Setter Property="Foreground">
    <Setter.Value>
      <LinearGradientBrush StartPoint="1,0" EndPoint="0.5,1">
        <GradientStop Color="#FF87899E" Offset="0"/>
        <GradientStop Color="#FF7F8086" Offset="1"/>
      </LinearGradientBrush>
    </Setter.Value>
  </Setter>
</Style>

<Style TargetType="Label" x:Key="LR01">
  <Setter Property="FontWeight" Value="Bold"/>
  <Setter Property="FontSize" Value="11.5"/>
  <Setter Property="Foreground" Value="#FFDBE4E4"/>
</Style>
<Style TargetType="Label" x:Key="LR02">

```

```

<Setter Property="FontWeight" Value="Bold"/>
<Setter Property="FontSize" Value="11.5"/>
<Setter Property="Foreground" Value="#FF3C3C3C"/>
</Style>

<Style TargetType="Border" x:Key="PHcolor1">
  <Setter Property="Background">
    <Setter.Value>
      <LinearGradientBrush StartPoint="1,0" EndPoint="0.5,1">
        <GradientStop Color="#FF3C4AD4" Offset="0.303"/>
        <GradientStop Color="#FF102288" Offset="1"/>
      </LinearGradientBrush>
    </Setter.Value>
  </Setter>
  <Setter Property="CornerRadius" Value="5"/>
  <Setter Property="Width" Value="100"/>
  <Setter Property="Height" Value="100"/>
  <Setter Property="Margin" Value="10, 0"/>
</Style>

<Style TargetType="Border" x:Key="PHcolor2">
  <Setter Property="Background">
    <Setter.Value>
      <LinearGradientBrush>
        <GradientStop Color="#FFEC0E41"/>
        <GradientStop Color="#FF4A0606" Offset="0.843"/>
      </LinearGradientBrush>
    </Setter.Value>
  </Setter>
  <Setter Property="CornerRadius" Value="5"/>
  <Setter Property="Width" Value="100"/>
  <Setter Property="Height" Value="100"/>
  <Setter Property="Margin" Value="10, 0"/>
</Style>

<Style TargetType="Border" x:Key="PHcolor3">
  <Setter Property="Background">
    <Setter.Value>
      <LinearGradientBrush>
        <GradientStop Color="#FF85EC0E" Offset="0.05"/>
        <GradientStop Color="#FF399C63" Offset="0.397"/>
        <GradientStop Color="#FF0EECC8" Offset="0.777"/>
      </LinearGradientBrush>
    </Setter.Value>
  </Setter>
  <Setter Property="CornerRadius" Value="5"/>

```

```
<Setter Property="Width" Value="100"/>
<Setter Property="Height" Value="100"/>
<Setter Property="Margin" Value="10, 0"/>
</Style>
```

```
</ResourceDictionary>
```

Menu001.xaml

```
<UserControl x:Class="Visi0n._0.Menu001"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0"
    mc:Ignorable="d"
    d:DesignHeight="100" d:DesignWidth="100">
    <Grid Margin="5">
        <Border Background="AliceBlue" CornerRadius="10" Height="50" Width="70">
            <Button Name="ActB" Margin="5" Content="Action" Click="Button_Click"/>
        </Border>
    </Grid>
</UserControl>
```

Menu001.xaml.cs

```
using Model_;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Visi0n._0.Pages.General;
```

```

namespace Visi0n._0
{
    /// <summary>
    /// Interaction logic for Menu001.xaml
    /// </summary>
    public partial class Menu001 : UserControl
    {
        User _user;
        Frame _frame;
        int _type;

        public Menu001(Frame f, User u, int t = 0)
        {
            InitializeComponent();
            this.DataContext = this;
            _user = u;
            _type = t;
            _frame = f;

            switch (_type)
            {
                case 1:
                    ActB.Content = "Calendar"; break;
                case 2:
                    ActB.Content = "Notes"; break;
                default:
                    ActB.Content = "Reminders"; break;
            }
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            switch (_type)
            {
                case 1:
                {
                    _frame.Navigate(new CalendarGP(_frame, _user));
                    break;
                }
                case 2:
                {
                    _frame.Navigate(new NotesGP(_frame, _user));
                    break;
                }
            }
        }
    }
}

```

```

    }
    default:
    {
        _frame.Navigate(new RemindersGP(_frame, _user));
        break;
    }
}
}
}
}
}

```

CompanyDetailsPage.xaml

```

<Page x:Class="Visi0n._0.Pages.Company.CompanyDetailsPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0.Pages.Company"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="CompanyDetailsPage">

    <Grid Margin="10">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="100*"/>
            <ColumnDefinition Width="50*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="6*"/>
            <RowDefinition Height="1*"/>
        </Grid.RowDefinitions>

        <!--ListView Margin="10,10,10,10" Grid.ColumnSpan="1">
            <ListViewItem Content="User 1"/>
            <ListViewItem Content="User 2"/>
        </ListView-->
        <Label x:Name="UsersText" Content="Users" Background="AliceBlue" Margin="10"/>

        <Border Background="Beige" Margin="10" Grid.Column="1">
            <Label Name="DetailsText" Content="Company"/>
        </Border>
    </Grid>

```

```

        <StackPanel Grid.Row="1" Grid.Column="1" Orientation="Horizontal"
        FlowDirection="RightToLeft">
            <Button Content="Save" Margin="10" Width="90"/>
            <Button Content="Remove" Margin="10" Width="90"/>
        </StackPanel>

        <TextBox Grid.Row="1" Margin="10" Text="This feature isn't working yet"/>
    </Grid>
</Page>

```

CompanyDetailsPage.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Model_;
using VModel_;

namespace Visi0n._0.Pages.Company
{
    /// <summary>
    /// Interaction logic for CompanyDetailsPage.xaml
    /// </summary>
    public partial class CompanyDetailsPage : Page
    {
        User _user;
        //Frame _frame;
        Corp _corporation;

        public CompanyDetailsPage(*Frame frame,* / User u)
        {
            InitializeComponent();
            //_frame = frame;

```

```

        _user = u;

        DataSetUp(_user);
    }

    private void DataSetUp(User u)
    {
        _corporation = UserService.Corporative(u);
        DetailsText.Content = "Company: \n\n";

        if (_corporation != null)
        {
            DetailsText.Content += "Name: " + _corporation._cName + "\n";
            DetailsText.Content += "Corp ID: " + _corporation._cid + "\n";
            DetailsText.Content += "Username: " + _corporation._usrName + "\n";
            DetailsText.Content += "Password: " + _corporation._pwd + "\n";
            DetailsText.Content += "Type: " + _corporation._type + "\n";
            DetailsText.Content += "User ID: " + _corporation._absId + "\n";
        }
        DetailsText.Content += "\nUser: \n\n";

        DetailsText.Content += "Username: " + u._usrName + "\n";
        DetailsText.Content += "Password: " + u._pwd + "\n";
        DetailsText.Content += "Type: " + u._type + "\n";
        DetailsText.Content += "User ID: " + u._absId + "\n";

        if (_corporation != null)
        {
            UsersText.Content = "Company users: \n";
            List<Person> ll = UserService.LaCampanella(_corporation);
            foreach (Person p in ll)
            {
                UsersText.Content += (" " + p._absId + ", " + p._fName + " " + p._lName + "; ");
            }
        }
    }
}

```

CompanyEventsPage.xaml

```

<Page x:Class="Visi0n._0.Pages.Company.CompanyEventsPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

```



```

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:local="clr-namespace:Visi0n._0.Pages.Company"
mc:Ignorable="d"
d:DesignHeight="450" d:DesignWidth="800"
Title="CompanyEventsPage">

<Grid>
  <Border Background="AliceBlue" Margin="10">
    <Grid>
      <Grid.RowDefinitions>
        <RowDefinition Height="7*"/>
        <RowDefinition Height="30*"/>
        <RowDefinition Height="7*"/>
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="5*"/>
        <ColumnDefinition Width="5*"/>
      </Grid.ColumnDefinitions>
      <TextBox Text="Event Name" Margin="5" Name="Ename"/>
      <TextBox Text="User ID (internal only)" Margin="5" Grid.Column="1" Name="Uid"/>
      <TextBox Text="Event Details" Margin="5" Grid.Row="1" Name="Etext"/>
      <StackPanel Orientation="Horizontal" FlowDirection="RightToLeft" Grid.Column="1"
Grid.Row="2">
        <Button Margin="5" Width="100" Content="Send" Click="Button_Click"/>
      </StackPanel>
      <Grid Grid.Row="2">
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="5*"/>
          <ColumnDefinition Width="5*"/>
          <ColumnDefinition Width="7*"/>
        </Grid.ColumnDefinitions>
        <TextBox Height="50" Width="50" Grid.Column="0" Name="Tdd" Text="dd"/>
        <TextBox Height="50" Width="50" Grid.Column="1" Name="Tmm" Text="mm"/>
        <TextBox Height="50" Width="70" Grid.Column="2" Name="Tyy" Text="yyy"/>
      </Grid>
    </Grid>
  </Border>
</Grid>
</Page>

```

CompanyEventsPage.xaml.cs

```
using Model_;
```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using VModel_;

namespace Visi0n._0.Pages.Company
{
    /// <summary>
    /// Interaction logic for CompanyEventsPage.xaml
    /// </summary>
    public partial class CompanyEventsPage : Page
    {
        User _user;
        Corp _self;

        public CompanyEventsPage(User u)
        {
            InitializeComponent();

            _user = u;
            _self = UserService.Corporative(_user); //assuming validity
        }

        private void SendEvent()
        {
            bool f = true;
            try
            {
                int id = int.Parse(Uid.Text);
                int ddat = int.Parse(Tdd.Text + Tmm.Text + Tyy.Text);
            }
            catch { f = false; }
        }
    }
}

```

```

        if (f && Tdd.Text.Length == 2 && Tmm.Text.Length == 2 && Tyy.Text.Length == 4 &&
            int.Parse(Tdd.Text) < 32 && int.Parse(Tmm.Text) < 13)
        {
            List<Person> emp = UserService.LaCampanella(_self);
            User u0 = UserService.Get(int.Parse(Uid.Text));
            User uu = null;
            foreach (User u in emp)
            {
                if (u._absId == u0._absId) uu = u0;
            }

            if (uu != null) { MessageBox.Show($"Attempting Event for id: {uu._absId}");
                EventService.WriteEclipse(uu, null, Ename.Text, Etext.Text, "" + Tdd.Text + "/" + Tmm.Text + "/"
                + Tyy.Text, _self._cid); }
            else MessageBox.Show("Make sure Uid is registered in the company");
        }
        else MessageBox.Show("Make sure input matches target types");
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        SendEvent();
    }
}

```

CompanyNotesPage.xaml

```

<Page x:Class="Visi0n._0.Pages.Company.CompanyNotesPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0.Pages.Company"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="CompanyNotesPage">

    <Grid>
        <Border Margin="10" Background="AliceBlue">
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="7*" />
                    <RowDefinition Height="30*" />
                </Grid.RowDefinitions>
            </Grid>
        </Border>
    </Grid>

```

```

        <RowDefinition Height="7*"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="1*"/>
        <ColumnDefinition Width="1*"/>
    </Grid.ColumnDefinitions>
    <TextBox Text="Note Name" Margin="5" Name="Nname"/>
    <TextBox Text="User ID (internal only)" Margin="5" Grid.Column="1" Name="Uid"/>
    <TextBox Text="Note Text" Margin="5" Grid.Row="1" Name="Ntext"/>
    <StackPanel Orientation="Horizontal" FlowDirection="RightToLeft" Grid.Column="1"
Grid.Row="2">
        <Button Margin="5" Width="100" Content="Send" Click="Button_Click"/>
    </StackPanel>
</Grid>
</Border>
</Grid>
</Page>

```

CompanyNotesPage.xaml.cs

```

using Model_;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using VModel_;

namespace Visi0n._0.Pages.Company
{
    /// <summary>
    /// Interaction logic for CompanyNotesPage.xaml
    /// </summary>
    public partial class CompanyNotesPage : Page
    {

```

```

Corp _self;
User _user;

public CompanyNotesPage(User u)
{
    InitializeComponent();

    _user = u;
    _self = UserService.Corporative(_user); //assuming validity
}

private void Button_Click(object sender, RoutedEventArgs e)
{
    SendNote();
}

// one for all will be probably added in the future
private void SendNote()
{
    bool f = true;
    try
    {
        int id = int.Parse(Uid.Text);
    }
    catch { f = false; }
    if (f)
    {
        List<Person> emp = UserService.LaCampanella(_self);
        User u0 = UserService.Get(int.Parse(Uid.Text));
        User uu = null;
        foreach (User u in emp)
        {
            if (u._absId == u0._absId) uu = u0;
        }

        if (uu != null) { MessageBox.Show($"Attempting Note for id: {uu._absId}");
        NoteService.ScrapWrite(Nname.Text, Ntext.Text, uu._absId, _self._cid); }
        else MessageBox.Show("Make sure Uid is registered in the company");
    }
    else MessageBox.Show("Make sure input matches target types");
}
}
}

```

CompanyPersonalPage.xaml

```
<Page x:Class="Visi0n._0.Pages.Company.CompanyPersonalPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0.Pages.Company"
    xmlns:controls="clr-namespace:Visi0n._0"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="Page1">

    <Grid>
        <Border Background="AliceBlue">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="7*"/>
                    <ColumnDefinition Width="1*"/>
                </Grid.ColumnDefinitions>
                <Grid.RowDefinitions>
                    <RowDefinition Height="5*"/>
                    <RowDefinition Height="30*"/>
                </Grid.RowDefinitions>
                <Border Background="White" Margin="5,10,10,10" Grid.RowSpan="2"
Grid.Column="1">
                    <StackPanel Orientation="Vertical" Name="controlPanel">
                        <!--Button Margin="3" Height="70" Content="Calendar" Click="CaleB"/>
                        <Button Margin="3" Height="70" Content="Notes" Click="NotesB"/>
                        <Button Margin="3" Height="70" Content="Remindes" Click="RemB"/-->
                    </StackPanel>
                </Border>
                <Border Background="White" Margin="10,10,5,10" Grid.Row="0" Grid.RowSpan="2">
                    <Frame x:Name="OverruleF"/>
                </Border>
            </Grid>
        </Border>
    </Grid>
</Page>
```

CompanyPersonalPage.xaml.cs

```
using Model_;
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Visi0n._0.Pages.General;

namespace Visi0n._0.Pages.Company
{
    /// <summary>
    /// Interaction logic for Page1.xaml
    /// </summary>
    public partial class CompanyPersonalPage : Page
    {
        User _user;

        public CompanyPersonalPage(User u)
        {
            InitializeComponent();

            _user = u;

            controlPanel.Children.Add(new Menu001(OverruleF, _user, 1));
            controlPanel.Children.Add(new Menu001(OverruleF, _user, 2));
            controlPanel.Children.Add(new Menu001(OverruleF, _user));
        }

        private void CaleB(object sender, RoutedEventArgs e)
        {
            OverruleF.Navigate(new CalendarGP(OverruleF, _user));
        }

        private void NotesB(object sender, RoutedEventArgs e)
        {
            OverruleF.Navigate(new NotesGP(OverruleF, _user));
        }
    }
}

```

```

    }

    private void RemB(object sender, RoutedEventArgs e)
    {
        OverruleF.Navigate(new RemindersGP(OverruleF, _user));
    }
}
}

```

CompanyRemindersPage.xaml

```

<Page x:Class="Visi0n._0.Pages.Company.CompanyRemindersPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0.Pages.Company"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="CompanyRemindersPage">

    <Grid>
        <Border Margin="10" Background="AliceBlue">
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="7*" />
                    <RowDefinition Height="30*" />
                    <RowDefinition Height="7*" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="1*" />
                    <ColumnDefinition Width="1*" />
                </Grid.ColumnDefinitions>
                <TextBox Text="Reminder" Margin="5" Name="RText"/>
                <TextBox Text="User ID (internal only)" Margin="5" Grid.Column="1" Name="Uid"/>
                <StackPanel Orientation="Horizontal" FlowDirection="RightToLeft" Grid.Column="1"
Grid.Row="2">
                    <Button Margin="5" Width="100" Content="Send" Click="Button_Click"/>
                </StackPanel>
            </Grid>
        </Border>
    </Grid>
</Page>

```


CompanyRemindersPage.xaml.cs

```
using Model_;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Xml.Linq;
using VModel_;

namespace VisiOn._0.Pages.Company
{
    /// <summary>
    /// Interaction logic for CompanyRemindersPage.xaml
    /// </summary>
    public partial class CompanyRemindersPage : Page
    {
        Corp _self;

        public CompanyRemindersPage(User u)
        {
            InitializeComponent();

            _self = UserService.Corporative(u);
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            SendReminder();
        }

        private void SendReminder()
        {
            bool f = true;
            try
```

```

    {
        int id = int.Parse(Uid.Text);
    }
    catch { f = false; }
    if (f)
    {
        List<Person> emp = UserService.LaCampanella(_self);
        User u0 = UserService.Get(int.Parse(Uid.Text));
        User uu = null;
        foreach (User u in emp)
        {
            if (u._absId == u0._absId) uu = u0;
        }

        if (uu != null) { MessageBox.Show($"Attempting Note for id: {uu._absId}");
ReminderService.ListNewReminder(uu._absId, RText.Text, _self._cid); }
        else MessageBox.Show("Make sure Uid is registered in the company");
    }
    else MessageBox.Show("Make sure input matches target types");
}
}
}

```

__CaleActionGP.xaml

```

<Page x:Class="Visi0n._0.Pages.General.__CaleActionGP"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0.Pages.General"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="__CaleActionGP">

    <Grid Margin="5">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
        </Grid.RowDefinitions>
    </Grid>

```

```

</Grid.RowDefinitions>

<TextBox x:Name="AcName" Text="Event Name" Width="300" Height="40"
VerticalAlignment="Center" HorizontalAlignment="Center" Grid.ColumnSpan="3"/>
<TextBox x:Name="Detl" Text="Details" Width="300" Height="40"
VerticalAlignment="Center" HorizontalAlignment="Center" Grid.Row="1"
Grid.ColumnSpan="3"/>
<Button Content="Cancel" Height="50" Width="150" VerticalAlignment="Center"
HorizontalAlignment="Center" Grid.Row="2" Click="Back"/>
<Button Content="Save" Height="50" Width="150" VerticalAlignment="Center"
HorizontalAlignment="Center" Grid.Row="2" Grid.Column="2" Click="Saved"/>
</Grid>
</Page>

```

__CaleActionGP.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Model_;
using VModel_;

namespace Visi0n._0.Pages.General
{
    /// <summary>
    /// Interaction logic for __CaleActionGP.xaml
    /// </summary>
    public partial class __CaleActionGP : Page
    {
        Frame _frame;
        string _date;
        Event _target;
    }
}

```

```

User _user;

public __CaleActionGP(Frame frame, User u, string date, Event target = null)
{
    InitializeComponent();

    _frame = frame;
    _date = date;
    _target = target;
    _user = u;

    if (_target != null) { AcName.Text = _target._name; Detl.Text = _target._description; }
}

private void Save() => EventService.WriteEclipse(_user, _target, AcName.Text, Detl.Text,
_date); // reform to simplify other ends
/*{
    if (_target == null) // add new
    {
        _target = new Event();

        _target._name = AcName.Text;
        _target._description = Detl.Text;
        _target._ID = EventService.CreateNewID();
        _target._date = _date.Replace(" ", "");
        _target._uid = _user._absId;

        EventService.Write(_target);
    }
    else // edit selected
    {
        Event tor = _target.Copy();

        _target._name = AcName.Text;
        _target._description = Detl.Text;

        EventService.ReWrite(tor, _target);
    }
}*/

private void Saved(object sender, RoutedEventArgs e)
{
    Save();
    _frame.Navigate(new _CaleDayViewGP(_frame, _user, _date));
}

```

```

    }

    private void Back(object sender, RoutedEventArgs e)
    {
        _frame.Navigate(new _CaleDayViewGP(_frame, _user, _date));
    }
}
}

```

_CaleDayViewGP.xaml

```

<Page x:Class="Visi0n._0.Pages.General._CaleDayViewGP"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0.Pages.General"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="_CaleDayViewGP">

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="12*"/>
            <ColumnDefinition Width="3*"/>
        </Grid.ColumnDefinitions>

        <Border Grid.Column="0" Margin="10">

            <!--ListView x:Name="EventList" SelectionChanged="EventList_SelectionChanged">
                <ListView.View>
                    <GridView>
                        <GridViewColumn Header="ID" Width="80" DisplayMemberBinding="{Binding
_ID}"/>
                        <GridViewColumn Header="Name" Width="200"
DisplayMemberBinding="{Binding _name}"/>
                    </GridView>
                </ListView.View>
            </ListView-->

            <Grid x:Name="Table">
                <Grid.RowDefinitions>
                    <RowDefinition Height="10*"/>
                    <RowDefinition Height="10*"/>

```

```

        <RowDefinition Height="10*"/>
        <RowDefinition Height="10*"/>
        <RowDefinition Height="10*"/>
        <RowDefinition Height="10*"/>
        <RowDefinition Height="10*"/>
        <RowDefinition Height="10*"/>
    </Grid.RowDefinitions>

</Grid>

</Border>

<Grid Grid.Column="1" Margin="10">
    <Grid.RowDefinitions>
        <RowDefinition Height="5*"/>
        <RowDefinition Height="10*"/>
    </Grid.RowDefinitions>

    <Label x:Name="DateStringLabel" Content="Current date" FontSize="16"
VerticalAlignment="Top"/>

    <StackPanel Grid.Row="1" Orientation="Vertical" VerticalAlignment="Bottom">
        <Button x:Name="AD" Content="Add New" Height="50" Click="AD_Click"/>
        <Button x:Name="ED" Content="Edit" Height="50" Click="ED_Click"/>
        <Button x:Name="RE" Content="Remove" Height="50" Click="RE_Click"/>
        <Button x:Name="BackB" Content="Back" Height="50" Click="BackB_Click"/>
    </StackPanel>

</Grid>
</Grid>
</Page>

```

_CaleDayViewGP.xaml.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;

```

```

using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Model_;
using VModel_;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace Visi0n._0.Pages.General
{
    /// <summary>
    /// Interaction logic for _CaleDayViewGP.xaml
    /// </summary>
    public partial class _CaleDayViewGP : Page
    {
        Frame _frame;
        User _user;
        string _date;
        ObservableCollection<Event> _events = new ObservableCollection<Event>();
        Event _selected;

        int posCur;

        public _CaleDayViewGP(Frame frame, User u, string date, Event edited = null)
        {
            InitializeComponent();

            _frame = frame;
            _date = date;
            _user = u;
            _events = EventService.Vortex(_user, Date());

            // no idea what is this, but the thing works
            if (edited != null) {
                bool f = false;
                foreach (Event e in _events)
                {
                    if (e._ID == edited._ID) f = true;
                }
                if (f)

```

```

    {
        // update db (done)
    }
    else
    {
        _events.Add(edited);
        // add to db (done)
    }
    // check edit
    // if new - add
    // if edited - replace
    // write to db
    // identifier - ID
    // needs fixing (no longer)
}

posCur = 0;

DateStringLabel.Content = _date;

Inprint(_events);
}

private void BackB_Click(object sender, RoutedEventArgs e)
{
    _frame.Navigate(new CalendarGP(_frame, _user));
}

public void Inprint(ObservableCollection<Event> evs) // I have noticed a bug with "_" in ui,
but I'll ignore it for now
{
    foreach (Event e in evs) { AddItem(e); }
}
private void AddItem(Event ev)
{
    Label label = new Label() { Content = ev._name, Margin = new Thickness(5, 5, 5, 5) };
    label.Style = (Style)FindResource("Note01");
    Grid.SetRow(label, posCur);
    label.MouseDown += new MouseButtonEventHandler(label_MouseDown);
    Table.Children.Add(label);
    posCur++;
}

```



```
private string Date() => _date.Replace(" ", ""); // unspaced version - as in the database
```

```
private void label_MouseDown(object sender, MouseButtonEventArgs e)
{
    string name = ((Label)sender).Content.ToString();
    _selected = EventService.Find(name, _user, Date());

    if (_selected != null) MessageBox.Show(_selected._ID + "\n" + _selected._name + "\n" +
        _selected._description + "\n" + _selected._date + "\n" + _selected._uid + "\n" + _selected._cid);
    else MessageBox.Show("Error - event not found");
}
```

```
private void AD_Click(object sender, RoutedEventArgs e)
{
    if (_selected != null) MessageBox.Show("Selected: " + _selected._name);
    _frame.Navigate(new __CaleActionGP(_frame, _user, _date));
}
```

```
private void ED_Click(object sender, RoutedEventArgs e)
{
    if (_selected == null) MessageBox.Show("Please select an event first");
    else
    {
        MessageBox.Show("Selected: " + _selected._name);
        _frame.Navigate(new __CaleActionGP(_frame, _user, _date, _selected));
    }
}
```

```
private void RE_Click(object sender, RoutedEventArgs e)
{
    if (_selected == null) MessageBox.Show("Please select an event first");
    else
    {
        MessageBox.Show("Selected to be removed: " + _selected._name);
        EventService.Tear(_selected);
        //_events.Remove(_selected);
        Reload();
    }
}
```

```
/*private void ForceReload()
```

```

{
    Table.Children.Clear();
    posCur = 0;
    Inprint(_events);
}*/

private void Reload()
{
    _frame.Navigate(new _CaleDayViewGP(_frame, _user, _date));
}
}
}

```

_NotesViewGP.xaml

```

<Page x:Class="VisiOn._0.Pages.General._NotesViewGP"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:VisiOn._0.Pages.General"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="_NotesViewGP">

    <Grid Margin="5">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="5*"/>
            <RowDefinition Height="1*"/>
        </Grid.RowDefinitions>

        <TextBox x:Name="Name_" Text="Name" Margin="5,5,5,5" Grid.ColumnSpan="3"/>
        <TextBox x:Name="Text_" Text="Text" Margin="5,5,5,5" Grid.Row="1"
Grid.ColumnSpan="3"/>
        <Button x:Name="cancelB" Content="Cancel" Margin="5,5,5,5" Grid.Row="2"
Click="cancelB_Click"/>
        <Button x:Name="saveB" Content="Save" Margin="5,5,5,5" Grid.Row="2" Grid.Column="2"
Click="saveB_Click"/>
    </Grid>

```

```
<Button x:Name="deleteB" Content="Delete" Margin="5,5,5,5" Grid.Row="2"
Grid.Column="1" Click="deleteB_Click"/>
</Grid>
</Page>
```

_NotesViewGP.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Visi0n._0.Pages.Pesonal;
using Model_;
using VModel_;

namespace Visi0n._0.Pages.General
{
    /// <summary>
    /// Interaction logic for _NotesViewGP.xaml
    /// </summary>
    public partial class _NotesViewGP : Page
    {
        Frame _frame;
        User _user;
        NoteItem _item;
        int _type;

        public _NotesViewGP(Frame frame, User usr, NoteItem n)
        {
            InitializeComponent();
            _frame = frame;
            _user = usr;
            _item = n;
        }
    }
}
```

```

    Render(_item);
    //MessageBox.Show("" + _user._absId);
}

private void Render(NoteItem n)
{
    Name_.Text = n._name;
    Text_.Text = n._text;
    //MessageBox.Show($"Uid: {n._uid}");
}

private void SetBack()
{
    _frame.Navigate(new NotesGP(_frame, _user));
}

private void cancelB_Click(object sender, RoutedEventArgs e)
{
    SetBack();
}

private void saveB_Click(object sender, RoutedEventArgs e)
{
    MessageBox.Show($"Will be saved: ({_item._uid}, {_item._name }, {_item._text }) :({uid,
name, text}");
    if (_item._uid > 0)
    {
        NoteService.UpdateNote(_item, Name_.Text, Text_.Text);
    }
    else
    {
        _item._uid = _user._absId;
        _item._name = Name_.Text;
        _item._text = Text_.Text;
        NoteService.WriteNote(_item);
    }
    SetBack();
}

private void deleteB_Click(object sender, RoutedEventArgs e)
{
    MessageBox.Show($"Will be deleted: ({_item._uid}, {_item._name}, {_item._text}) :({uid,
name, text}");

```

```

        NoteService.DeleteNote(new NoteItem(_item._name, _item._text, _item._uid));
        SetBack();
    }
}
}

```

CalendarGP.xaml

```

<Page x:Class="Visi0n._0.Pages.General.CalendarGP"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0.Pages.General"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="CalendarGP">

    <Grid Margin="15">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="120*" />
            <ColumnDefinition Width="60*" />
        </Grid.ColumnDefinitions>

        <Border Background="AliceBlue">
            <Grid Margin="5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="50*" />
                    <RowDefinition Height="50*" />
                    <RowDefinition Height="300*" />
                </Grid.RowDefinitions>

                <Grid>
                    <Label x:Name="TheDateOfToday" Content="Today" FontSize="22"
                        FontWeight="Bold" HorizontalAlignment="Center" VerticalAlignment="Center" />
                </Grid>

                <Grid Grid.Row="1">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="10*" />
                        <ColumnDefinition Width="1*" />
                        <ColumnDefinition Width="10*" />
                    </Grid.ColumnDefinitions>

```

```

<Grid Margin="20,0">
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="5*" />
    <ColumnDefinition Width="10*" />
    <ColumnDefinition Width="5*" />
  </Grid.ColumnDefinitions>

  <Label x:Name="Month" Content="Month" FontSize="18" FontWeight="Bold"
HorizontalAlignment="Center" VerticalAlignment="Center" Grid.Column="1"/>
  <Button x:Name="preM" Style="{StaticResource CaleControl01}" Content=" prev
" Grid.Column="0" Click="preM_Click"/>
  <Button x:Name="nxtM" Style="{StaticResource CaleControl01}" Content=" next
" Grid.Column="2" Click="nxtM_Click" />
</Grid>
<Grid Grid.Column="2" Margin="20,0">
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="5*" />
    <ColumnDefinition Width="10*" />
    <ColumnDefinition Width="5*" />
  </Grid.ColumnDefinitions>

  <Label x:Name="Year" Content="Year" FontSize="18" FontWeight="Bold"
HorizontalAlignment="Center" VerticalAlignment="Center" Grid.Column="1"/>
  <Button x:Name="preY" Style="{StaticResource CaleControl01}" Content=" prev
" Grid.Column="0" Click="preY_Click"/>
  <Button x:Name="nxtY" Style="{StaticResource CaleControl01}" Content=" next
" Grid.Column="2" Click="nxtY_Click"/>
</Grid>
  <Label Content="|" Grid.Column="1" FontWeight="Bold" FontSize="18"
Foreground="#FF757575" VerticalAlignment="Center" HorizontalAlignment="Center"/>
</Grid>

<Grid x:Name="CaleGrid" Grid.Row="2" Margin="5">
  <Grid.RowDefinitions>
    <RowDefinition Height="10*" />
    <RowDefinition Height="10*" />
    <RowDefinition Height="10*" />
    <RowDefinition Height="10*" />
    <RowDefinition Height="10*" />
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="10*" />
    <ColumnDefinition Width="10*" />
    <ColumnDefinition Width="10*" />
  </Grid.ColumnDefinitions>

```

```
<ColumnDefinition Width="10*" />
<ColumnDefinition Width="10*" />
<ColumnDefinition Width="10*" />
<ColumnDefinition Width="10*" />
</Grid.ColumnDefinitions>
```

<!-- due to the setup each month will show 31 days until further notice -->
 <!-- and while I indeed can make the buttons in a better way I already wrote this so
 this shall stay -->

```
<RadioButton Style="{StaticResource CDay02}" x:Name="d1" Content="01"
Margin="5" Grid.Column="0" Grid.Row="0" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d2" Content="02"
Margin="5" Grid.Column="1" Grid.Row="0" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d3" Content="03"
Margin="5" Grid.Column="2" Grid.Row="0" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d4" Content="04"
Margin="5" Grid.Column="3" Grid.Row="0" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d5" Content="05"
Margin="5" Grid.Column="4" Grid.Row="0" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d6" Content="06"
Margin="5" Grid.Column="5" Grid.Row="0" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d7" Content="07"
Margin="5" Grid.Column="6" Grid.Row="0" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d8" Content="08"
Margin="5" Grid.Column="0" Grid.Row="1" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d9" Content="09"
Margin="5" Grid.Column="1" Grid.Row="1" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d10" Content="10"
Margin="5" Grid.Column="2" Grid.Row="1" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d11" Content="11"
Margin="5" Grid.Column="3" Grid.Row="1" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
```

```

<RadioButton Style="{StaticResource CDay02}" x:Name="d12" Content="12"
Margin="5" Grid.Column="4" Grid.Row="1" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d13" Content="13"
Margin="5" Grid.Column="5" Grid.Row="1" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d14" Content="14"
Margin="5" Grid.Column="6" Grid.Row="1" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d15" Content="15"
Margin="5" Grid.Column="0" Grid.Row="2" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d16" Content="16"
Margin="5" Grid.Column="1" Grid.Row="2" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d17" Content="17"
Margin="5" Grid.Column="2" Grid.Row="2" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d18" Content="18"
Margin="5" Grid.Column="3" Grid.Row="2" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d19" Content="19"
Margin="5" Grid.Column="4" Grid.Row="2" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d20" Content="20"
Margin="5" Grid.Column="5" Grid.Row="2" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d21" Content="21"
Margin="5" Grid.Column="6" Grid.Row="2" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d22" Content="22"
Margin="5" Grid.Column="0" Grid.Row="3" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d23" Content="23"
Margin="5" Grid.Column="1" Grid.Row="3" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d24" Content="24"
Margin="5" Grid.Column="2" Grid.Row="3" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
<RadioButton Style="{StaticResource CDay02}" x:Name="d25" Content="25"
Margin="5" Grid.Column="3" Grid.Row="3" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>

```



```

        <RadioButton Style="{StaticResource CDay02}" x:Name="d26" Content="26"
Margin="5" Grid.Column="4" Grid.Row="3" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
        <RadioButton Style="{StaticResource CDay02}" x:Name="d27" Content="27"
Margin="5" Grid.Column="5" Grid.Row="3" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
        <RadioButton Style="{StaticResource CDay02}" x:Name="d28" Content="28"
Margin="5" Grid.Column="6" Grid.Row="3" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
        <RadioButton Style="{StaticResource CDay02}" x:Name="d29" Content="29"
Margin="5" Grid.Column="0" Grid.Row="4" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
        <RadioButton Style="{StaticResource CDay02}" x:Name="d30" Content="30"
Margin="5" Grid.Column="1" Grid.Row="4" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>
        <RadioButton Style="{StaticResource CDay02}" x:Name="d31" Content="31"
Margin="5" Grid.Column="2" Grid.Row="4" Checked="d_Checked"
MouseDoubleClick="d_MouseDoubleClick"/>

```

```

    </Grid>

```

```

    </Grid>

```

```

</Border>

```

```

<Border Background="#FFF5EADC" Grid.Column="1">

```

```

    <Grid Margin="5">

```

```

        <Grid.RowDefinitions>

```

```

            <RowDefinition Height="1*" />

```

```

            <RowDefinition Height="0*" />

```

```

        </Grid.RowDefinitions>

```

```

        <Label x:Name="DateText" Content="-" />

```

```

    </Grid>

```

```

</Border>

```

```

</Grid>

```

```

</Page>

```

CalendarGP.xaml.cs

```

using System;
using System.Collections.Generic;

```

```

using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Visi0n._0.Pages.Pesonal;
using Model_;
using VModel_;

namespace Visi0n._0.Pages.General
{
    /// <summary>
    /// no idea what I have done here
    /// </summary>
    public partial class CalendarGP : Page
    {
        Frame _frame;
        int _day;
        int[] Marr;
        int MarrIndx;
        int Ynum;
        string __date;

        User _usr;
        //ObservableCollection<Event> _events; //same as list, but might be better sometimes
        DateCaleReader dc;

        public CalendarGP(Frame frame, User usr)
        {
            InitializeComponent();
            _frame = frame;
            _usr = usr;

            // _events = EventService.Load(_usr);
        }
    }
}

```

```

Marr = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
Ynum = 2024;

MarrIndx = 10;
Month.Content = Marr[MarrIndx].ToString();
Year.Content = Ynum.ToString();
_day = 1;
UpdateDate();

dc = new DateCaleReader(new RadioButton[]
{ d1, d2, d3, d4, d5, d6, d7, d8, d9, d10,
  d11, d12, d13, d14, d15, d16, d17, d18, d19, d20,
  d21, d22, d23, d24, d25, d26, d27, d28, d29, d30, d31 }, _usr);
dc.ReColor(__date);
}

private void UpdateDate()
{
    string ddd;
    string mmm;
    if (_day < 10) ddd = "0" + _day; else ddd = "" + _day;
    if (Marr[MarrIndx] < 10) mmm = "0" + Marr[MarrIndx]; else mmm = "" + Marr[MarrIndx];
    __date = " " + ddd + " / " + mmm + " / " + Ynum;
    TheDateOfToday.Content = __date;
}

private void UpdateColumn()
{
    DateText.Content = __date;
    DateText.Content += "\n\n";
    ObservableCollection<Event> evs = EventService.Vortex(_usr, __date.Replace(" ", "")); //
ideal placement, returns all events at date for user
    foreach (Event e in evs)
        DateText.Content += e._name + "\n";
}

private void preM_Click(object sender, RoutedEventArgs e)
{
    if (MarrIndx < 1) MarrIndx = 11;
    else
        MarrIndx--;

    Month.Content = Marr[MarrIndx].ToString();
    UpdateDate();
}

```

```

        UpdateColumn();
        dc.ReColor(__date);
    }

private void nxtM_Click(object sender, RoutedEventArgs e)
{
    if (MarrIndx > 10) MarrIndx = 0;
    else
        MarrIndx++;

    Month.Content = Marr[MarrIndx].ToString();
    UpdateDate();
    UpdateColumn();
    dc.ReColor(__date);
}

private void preY_Click(object sender, RoutedEventArgs e)
{
    Ynum--;
    Year.Content = Ynum.ToString();
    UpdateDate();
    UpdateColumn();
    dc.ReColor(__date);
}

private void nxtY_Click(object sender, RoutedEventArgs e)
{
    Ynum++;
    Year.Content = Ynum.ToString();
    UpdateDate();
    UpdateColumn();
    dc.ReColor(__date);
}

private void d_Checked(object sender, RoutedEventArgs e)
{
    _day = int.Parse(((RadioButton)sender).Content.ToString());
    UpdateDate();
    UpdateColumn();
}

private void d_MouseDoubleClick(object sender, MouseButtonEventArgs e)
{

```

```

//ObservableCollection<Event> storm = EventService.Vortex(_usr, __date.Replace(" ",
""));
_frame.Navigate(new _CaleDayViewGP(_frame, _usr, __date));
}

```

```

internal class DateCaleReader
{
    User uu;
    RadioButton[] Total;

    public DateCaleReader(RadioButton[] arr, User uu)
    {
        Total = arr;
        this.uu = uu;
    }

    public void ReColor(string rawdate)
    {
        ColorRaw();
        string _m = rawdate.Replace(" ", "").Split("/")[1]; string _y = rawdate.Replace(" ",
"").Split("/")[2];
        Color ColorDays = new Color(ColorPersonal); ColorDays(_m, _y);
        ColorDays = new Color(ColorCorp); ColorDays(_m, _y); // more of a proof of concept
    }

    public delegate void Color(string _m, string _y);

    public void ColorRaw()
    {
        foreach (RadioButton r in Total)
        {
            r.Background = Brushes.Beige;
        }
    }

    public void ColorPersonal(string _m, string _y)
    {
        EventService.ActivityDays Activity = new
EventService.ActivityDays(EventService.ActiveDays);
        List<string> days = Activity(uu._absId, _m, _y);
        foreach (string d in days)
        {

```

```

        foreach(RadioButton r in Total) if (r.Name == "d" + int.Parse(d)) r.Background =
Brushes.Pink;
    }
}

public void ColorCorp(string _m, string _y)
{
    EventService.ActivityDays Activity = new
EventService.ActivityDays(EventService.SuperActiveDays);
    List<string> days = Activity(uu._absId, _m, _y);
    foreach (string d in days)
    {
        foreach (RadioButton r in Total) if (r.Name == "d" + int.Parse(d)) r.Background =
Brushes.Plum;
    }
}

RadioButton[] ReturnAll() => Total;
}
}
}

```

NotesGP.xaml

```

<Page x:Class="Visi0n._0.Pages.General.NotesGP"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0.Pages.General"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="NotesGP">

    <Grid>
        <Grid Margin="5,0,5,5">
            <Grid Margin="5">
                <Grid.RowDefinitions>
                    <RowDefinition Height="70*" />
                    <RowDefinition Height="10*" />
                </Grid.RowDefinitions>
                <Border Background="AliceBlue">
                    <Grid x:Name="Table">
                        <Grid.RowDefinitions>

```

```

        <RowDefinition Height="10**"/>
        <RowDefinition Height="10**"/>
        <RowDefinition Height="10**"/>
        <RowDefinition Height="10**"/>
        <RowDefinition Height="10**"/>
        <RowDefinition Height="10**"/>
        <RowDefinition Height="10**"/>
        <RowDefinition Height="10**"/>
    </Grid.RowDefinitions>

    <!--Label Style="{StaticResource Note01}" Content="Note" Margin="5"
Grid.Row="0"/>
    <Label Style="{StaticResource Note01}" Content="Note" Margin="5"
Grid.Row="1"/>
    <Label Style="{StaticResource Note01}" Content="Note" Margin="5"
Grid.Row="2"/-->

    </Grid>
</Border>
<Border Grid.Row="1" Background="AliceBlue">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="10**"/>
            <ColumnDefinition Width="80**"/>
            <ColumnDefinition Width="10**"/>
        </Grid.ColumnDefinitions>
        <Label Style="{StaticResource Note01}" Content="+" Margin="5" Grid.Row="7"
MouseDoubleClick="Label_MouseDoubleClick" Grid.Column="1"/>
        <Button x:Name="PosPlus" Content="next" Grid.Column="3" Margin="1"
Click="PosPlus_Click"/>
        <Button x:Name="PosMinus" Content="prev" Grid.Column="0" Margin="1"
Click="PosMinus_Click"/>
    </Grid>
</Border>
</Grid>
</Grid>
</Grid>
</Page>

```

NotesGP.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;

```

```

using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using VisiOn._0.Pages.Pesonal;
using Model_;
using VModel_;

namespace VisiOn._0.Pages.General
{
    /// <summary>
    /// Interaction logic for NotesGP.xaml
    /// </summary>
    public partial class NotesGP : Page
    {
        Frame _frame;
        NotelItem _note;
        int posCur;

        User _usr;

        List<NotelItem> _notes;
        int pagePos;

        public NotesGP(Frame frame, User usr = null, int pagePos = 0)
        {
            InitializeComponent();
            _frame = frame;

            posCur = 0;
            _notes = new List<NotelItem>();

            if (usr != null) { _usr = usr; }
            else { _usr = new User(); }
            //MessageBox.Show($"UID = {_usr._absId}");

            this.pagePos = pagePos;
        }
    }
}

```



```

        SetNotes(_usr);
    }

    public void SetNotes(User usr)
    {
        List<NoteItem> notes = NoteService.GetNotes(usr);
        foreach (NoteItem n in notes)
        {
            _notes.Add(n);
        }
        DrawTable(_notes, pagePos);
    }

    private void Label_MouseDoubleClick(object sender, MouseButtonEventArgs e)
    {
        _frame.Navigate(new _NotesViewGP(_frame, _usr, new NoteItem()));
    }

    private void AddItem(NoteItem nt)
    {
        Label label = new Label() { Content = nt._name, Margin = new Thickness(5, 5, 5, 5) };
        label.Style = (Style)FindResource("Note01");
        if (nt._cid != "--") label.Style = (Style)FindResource("Note02");
        Grid.SetRow(label, posCur);
        label.MouseDown += new MouseButtonEventHandler(label_MouseDown);
        Table.Children.Add(label);
        posCur++;
    }

    private void DrawTable(List<NoteItem> list, int pos)
    {
        Table.Children.Clear(); posCur = 0;
        for (int i = pos * 8; i < (pos + 1) * 8; i++)
        {
            if (i < list.Count) AddItem(list[i]);
        }
    }

    private void label_MouseDown(object sender, MouseButtonEventArgs e)
    {

```

```

        NoteItem nt = NoteService.Find(_usr, ((Label)sender).Content.ToString());
        if (nt._uid == -1) MessageBox.Show("Note not found - an error occurred");
        else
        {
            MessageBox.Show($"Selected: Uid = {nt._uid}, Name = {nt._name}");
            _frame.Navigate(new _NotesViewGP(_frame, _usr, nt));
        }
    }

    private void PosMinus_Click(object sender, RoutedEventArgs e)
    {
        if (pagePos > 0) { pagePos--; } DrawTable(_notes, pagePos);
    }

    private void PosPlus_Click(object sender, RoutedEventArgs e)
    {
        pagePos++; DrawTable(_notes, pagePos);
    }
}

```

RemindersGP.xaml

```

<Page x:Class="Visi0n._0.Pages.General.RemindersGP"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0.Pages.General"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="RemindersGP">

    <Grid Margin="5,0,5,5">
        <Grid.RowDefinitions>
            <RowDefinition Height="120*" />
            <RowDefinition Height="10*" />
        </Grid.RowDefinitions>

        <Border Background="AliceBlue" Margin="5,5,5,0">
            <Grid x:Name="Rtable">
                <Grid.RowDefinitions>
                    <RowDefinition Height="10*" />
                    <RowDefinition Height="10*" />
                </Grid.RowDefinitions>
            </Grid>
        </Border>
    </Grid>

```

```

        <RowDefinition Height="10*" />
        <RowDefinition Height="10*" />
        <RowDefinition Height="10*" />
        <RowDefinition Height="10*" />
        <RowDefinition Height="10*" />
        <RowDefinition Height="10*" />
        <RowDefinition Height="10*" />
        <RowDefinition Height="10*" />
        <RowDefinition Height="10*" />
        <RowDefinition Height="10*" />
    </Grid.RowDefinitions>

</Grid>

</Border>

<Border Margin="5,0,5,5" Grid.Row="1">
    <Grid Background="AliceBlue">
        <Grid.RowDefinitions>
        </Grid.RowDefinitions>

        <Grid Grid.Row="1">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="75*" />
                <ColumnDefinition Width="625*" />
                <ColumnDefinition Width="150*" />
                <ColumnDefinition Width="75*" />
            </Grid.ColumnDefinitions>

            <TextBox x:Name="TextName" Margin="1" Grid.Column="1" Text="New
Reminder" />
            <Button Content="Add" Margin="1" Grid.Column="2" Click="AddRCommand" />
            <Button x:Name="PosPlus" Content="next" Grid.Column="3" Margin="1"
Click="PosPlus_Click" />
            <Button x:Name="Minus" Content="prev" Grid.Column="0" Margin="1"
Click="Minus_Click" />
        </Grid>
    </Grid>
</Border>

</Grid>
</Page>

```

RemindersGP.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection.Emit;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Model_;
using VModel_;

namespace Visi0n._0.Pages.General
{
    /// <summary>
    /// Interaction logic for RemindersGP.xaml
    /// </summary>
    public partial class RemindersGP : Page
    {
        Frame _frame;
        int posCur;

        User _usr;
        Person _prsn;

        List<Reminder> listed;
        int ppage;

        public RemindersGP(Frame frame, User usr = null, int pagePos = 0)
        {
            InitializeComponent();
            _frame = frame;
            posCur = 0;
            ppage = pagePos;
        }
    }
}
```

```

        listed = new List<Reminder>();

        if (usr != null) { _usr = usr; _prsn = UserService.LaPersona(_usr); }
        else _usr = new User();

        Load(_usr);
        //MessageBox.Show("Listed: " + listed.Count);
    }

    public void Load(User usr)
    {
        List<Reminder> reminders = ReminderService.GetReminders(usr);
        foreach (Reminder r in reminders)
        {
            AddReminder(r); // it draws it 12 times lmao
        }
        DrawTable(listed, ppage);
    }

    private void AddRCommand(object sender, RoutedEventArgs e)
    {
        AddReminder();
        DrawTable(listed, ppage);
    }

    private void AddReminder(Reminder r = null)
    {
        if (r == null)
        {
            r = new Reminder(_usr._absId, TextName.Text);
            //if (_prsn != null) r._cid = _prsn._cid; // for testing
            ReminderService.ListReminder(r);
        }
        listed.Add(r);
        //DrawTable(listed, ppage);
    }

    private void DrawR(Reminder r) // sets reminder in ui
    {
        CheckBox ck = new CheckBox() { Margin = new Thickness(4, 4, 4, 4) };
        ck.Content = r._text;
        ck.Style = (Style)FindResource("Reminder01");
        if (r._cid != "--") ck.Style = (Style)FindResource("Reminder02");
        ck.Checked += new RoutedEventHandler(CheckBox_Checked);
        Grid.SetRow(ck, posCur);
        posCur++;
    }

```

```

        Rtable.Children.Add(ck);
    }
    private void DrawTable(List<Reminder> list, int pos) // draws 12 reminders per page
    {
        Rtable.Children.Clear(); posCur = 0;
        for(int i = pos * 12; i < (pos + 1) * 12; i++)
        {
            if (i < list.Count) DrawR(list[i]);
        }
    }

    // on: Checked=""
    private void CheckBox_Checked(object sender, RoutedEventArgs e)
    {
        Reminder r = new Reminder(_usr._absId, ((CheckBox)sender).Content.ToString());
        MessageBox.Show("To be removed: " + r._uid + " (uid), " + r._text + " (text)");
        ReminderService.DropReminder(r);
        Refresh();
    }

    private void Refresh() { _frame.Navigate(new RemindersGP(_frame, _usr, ppage)); }

    private void PosPlus_Click(object sender, RoutedEventArgs e)
    {
        ppage++; DrawTable(listed, ppage);
    }

    private void Minus_Click(object sender, RoutedEventArgs e)
    {
        if (ppage > 0) ppage--;
        DrawTable(listed, ppage);
    }
}
}

```

LoginPage.xaml

```

<Page x:Class="Visi0n._0.Pages.LoginPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0.Pages"
    mc:Ignorable="d"

```

```
d:DesignHeight="450" d:DesignWidth="800"
Title="PLoginPage">

<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="100*" />
    <ColumnDefinition Width="100*" />
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="100*" />
    <RowDefinition Height="100*" />
    <RowDefinition Height="100*" />
    <RowDefinition Height="100*" />
  </Grid.RowDefinitions>

  <TextBox x:Name="usernameBox" Text="Username" Style="{StaticResource TextBox01}"
HorizontalAlignment="Center" VerticalAlignment="Center" Height="50" Width="300"
Grid.ColumnSpan="2" MouseDoubleClick="TextBoxClear" />
  <TextBox x:Name="usrpassBox" Text="Password" Style="{StaticResource TextBox01}"
HorizontalAlignment="Center" VerticalAlignment="Center" Height="50" Width="300"
Grid.Row="1" Grid.ColumnSpan="2" MouseDoubleClick="TextBoxClear" />
  <Button x:Name="usrLogin" Style="{StaticResource Button03}" Content="Login"
Grid.Row="2" Height="50" Width="100" Grid.ColumnSpan="2" Click="Start"/>
  <Button x:Name="usrNew" Style="{StaticResource Button03}" Content="Create New"
Grid.Row="3" Height="50" Width="100" Click="usrNew_Click"/>
  <Button x:Name="back" Style="{StaticResource Button03}" Content="Back" Grid.Row="3"
Grid.Column="1" Height="50" Width="100" Click="back_Click"/>
</Grid>
</Page>
```

LoginPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
```

```

using System.Windows.Navigation;
using System.Windows.Shapes;
using Model_;
using VModel_;

namespace Visi0n._0.Pages
{
    /// <summary>
    /// Interaction logic for PLoginPage.xaml
    /// </summary>
    public partial class LoginPage : Page
    {
        Window _win;
        Frame _frame;
        int _type;

        public LoginPage(Frame frame, Window win, int tp)
        {
            InitializeComponent();
            _win = win;
            _frame = frame;
            _type = tp;
        }

        private void Start(object sender, RoutedEventArgs e)
        {
            //MessageBox.Show(usurnameBox.Text + " " + usrpasBox.Text);

            if (UserService.LoginAtt(usurnameBox.Text, usrpasBox.Text, _type))
            {
                int idd = new UserDB().FindID(usurnameBox.Text);
                if (_type == 1) new PersonalW(UserService.Get(idd)).Show();
                else new CompanyW(UserService.Get(idd)).Show();
                _win.Close();
            }
            else
            {
                MessageBox.Show("Login info incorrect");
            }
        }

        private void back_Click(object sender, RoutedEventArgs e)
        {
            _frame.Navigate(new StartPage(_frame, _win));
        }
    }
}

```



```

    }

    private void usrNew_Click(object sender, RoutedEventArgs e)
    {
        _frame.Navigate(new NewUsr(_frame, _win, _type));
    }

    private void TextBoxClear(object sender, MouseButtonEventArgs e)
    {
        ((TextBox)sender).Text = string.Empty;
    }
}
}

```

NewUsr.xaml

```

<Page x:Class="Visi0n._0.Pages.NewUsr"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0.Pages"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="NewUsr">

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="100*" />
            <RowDefinition Height="100*" />
            <RowDefinition Height="100*" />
            <RowDefinition Height="100*" />
        </Grid.RowDefinitions>

        <TextBox x:Name="usernameBox" Style="{StaticResource TextBox01}" Text="Username"
            HorizontalAlignment="Center" VerticalAlignment="Center" Height="50" Width="300"
            MouseDoubleClick="TextBoxClear" />
        <TextBox x:Name="usrpassBox" Style="{StaticResource TextBox01}" Text="Password"
            HorizontalAlignment="Center" VerticalAlignment="Center" Height="50" Width="300"
            Grid.Row="1" MouseDoubleClick="TextBoxClear" />
        <Button x:Name="usrCreate" Style="{StaticResource Button03}" Content="Create"
            Grid.Row="2" Height="50" Width="100" Click="usrCreate_Click" />
        <Button x:Name="back" Style="{StaticResource Button03}" Content="Back" Grid.Row="3"
            Grid.Column="1" Height="50" Width="100" Click="back_Click" />
    </Grid>

```

```
</Grid>
</Page>
```

NewUsr.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using VModel_;
using Model_;

namespace Visi0n._0.Pages
{
    /// <summary>
    /// Interaction logic for NewUsr.xaml
    /// </summary>
    public partial class NewUsr : Page
    {
        Frame _frame;
        Window _win;
        int _type;

        public NewUsr(Frame frame, Window win, int type)
        {
            InitializeComponent();
            _frame = frame;
            _win = win;
            _type = type;
        }

        private void back_Click(object sender, RoutedEventArgs e)
        {
            _frame.Navigate(new LoginPage(_frame, _win, _type));
        }
    }
}
```

```

    }

    private void TextBoxClear(object sender, MouseButtonEventArgs e)
    {
        ((TextBox)sender).Text = string.Empty;
    }

    private void usrCreate_Click(object sender, RoutedEventArgs e)
    {
        User uu = UserService.MakeUser(usrnameBox.Text, usrpassBox.Text, _type);
        MessageBox.Show("attempted creation for: " +
            usrnameBox.Text + ", " + usrpassBox.Text + ", " + _type + " with next id");
        _frame.Navigate(new TypeUserCreatePage(uu, _type, _win));
    }
}
}

```

StartPage.xaml

```

<Page x:Class="VisiOn._0.Pages.StartPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:VisiOn._0.Pages"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="StartPage">

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="100*" />
            <RowDefinition Height="100*" />
        </Grid.RowDefinitions>

        <Button x:Name="LLp" Style="{StaticResource Button01}" Content="Personal" Height="50"
            Width="200" VerticalAlignment="Center" HorizontalAlignment="Center" Click="SetLoginP"/>
        <Button x:Name="LLc" Style="{StaticResource Button02}" Content="Corporate"
            Height="50" Width="200" VerticalAlignment="Center" HorizontalAlignment="Center"
            Grid.Row="1" Click="SetLoginC"/>
    </Grid>
</Page>

```

StartPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace Visi0n._0.Pages
{
    /// <summary>
    /// Interaction logic for StartPage.xaml
    /// </summary>
    public partial class StartPage : Page
    {
        Frame target;
        Window _win;

        public StartPage(Frame frame, Window win)
        {
            InitializeComponent();
            this.target = frame;
            this._win = win;
        }

        private void SetLoginP(object sender, RoutedEventArgs e)
        {
            target.Navigate(new LoginPage(target, _win, 1));
        }
        private void SetLoginC(object sender, RoutedEventArgs e)
        {
            target.Navigate(new LoginPage(target, _win, 2));
        }
    }
}
```

TypeUserCreatePage.xaml

```
<Page x:Class="Visi0n._0.Pages.TypeUserCreatePage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0.Pages"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="TypeUserCreatePage">

    <Grid>
        <Grid Name="PersonG" Visibility="Collapsed">
            <Grid.RowDefinitions>
                <RowDefinition Height="10*"/>
                <RowDefinition Height="10*"/>
                <RowDefinition Height="15*"/>
                <RowDefinition Height="10*"/>
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="7*"/>
                <ColumnDefinition Width="3*"/>
            </Grid.ColumnDefinitions>
            <TextBox x:Name="FNameT" Style="{StaticResource TextBox01}"
                HorizontalAlignment="Center" VerticalAlignment="Center" Height="50" Width="300" Text="First
                Name" MouseDoubleClick="TextBoxClear"/>
            <TextBox x:Name="LNameT" Style="{StaticResource TextBox01}"
                HorizontalAlignment="Center" VerticalAlignment="Center" Height="50" Width="300"
                Grid.Row="1" Text="Last Name" MouseDoubleClick="TextBoxClear"/>
            <TextBox x:Name="CorpT" Style="{StaticResource TextBox01}"
                HorizontalAlignment="Center" VerticalAlignment="Center" Height="50" Width="120"
                Grid.Column="1" Text="Cid (--)" MouseDoubleClick="TextBoxClear"/>
            <Button x:Name="CreatePersonB" Style="{StaticResource Button03}" Content="Create"
                Grid.Row="3" Grid.Column="1" Height="50" Width="100" Click="CreatePersonB_Click"/>
        </Grid>
        <Grid x:Name="CorpG">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="7*"/>
                <ColumnDefinition Width="3*"/>
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="10*"/>
                <RowDefinition Height="10*"/>
            </Grid.RowDefinitions>
        </Grid>
    </Grid>
```

```

        <RowDefinition Height="15*" />
        <RowDefinition Height="10*" />
    </Grid.RowDefinitions>
    <TextBox x:Name="CNameT" Style="{StaticResource TextBox01}"
HorizontalAlignment="Center" VerticalAlignment="Center" Height="50" Width="300"
Text="Company Name" MouseDoubleClick="TextBoxClear" />
    <TextBox x:Name="CIDT" Style="{StaticResource TextBox01}"
HorizontalAlignment="Center" VerticalAlignment="Center" Height="50" Width="120"
Grid.Column="1" Text="Cid" MouseDoubleClick="TextBoxClear" />
    <Button x:Name="CreateCorpB" Style="{StaticResource Button03}" Content="Create"
Grid.Row="3" Grid.Column="1" Height="50" Width="100" Click="CreateCorpB_Click" />
</Grid>
</Grid>
</Page>

```

TypeUserCreatePage.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using VModel_;
using Model_;

namespace Visi0n._0.Pages
{
    /// <summary>
    /// Interaction logic for TypeUserCreatePage.xaml
    /// </summary>
    public partial class TypeUserCreatePage : Page
    {
        User _user;
        int _type;
        Window _win;
    }
}

```

```

string _cid;
string _cName;
string _fName;
string _IName;

public TypeUserCreatePage(User user, int type, Window wind)
{
    InitializeComponent();

    _user = user;
    _win = wind;

    if (type == 1) { _type = 1; PersonG.Visibility = Visibility.Visible; CorpG.Visibility =
Visibility.Collapsed; }
    else if (type == 2) { _type = 2; PersonG.Visibility = Visibility.Collapsed; CorpG.Visibility =
Visibility.Visible; }
}

private void SetUp()
{
    if (_type == 1)
    {
        _fName = FNameT.Text;
        _IName = LNameT.Text;
        _cid = CorpT.Text;
        if (CorpT.Text == "Cid (--)" _cid = "--";
        UserService.CreatePerson(_user, _fName, _IName, _cid);
        MessageBox.Show("Attempted (P): " + _user._absId + ", " + _fName + ", " + _IName +
", " + _cid);
    }
    else if (_type == 2)
    {
        _cName = CNameT.Text;
        _cid = CIDT.Text;
        UserService.CreateCorp(_user, _cid, _cName);
        MessageBox.Show("Attempted (C): " + _user._absId + ", " + _cName + ", " + _cid);
    }
    new LoginW().Show();
    this._win.Close();
}

private void TextBoxClear(object sender, MouseButtonEventArgs e)
{

```

```

        ((TextBox)sender).Text = string.Empty;
    }

    private void CreatePersonB_Click(object sender, RoutedEventArgs e)
    {
        SetUp();
    }

    private void CreateCorpB_Click(object sender, RoutedEventArgs e)
    {
        SetUp();
    }
}

```

PersonalHome.xaml

```

<Page x:Class="Visi0n._0.Pages.Pesonal.PersonalHome"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Visi0n._0.Pages.Pesonal"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="PersonalHome">

    <Grid Margin="5,0,5,5">
        <Border Margin="5">
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="1*" />
                    <RowDefinition Height="2*" />
                    <RowDefinition Height="2*" />
                    <RowDefinition Height="2*" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    </Grid.ColumnDefinitions>

                    <Label x:Name="NameTitle" Style="{StaticResource LL01}" Content="Welcome
[fname] [lname]" VerticalAlignment="Center" FontSize="20" Margin="10,0"/>

                    <StackPanel Grid.Row="1" Margin="3" FlowDirection="LeftToRight"
Orientation="Horizontal">

```



```

        <Border Style="{StaticResource PHcolor1}">
            <Label Name="Lcal" Style="{StaticResource LR01}" Content="Use the
calendar" MouseDoubleClick="Label_MouseDoubleClick"/>
        </Border>
        <Border Style="{StaticResource PHcolor2}">
            <Label Name="Lnot" Style="{StaticResource LR01}" Content="Write notes"
MouseDoubleClick="Label_MouseDoubleClick"/>
        </Border>
        <Border Style="{StaticResource PHcolor3}">
            <Label Name="Lrem" Style="{StaticResource LR02}" Content="Set reminders"
MouseDoubleClick="Label_MouseDoubleClick"/>
        </Border>
    </StackPanel>
</Grid>
</Border>
</Grid>
</Page>

```

PersonalHome.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Model_;
using Visi0n._0.Pages.General;

namespace Visi0n._0.Pages.Pesonal
{
    /// <summary>
    /// Interaction logic for PersonalHome.xaml
    /// </summary>
    public partial class PersonalHome : Page

```

```

{
    Person _person;
    Frame _frame;

    public PersonalHome(Frame frame, Person p = null)
    {
        InitializeComponent();

        _frame = frame;
        if (p != null) _person = p;

        if (_person != null) NameTitle.Content = $"Welcome {_person._fName}
{_person._lName}";

        //MessageBox.Show(_person._usrName + "; " + _person._pwd + "; " + _person._absId +
"; ");
    }

    private void Label_MouseDoubleClick(object sender, MouseButtonEventArgs e)
    {
        switch (((Label)sender).Content.ToString())
        {
            case "Use the calendar":
                _frame.Navigate(new CalendarGP(_frame, (User)_person));
                break;
            case "Write notes":
                _frame.Navigate(new NotesGP(_frame, (User)_person));
                break;
            case "Set reminders":
                _frame.Navigate(new RemindersGP(_frame, (User)_person));
                break;
            default:
                MessageBox.Show("How did you manage?");
                break;
        }
    }
}
}
}

```

CompanyW.xaml

```

<Window x:Class="Visi0n._0.CompanyW"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

```

```

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:local="clr-namespace:Visi0n._0"
mc:Ignorable="d"
Title="Visi0n - Company" Height="450" Width="800"
MinHeight="450" MinWidth="800"
WindowStyle="None" AllowsTransparency="True"
MouseLeftButtonDown="Drag">

<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="8*"/>
    <ColumnDefinition Width="2*"/>
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="6*"/>
    <RowDefinition Height="33*"/>
  </Grid.RowDefinitions>

  <StackPanel x:Name="ControlButtons" Margin="5" Orientation="Horizontal"
  FlowDirection="RightToLeft" Grid.Column="1" Grid.Row="0">
    <Button x:Name="Quit" Width="65" Content="Quit" Margin="3,0" Click="Quit_Click"/>
    <Button x:Name="LogOut" Width="65" Content="Log Out" Margin="3,0"
  Click="LogOut_Click"/>
  </StackPanel>
  <StackPanel x:Name="ActionButtons" Margin="5" Orientation="Horizontal"
  FlowDirection="LeftToRight" Grid.Column="0" Grid.Row="0">
    <Grid x:Name="Stats" Width="120" Margin="12,3">
      <StackPanel Orientation="Vertical" FlowDirection="LeftToRight">
        <Label x:Name="CompanyNameLabel" Content="Company Name" FontSize="11"
        FontWeight="Bold"></Label>
        <Label x:Name="currentStateLabel" Content="Current: " FontSize="11"
        FontWeight="Bold"></Label>
      </StackPanel>
    </Grid>
    <Button x:Name="CompanyDetailsB" Width="65" Content="Details" Margin="3,0"
  Click="CompanyDetailsB_Click"/>
    <Button x:Name="CalendarB" Width="65" Content="Calendar" Margin="3,0"
  Click="CaleClick"/>
    <Button x:Name="NotesB" Width="65" Content="Notes" Margin="3,0"
  Click="NotesB_Click"/>
    <Button x:Name="RemindersB" Width="65" Content="Reminders" Margin="3,0"
  Click="RemindersB_Click"/>

```

```

        <Button x:Name="SelfPB" Width="65" Content="Self" Margin="3,0"
Click="SelfPB_Click"/>
    </StackPanel>
    <Grid x:Name="MainPanel" Margin="5" Grid.ColumnSpan="2" Grid.Row="1">
        <Frame x:Name="Screen" NavigationUIVisibility="Hidden">

            </Frame>
        </Grid>
    </Grid>
</Window>

```

CompanyW.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using Visi0n._0.Pages.Company;
using Visi0n._0.Pages.General;
using Visi0n._0.Pages.Pesonal;
using Model_;
using VModel_;

namespace Visi0n._0
{
    /// <summary>
    /// Interaction logic for Company.xaml
    /// </summary>
    public partial class CompanyW : Window
    {
        User _user;

        public CompanyW(User usr = null)
        {
            InitializeComponent();

```

```

        if (usr != null) UserSetUp(usr);

        Screen.Navigate(new CompanyDetailsPage(_user));
        currentStateLabel.Content = "Current: Details";
    }
    private void UserSetUp(User u)
    {
        _user = u;
        Corp c = UserService.Corporative(u);
        if (c != null) CompanyNameLabel.Content = c._cName + " (" + c._cid + ")";
    }

    private void Drag(object sender, RoutedEventArgs e)
    {
        try { DragMove(); }
        catch { }
    }

    private void Quit_Click(object sender, RoutedEventArgs e)
    {
        this.Close();
    }

    private void LogOut_Click(object sender, RoutedEventArgs e)
    {
        new LoginW().Show();
        this.Close();
    }

    private void CalendarB_Click(object sender, RoutedEventArgs e)
    {
        Screen.Navigate(new CalendarGP(Screen, _user));
        currentStateLabel.Content = "Current: Calendar";
    }

    private void NotesB_Click(object sender, RoutedEventArgs e)
    {
        Screen.Navigate(new CompanyNotesPage(_user));
        currentStateLabel.Content = "Current: Notes";
    }

```

```

private void RemindersB_Click(object sender, RoutedEventArgs e)
{
    Screen.Navigate(new CompanyRemindersPage(_user));
    currentStateLabel.Content = "Current: Reminders";
}

private void CompanyDetailsB_Click(object sender, RoutedEventArgs e)
{
    Screen.Navigate(new CompanyDetailsPage(_user));
    currentStateLabel.Content = "Current: Details";
}

private void SelfPB_Click(object sender, RoutedEventArgs e)
{
    Screen.Navigate(new CompanyPersonalPage(_user));
    currentStateLabel.Content = "Current: Client";
}

private void CaleClick(object sender, RoutedEventArgs e)
{
    Screen.Navigate(new CompanyEventsPage(_user));
    currentStateLabel.Content = "Current: Calendar";
}
}
}

```

LoginW.xaml

```

<Window x:Class="Visi0n._0.LoginW"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:Visi0n._0"
    mc:Ignorable="d"
    Title="Visi0n" Height="450" Width="600"
    AllowsTransparency="True" WindowStyle="None" MouseLeftButtonDown="Drag"
    Background="Transparent" BorderBrush="Transparent" Foreground="Transparent">

    <Border CornerRadius="10">
        <Border.Background>
            <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                <GradientStop Color="#FF2D2A3F"/>
                <GradientStop Color="#FF151320" Offset="1"/>
            </LinearGradientBrush>
        </Border.Background>
    </Border>

```

```

        </LinearGradientBrush>
    </Border.Background>
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="30*" />
            <RowDefinition Height="400*" />
        </Grid.RowDefinitions>

        <Grid Margin="12,0">
            <Grid.RowDefinitions>
                <RowDefinition Height="1*" />
            </Grid.RowDefinitions>

            <StackPanel x:Name="TopMenu" Orientation="Horizontal"
                FlowDirection="RightToLeft">
                <Button x:Name="Quit" Content="X" Width="50" Click="Quit_Click"
                    Foreground="AliceBlue" Background="Transparent" BorderBrush="Transparent"
                    BorderThickness="0" />
                <Button x:Name="Mini" Content="___" Width="50" Click="Mini_Click"
                    Foreground="AliceBlue" Background="Transparent" BorderBrush="Transparent"
                    BorderThickness="0" />
                <Button x:Name="Maxi" Content="[ ]" Width="50" Click="Maxi_Click"
                    Foreground="AliceBlue" Background="Transparent" BorderBrush="Transparent"
                    BorderThickness="0" Visibility="Collapsed" />
                <Button x:Name="Info" Content="i" Width="50" Click="Info_Click"
                    Foreground="AliceBlue" Background="Transparent" BorderBrush="Transparent"
                    BorderThickness="0" Visibility="Collapsed" />
            </StackPanel>
        </Grid>
        <Grid Margin="10" Grid.Row="1">
            <Grid.RowDefinitions>
                <RowDefinition Height="1*" />
            </Grid.RowDefinitions>

            <Frame x:Name="FrameL" NavigationUIVisibility="Hidden" Grid.RowSpan="2" />
        </Grid>
    </Grid>
</Border>

</Window>

```

LoginW.xaml.cs

using System;

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Visi0n._0.Pages;

namespace Visi0n._0
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class LoginW : Window
    {
        public LoginW()
        {
            InitializeComponent();
            FrameL.Navigate(new StartPage(FrameL, this));
            //MessageBox.Show("Click the i button for info and instructions");
        }

        private void Quit_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }

        private void Drag(object sender, RoutedEventArgs e)
        {
            DragMove();
        }

        private void Mini_Click(object sender, RoutedEventArgs e)
        {
            this.WindowState = WindowState.Minimized;
        }
    }
}

```



```
private void Maxi_Click(object sender, RoutedEventArgs e)
{
    if(this.WindowState != WindowState.Maximized)
        WindowState = WindowState.Maximized;
    else
        WindowState = WindowState.Normal;
}

private void Info_Click(object sender, RoutedEventArgs e)
{
    new InfoW().Show();
}
}
```

PersonalW.xaml

```
<Window x:Class="Visi0n._0.PersonalW"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:Visi0n._0"
    mc:Ignorable="d"
    Title="Visi0n - Personal" Height="450" Width="800"
    MinHeight="450" MinWidth="800"
    WindowStyle="None" AllowsTransparency="True" Background="Transparent"
    MouseLeftButtonDown="Drag">

    <Border x:Name="BorderMain" CornerRadius="10">
        <Border.Background>
            <LinearGradientBrush EndPoint="0.35,1" StartPoint="1,0">
                <GradientStop Color="#FFF3FDFF" Offset="0"/>
                <GradientStop Color="#FFF0F3FF" Offset="0.6"/>
                <GradientStop Color="#FFFFFF5FE" Offset="1"/>
            </LinearGradientBrush>
        </Border.Background>

        <Grid x:Name="Grid01">
            <Grid.RowDefinitions>
                <RowDefinition Height="50"/>
                <RowDefinition Height="1*"/>
            </Grid.RowDefinitions>
```

```

<Grid.ColumnDefinitions>
  <ColumnDefinition Width="120"/>
  <ColumnDefinition Width="1*"/>
</Grid.ColumnDefinitions>

<Grid x:Name="grid02" Margin="5,7,10,7">
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="1*"/>
    <ColumnDefinition Width="1*"/>
  </Grid.ColumnDefinitions>

  <Border x:Name="pfp" Background="Gray" CornerRadius="20" Height="30"
Width="30" MouseDown="pfpC">
    <Border.Clip>
      <RectangleGeometry RadiusX="20" RadiusY="20" Rect="0,0,30,30"/>
    </Border.Clip>

    <Image Source="/Core/Pictures/sea.jpg" Stretch="Fill"/>
  </Border>
  <Image x:Name="Settings" Grid.Column="1" Source="/Core/Pictures/settings.png"
Margin="2" MouseDown="Settings_MouseDown" />
</Grid>

<Frame x:Name="Frame01" Grid.Column="1" Grid.Row="1"
Navigated="Frame01_Navigated" NavigationUIVisibility="Hidden" Visibility="Collapsed"/>

<!--acc-->
<Grid x:Name="AccSec" Grid.Row="1" Grid.Column="1" Visibility="Visible">
  <Grid.RowDefinitions>
    <RowDefinition Height="35*"/>
    <RowDefinition Height="35*"/>
    <RowDefinition Height="35*"/>
    <RowDefinition Height="35*"/>
    <RowDefinition Height="300*"/>
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="25*"/>
    <ColumnDefinition Width="35*"/>
    <ColumnDefinition Width="40*"/>
  </Grid.ColumnDefinitions>
  <Label Style="{StaticResource LL01}" Name="usrName" Content="Username"
Grid.Row="0" Grid.Column="0" FontSize="14"/>

```

```

        <Label Style="{StaticResource LL01}" Name="usrPass" Content="Password"
Grid.Row="1" Grid.Column="0" FontSize="14"/>
        <Label Style="{StaticResource LL01}" Name="usrType" Content="User Type"
Grid.Row="2" Grid.Column="0" FontSize="14"/>
        <Label Style="{StaticResource LL01}" Name="usrID" Content="ID" Grid.Row="3"
Grid.Column="0" FontSize="14"/>
        <!--Label Style="{StaticResource LL01}" Content="Username" Grid.Row="0"
FontSize="14"/>
        <Label Style="{StaticResource LL01}" Content="Password" Grid.Row="1"
FontSize="14"/>
        <Label Style="{StaticResource LL01}" Content="User Type" Grid.Row="2"
FontSize="14"/>
        <Label Style="{StaticResource LL01}" Content="ID" Grid.Row="3" FontSize="14"/>
        <Label Style="{StaticResource LL01}" Name="prsnFName" Content="First Name"
Grid.Row="0" Grid.Column="1" FontSize="14"/>
        <Label Style="{StaticResource LL01}" Name="prsnLName" Content="Last Name"
Grid.Row="1" Grid.Column="1" FontSize="14"/>
        <Label Style="{StaticResource LL01}" Name="prsnCorp" Content="Company"
Grid.Row="3" Grid.Column="1" FontSize="14"/>
        <Button Style="{StaticResource BP01}" Content="Log Out" Height="50" Width="100"
Margin="5,10" HorizontalAlignment="Left" VerticalAlignment="Top" Grid.Row="7"
Click="logout_Click" Grid.ColumnSpan="5"/>
        <!--Grid Grid.Column="2" Grid.RowSpan="10">
        <Grid.RowDefinitions>
        </Grid.RowDefinitions>
        <Label x:Name="archData" Style="{StaticResource LL01}" Content="Text :
[preson]" FontSize="14"/>
        </Grid-->
    </Grid>

    <!--set-->
    <Grid x:Name="SettSec" Grid.Row="1" Grid.Column="1" Visibility="Collapsed">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
        </Grid.ColumnDefinitions>
        <RadioButton x:Name="darkLightMode" Height="25" Width="50" Margin="5,5,0,0"
VerticalAlignment="Top" HorizontalAlignment="Left" Checked="darkLightMode_Click"/>
        <Label x:Name="togglelabel" Content="light" Grid.Column="1"/>
    </Grid>

    <StackPanel x:Name="MenuP0" Grid.Column="0" Grid.Row="1" Margin="0,0,0,0">
        <RadioButton Style="{StaticResource BP02}" x:Name="HomeP" Content="Home"
Height="50" Checked="HomeP_Click"/>
    </StackPanel>

```

```

        <RadioButton Style="{StaticResource BP02}" x:Name="CalP" Content="Callendar"
Height="50" Checked="CalP_Click"/>
        <RadioButton Style="{StaticResource BP02}" x:Name="NoteP" Content="Notes"
Height="50" Checked="NoteP_Click"/>
        <RadioButton Style="{StaticResource BP02}" x:Name="RemindP"
Content="Reminders" Height="50" Checked="RemindP_Click"/>
    </StackPanel>
    <StackPanel Grid.Row="0" Grid.Column="1" Orientation="Horizontal"
FlowDirection="RightToLeft" Margin="0,0,10,0">
        <!--RadioButton Style="{StaticResource BP02}" x:Name="logout" Content="Log Out"
Width="75" Checked="logout_Click"/-->
        <RadioButton Style="{StaticResource BP02}" x:Name="quit" Content="X" Width="50"
Checked="quit_Click"/>
        <RadioButton Style="{StaticResource BP02}" x:Name="minmax" Content="[ ]"
Width="50" Checked="minmax_Checked"/>
        <RadioButton Style="{StaticResource BP02}" x:Name="fold" Content="_____"
Width="50" Checked="fold_Checked" />
    </StackPanel>
    <Label x:Name="CurrentPage" Style="{StaticResource LL02}" Content="Home"
FontSize="24" FontWeight="Bold" Grid.Column="1" Width="150" VerticalAlignment="Center"
HorizontalAlignment="Left"/>

    </Grid>
</Border>

</Window>

```

PersonalW.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using Visi0n._0.Pages;
using Visi0n._0.Pages.Pesonal;

```

```

using Visi0n._0.Pages.General;
using System.Windows.Controls.Primitives;
using Model_;
using VModel_;

namespace Visi0n._0
{
    /// <summary>
    /// Interaction logic for Window1.xaml
    /// </summary>
    public partial class PersonalW : Window
    {
        User _user;
        Person _person;

        public PersonalW(User usr = null)
        {
            InitializeComponent();
            Section("f");
            if (usr != null) UserSetUp(usr);
            Frame01.Navigate(new PersonalHome(Frame01, _person));
        }

        private void UserSetUp(User u)
        {
            _user = u;

            usrName.Content = "Username: " + _user._usrName;
            usrPass.Content = "Password: " + _user._pwd;
            usrType.Content = "User Type: " + _user._type;
            usrID.Content = "User ID: " + _user._absId;

            _person = UserService.LaPersona(_user);
            if (_person == null) MessageBox.Show("person not found");

            if (_person != null)
            {
                prsnFName.Content = "First Name: " + _person._fName;
                prsnLName.Content = "Last Name: " + _person._lName;
                prsnCorp.Content = "Company: " + UserService.UnGroupe(_person)._cName;
            }
        }
    }
}

```

```

/*if (_person != null)
{
    string ttype = "unknown";
    if (_user._type == 1 || _user._type == 3) ttype = "person";
    archData.Content = $"Person: \n{_person._usrName} ({_person._pwd}) [t : {ttype}] \n"
        + $"({_person._fName} {_person._lName}) \n"
        + $"Corp : {_person._cid} \n"
        + $"ID : {_person._absId} \n\n"
        + "User: \n" + $"({_user._usrName} ({_user._pwd}) \n"
        + "Type : " + _user._type + " \nID : " + _user._absId;
}*/
}

```

```

private void Drag(object sender, RoutedEventArgs e)
{
    try { DragMove(); }
    catch { } // if cant drag, then don't
}

```

```

private void quit_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}

```

```

private void logout_Click(object sender, RoutedEventArgs e)
{
    new LoginW().Show();
    this.Close();
}

```

```

private void Frame01_Navigated(object sender,
System.Windows.Navigation.NavigationEventArgs e) { }

```

```

private void HomeP_Click(object sender, RoutedEventArgs e)
{
    Section("f");
    Frame01.Navigate(new PersonalHome(Frame01, _person));
    CurrentPage.Content = "Home";
    HomeP.IsChecked = false;
}

```

```

        /*MessageBox.Show("Ideas to make this unique and better: \n" +
            "Create gaming (skyblock) profile page \n" +
            "Create sprort and sw page \n" +
            "Make corporate chat \n" +
            "");*/
    }
    private void CalP_Click(object sender, RoutedEventArgs e)
    {
        Section("f");
        Frame01.Navigate(new CalendarGP(Frame01, _user));
        CurrentPage.Content = "Calendar";
        CalP.IsChecked = false;
    }
    private void NoteP_Click(object sender, RoutedEventArgs e)
    {
        Section("f");
        Frame01.Navigate(new NotesGP(Frame01, _user));
        CurrentPage.Content = "Notes";
        NoteP.IsChecked = false;
    }
    private void RemindP_Click(object sender, RoutedEventArgs e)
    {
        Section("f");
        Frame01.Navigate(new RemindersGP(Frame01, _user));
        CurrentPage.Content = "Reminders";
        RemindP.IsChecked = false;
    }

    private void minmax_Checked(object sender, RoutedEventArgs e)
    {
        if (this.WindowState == WindowState.Normal) this.WindowState =
WindowState.Maximized;
        else this.WindowState = WindowState.Normal;

        minmax.IsChecked = false;
    }
    private void fold_Checked(object sender, RoutedEventArgs e)
    {
        this.WindowState = WindowState.Minimized;
        fold.IsChecked = false;
    }
}

```

```

private void pfpC(object sender, MouseButtonEventArgs e) { Section("a"); }
private void Settings_MouseDown(object sender, MouseButtonEventArgs e) { Section("s");
}

private void Section(string type__)
{
    if (type__ == "a")
    {
        Frame01.Visibility = Visibility.Collapsed;
        AccSec.Visibility = Visibility.Visible;
        SettSec.Visibility = Visibility.Collapsed;
    }
    if (type__ == "s")
    {
        Frame01.Visibility = Visibility.Collapsed;
        AccSec.Visibility = Visibility.Collapsed;
        SettSec.Visibility = Visibility.Visible;
    }
    if (type__ == "f")
    {
        Frame01.Visibility = Visibility.Visible;
        AccSec.Visibility = Visibility.Collapsed;
        SettSec.Visibility = Visibility.Collapsed;
    }
}

private void darkLightMode_Click(object sender, RoutedEventArgs e)
{
    if (togglelabel.Content == "dark (not coming soon)")
    {
        togglelabel.Content = "light";
        darkLightMode.IsChecked = false;
    }
    else
    {
        togglelabel.Content = "dark (not coming soon)";
        darkLightMode.IsChecked = false;
    }
}
}
}

```


BaseDB.cs

```
using Model_;
using System;
using System.Collections.Generic;
using System.Data.OleDb;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Markup;

namespace VModel_
{
    /// <summary>
    /// The base class for all database communication classes
    /// Is working with OleDb 8.0.0 & x86 or x64 (.16/.12 provider versions)
    /// </summary>
    public abstract class BaseDB
    {
        private static string dir = Directory.GetCurrentDirectory();
        private static string dirr =
Path.GetDirectoryPath(Path.GetDirectoryPath(Path.GetDirectoryPath(Path.GetDirectoryPath(dir))));

        public string connectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" + dirr + "\\VModel!\DB\DatabaseTest1.accdb" +
        ";Persist Security Info=True";

        // a static connection string for the service, the first one is not valid for it
        public string connestringStatic = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\User\Desktop\VisiOn\VisiOn\VModel!\DB\DatabaseTest1.accdb;Persist Security Info=True";

        public OleDbConnection connection;
        public OleDbCommand command;
        public OleDbDataReader reader;
        public System.Data.OleDb.OleDbDataReader asncReader;
        public OleDbTransaction trans;

        protected BaseDB()
        {
            // setup
        }
    }
}
```

```

        this.connection = new OleDbConnection(this.connestringStatic);
        this.command = new OleDbCommand();
        this.command.Connection = this.connection;
    }

```

```

protected void CloseSetup()
{
    if (reader != null) reader.Close();
    if (connection.State == System.Data.ConnectionState.Open)
        connection.Close();
}

```

protected abstract Entity EGen(); // Generation of highest level entity, such as (Entity)User;
in specified class
public abstract void CreateModel(Entity e); // Setup of the entity; in specified class

```

// Returns all entities, internal command
protected List<Entity> Collect(string cmdTxt)
{
    command.CommandText = cmdTxt;
    List<Entity> el = new List<Entity>();

    try
    {
        command.Connection = connection;
        connection.Open();
        reader = command.ExecuteReader();
        Entity tmp = new Entity();

        while (reader.Read()) // each new reader line - each progression
        {
            tmp = EGen();
            CreateModel(tmp);
            el.Add(tmp);
        }
    }
    catch (Exception ex)
    { Console.WriteLine(" Could not load database \n Error message: \n " + ex.Message); } //
print error if is

```

```

        finally // close
        {
            CloseSetup();
        }

        return el;
    }

    // The reflection of Collect in all specified classes, with a more self explanatory name
    public abstract List<Entity> SelectAll(string cmdTxt);

    // Returns target entity (mostly w/ Collect as well); not necessary in all classes
    public abstract Entity TargetSelect(int id, string cmdTxt);

    // The execution of Insert, Update & Remove commands
    protected async Task<int> Edit(string sqlStr = "emptyspace", int records = 0)
    {
        trans = null;
        try
        {
            //command.CommandText = sqlStr;
            //connection.Open();
            trans = connection.BeginTransaction();
            command.Transaction = trans;
            records = command.ExecuteNonQuery(); // action; + Async & await for later
            trans.Commit();
        }
        catch (Exception e)
        {
            trans.Rollback();
            Console.WriteLine("Error message: \n " + e.Message);
        }
        finally
        {
            CloseSetup();
        }
        return records;
    }

    protected void CommandSet(string sqlStr)
    {
        command.CommandText = sqlStr;
        connection.Open();
    }

```

```

    }

    public abstract Task<int> Insert(Entity e);
    public abstract Task<int> Update(Entity e0, Entity e1);
    public abstract Task<int> Remove(Entity e);
}
}

```

CaleDB.cs

```

using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Reflection.PortableExecutable;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using Model_;

namespace VModel_
{
    public class CaleDb : BaseDB
    {
        public CaleDb() : base() { }

        public ObservableCollection<Event> SelectPerId(User user, string cmdTxt = "SELECT *
FROM Cale_Tbl")
        {
            int id = user._absId;
            ObservableCollection<Event> nl = new ObservableCollection<Event>();
            List<Entity> el = SelectAll();
            foreach (Event e in el)
            { if (e._uid == id) nl.Add(e); }
            return nl;
        }

        public override List<Entity> SelectAll(string cmdTxt = "SELECT * FROM Cale_Tbl")
        {
            return base.Collect(cmdTxt);
        }

        public override Entity TargetSelect(int id, string cmdTxt = "SELECT * FROM Cale_Tbl")
        {

```

```

        List<Entity> cl = base.Collect(cmdTxt);
        foreach (Event e in cl)
        { if (e._ID == id) return e; }
        return new Event();
    }

    public override void CreateModel(Entity e)
    {
        Event n = e as Event;
        n._uid = int.Parse(reader["Uid"].ToString());
        n._name = reader["EventName"].ToString();
        n._description = reader["EventContent"].ToString();
        n._date = reader["EventDate"].ToString();
        n._ID = int.Parse(reader["ID"].ToString());
        n._cid = reader["Corp"].ToString();
    }

    protected override Entity EGen()
    {
        return new Event();
    }

    public int ReturnNextID(string cmdTxt = "SELECT * FROM Cale_Tbl")
    {
        command.CommandText = cmdTxt;
        int nMax = 0;

        try
        {
            command.Connection = connection;
            connection.Open();
            reader = command.ExecuteReader();

            while (reader.Read())
            {
                nMax++;
            }

            nMax++;
        }
        catch (Exception ex) { Console.WriteLine(" Could not load database \n Error message:
\n " + ex.Message); }
        finally

```

```

    {
        CloseSetup();
    }
    return nMax;
}

public int FindID(string ename, User u, string date)
{
    List<Entity> el = SelectAll();
    int found = -1;
    foreach (Event e in el)
    {
        if (u._absId == e._uid && ename == e._name && date == e._date) found = e._ID;
    }
    return found; // -1 return as not found
}

public Event FindEvent(string ename, User u, string date)
{
    List<Entity> el = SelectAll(); int id = FindID(ename, u, date); Event eve = new Event();
    foreach (Event e in el) { if (e._ID == id) eve = e; }
    return eve; // empty event return (_ID = -1, _uid = -1) as not found
}

public override async Task<int> Insert(Entity e)
{
    Event n;
    if (e is Event) n = e as Event;
    else return -1;
    int records = 0;

    string sqlStr = string.Format($"INSERT INTO Cale_Tbl (Uid, EventName, EventContent,
EventDate, Corp) VALUES (?, ?, ?, ?, ?);");
    CommandSet(sqlStr);
    command.Parameters.AddWithValue("?", n._uid);
    command.Parameters.AddWithValue("?", n._name);
    command.Parameters.AddWithValue("?", n._description);
    command.Parameters.AddWithValue("?", n._date);
    command.Parameters.AddWithValue("?", n._cid);
    return Edit().Result;
}

public override async Task<int> Remove(Entity e)
{

```

```

    Event n;
    if (e is Event) n = e as Event;
    else return -1;
    int records = 0;

    string sqlStr = $"DELETE FROM Cale_Tbl WHERE (Uid = ? AND EventContent = ?
AND EventName = ? AND EventDate = ?)";
    CommandSet(sqlStr);
    command.Parameters.AddWithValue("?", n._uid);
    command.Parameters.AddWithValue("?", n._description);
    command.Parameters.AddWithValue("?", n._name);
    command.Parameters.AddWithValue("?", n._date);
    return Edit().Result;
}

public override async Task<int> Update(Entity e0, Entity e1)
{
    Event n0; // old
    Event n1; // new
    if (e0 is Event && e1 is Event) { n0 = e0 as Event; n1 = e1 as Event; }
    else return -1;
    int records = 0;

    string sqlStr = $"UPDATE Cale_Tbl SET " +
        $" EventName = ?, " +
        $" EventContent = ? " +
        $" WHERE (Uid = ? AND EventContent = ? AND EventName = ? AND EventDate =
?)" ;

    CommandSet(sqlStr);
    command.Parameters.AddWithValue("?", n1._name);
    command.Parameters.AddWithValue("?", n1._description);
    command.Parameters.AddWithValue("?", n0._uid);
    command.Parameters.AddWithValue("?", n0._description);
    command.Parameters.AddWithValue("?", n0._name);
    command.Parameters.AddWithValue("?", n0._date);
    return Edit().Result;
}

public List<string> DaysActive(int id, string month, string year, int t = 0)
{
    List<string> ll = new List<string>();
    foreach(Event l in SelectAll())
    {

```

```

        if (l._uid == id)
        {
            if (t == 0)
            {
                if (l._date.Split("/")[1] == month.ToString() && l._date.Split("/")[2] ==
year.ToString())
                {
                    ll.Add(l._date.Split("/")[0]);
                }
            }
            else
            {
                if (l._date.Split("/")[1] == month.ToString() && l._date.Split("/")[2] ==
year.ToString() && (l._cid != "--" && l._cid != "0"))
                {
                    ll.Add(l._date.Split("/")[0]);
                }
            }
        }
    }
    return ll;
}
}
}

```

CorpDB.cs

```

using Model_;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace VModel_
{
    public class CorpDB : BaseDB
    {
        /*
        SELECT * FROM (T1 INNER JOIN T2 ON T2.C = T1.C) INNER JOIN (T3 LEFT JOIN T4
ON T4.C = T3.C) ON T3.C = T2.C
        SELECT * FROM Corp_Tbl INNER JOIN (Usr_Tbl INNER JOIN TypeUser_Tbl ON
Usr_Tbl.ID = TypeUser_Tbl.Id) ON Corp_Tbl.Uid = Usr_Tbl.ID
        */
    }
}

```



```

*/

public CorpDB() : base() { }

    public override List<Entity> SelectAll(string cmdTxt = "SELECT * FROM Corp_Tbl INNER
JOIN (Usr_Tbl INNER JOIN TypeUser_Tbl ON Usr_Tbl.ID = TypeUser_Tbl.Id) ON Corp_Tbl.Uid
= Usr_Tbl.ID")
    {
        return base.Collect(cmdTxt);
        //"SELECT * FROM Corp_Tbl"
    }

    public override void CreateModel(Entity e)
    {
        Corp c = e as Corp;
        c._absId = int.Parse(reader["Uid"].ToString());

        /*User u = (User)(new UserDB().TargetSelect(c._absId));
        c._usrName = u._usrName;
        c._pwd = u._pwd;
        c._type = 2;*/

        c._usrName = reader["UserName"].ToString();
        c._pwd = reader["PassCode"].ToString();
        c._type = int.Parse(reader["Type"].ToString());

        c._cid = reader["Cid"].ToString();
        c._cName = reader["CorpName"].ToString();
    }

    protected override Entity EGen()
    {
        return new Corp();
    }

    public override Entity TargetSelect(int id, string cmdTxt = "SELECT * FROM Corp_Tbl
INNER JOIN (Usr_Tbl INNER JOIN TypeUser_Tbl ON Usr_Tbl.ID = TypeUser_Tbl.Id) ON
Corp_Tbl.Uid = Usr_Tbl.ID")
    {
        List<Entity> cl = base.Collect(cmdTxt);
    }

```

```

        foreach (Corp c in cl) { if (c._absId == id) return c; }
        return new Corp();
    }

    public Corp Call(string id, string cmdTxt = "SELECT * FROM Corp_Tbl INNER JOIN
(Usr_Tbl INNER JOIN TypeUser_Tbl ON Usr_Tbl.ID = TypeUser_Tbl.Id) ON Corp_Tbl.Uid =
Usr_Tbl.ID")
    {
        List<Entity> cl = base.Collect(cmdTxt);
        foreach (Corp c in cl) { if (c._cid == id) return c; }
        return new Corp();
    }

    public override async Task<int> Insert(Entity e)
    {
        Corp corp;
        if (e is Corp) corp = e as Corp;
        else return -1;

        int records = 0;

        string sqlStr = string.Format("INSERT INTO Corp_Tbl (Uid, Cid, CorpName) "
            + "VALUES (?, ?, ?);");
        CommandSet(sqlStr);
        command.Parameters.AddWithValue("?", corp._absId);
        command.Parameters.AddWithValue("?", corp._cid);
        command.Parameters.AddWithValue("?", corp._cName);
        return Edit().Result;
    }

    public override async Task<int> Remove(Entity e) { return -16; } // empty
    public override async Task<int> Update(Entity e0, Entity e1) { return -16; } // empty
}
}

```

NoteDB.cs

```

using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Reflection.PortableExecutable;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

```

```

using Model_;

namespace VModel_
{
    public class NoteDB : BaseDB
    {
        public NoteDB() : base() { }

        public List<NoteItem> SelectPerId(User user, string cmdTxt = "SELECT * FROM
Note_Tbl")
        {
            int id = user._absId;
            List<NoteItem> nl = new List<NoteItem>();
            List<Entity> el = base.Collect(cmdTxt);
            foreach (NoteItem n in el)
            { if (n._uid == id) nl.Add(n); }
            return nl;
        }

        public override void CreateModel(Entity e)
        {
            NoteItem n = e as NoteItem;
            n._uid = int.Parse(reader["Uid"].ToString());
            n._name = reader["NoteName"].ToString();
            n._text = reader["NoteText"].ToString();
            n._cid = reader["Corp"].ToString();
        }

        public override List<Entity> SelectAll(string cmdTxt = "SELECT * FROM Note_Tbl")
        {
            return base.Collect(cmdTxt);
        }

        public override Entity TargetSelect(int id, string cmdTxt = "SELECT * FROM Note_Tbl")
        {
            return new NoteItem(); // also unused
        }

        protected override Entity EGen()
        {
            return new NoteItem();
        }
    }
}

```

```

public override async Task<int> Insert(Entity e)
{
    NoteItem n;
    if (e is NoteItem) n = e as NoteItem;
    else return -1;
    int records = 0;

    string sqlStr = string.Format($"INSERT INTO Note_Tbl (Uid, NoteName, NoteText, Corp)
VALUES (?, ?, ?, ?);");
    CommandSet(sqlStr);
    command.Parameters.AddWithValue("?", n._uid);
    command.Parameters.AddWithValue("?", n._name);
    command.Parameters.AddWithValue("?", n._text);
    command.Parameters.AddWithValue("?", n._cid);
    return Edit().Result;
}

public override async Task<int> Remove(Entity e)
{
    NoteItem n;
    if (e is NoteItem) n = e as NoteItem;
    else return -1;
    int records = 0;

    string sqlStr = "DELETE FROM Note_Tbl WHERE (Uid = ? AND NoteText = ? AND
NoteName = ?)";
    CommandSet(sqlStr);
    command.Parameters.AddWithValue("?", n._uid);
    command.Parameters.AddWithValue("?", n._text);
    command.Parameters.AddWithValue("?", n._name);
    return Edit().Result;
}

public override async Task<int> Update(Entity e0, Entity e1)
{
    NoteItem n0; NoteItem n1;
    if (e0 is NoteItem && e1 is NoteItem) { n0 = e0 as NoteItem; n1 = e1 as NoteItem; }
    else return -1;
    int records = 0;

    string sqlStr = $"UPDATE Note_Tbl SET " +
        $" NoteName = ?, " +
        $" NoteText = ? " +
        $" WHERE (Uid = ? AND NoteText = ? AND NoteName = ?)";

```

```

        CommandSet(sqlStr);
        command.Parameters.AddWithValue("?", n1._name);
        command.Parameters.AddWithValue("?", n1._text);
        command.Parameters.AddWithValue("?", n0._uid);
        command.Parameters.AddWithValue("?", n0._text);
        command.Parameters.AddWithValue("?", n0._name);
        return Edit().Result;
    }
}
}

```

PersonalDB.cs

```

using Model_;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace VModel_
{
    public class PersonalDB : BaseDB
    {
        public PersonalDB() : base() { }

        public override List<Entity> SelectAll(string cmdTxt = "SELECT * FROM Personal_Tbl
INNER JOIN (Usr_Tbl INNER JOIN TypeUser_Tbl ON Usr_Tbl.ID = TypeUser_Tbl.Id) ON
Personal_Tbl.Uid = Usr_Tbl.ID")
        {
            return base.Collect(cmdTxt);
        }

        public override void CreateModel(Entity e)
        {
            Person p = e as Person;
            p._absId = int.Parse(reader["Uid"].ToString());

            p._usrName = reader["UserName"].ToString();
            p._pwd = reader["PassCode"].ToString();
            p._type = int.Parse(reader["Type"].ToString());
        }
    }
}

```

```

        p._fName = reader["FirstName"].ToString();
        p._lName = reader["LastName"].ToString();
        p._cid = reader["Corp"].ToString();
    }

    protected override Entity EGen()
    {
        return new Person();
    }

    public override Entity TargetSelect(int id, string cmdTxt = "SELECT * FROM Personal_Tbl
INNER JOIN (Usr_Tbl INNER JOIN TypeUser_Tbl ON Usr_Tbl.ID = TypeUser_Tbl.Id) ON
Personal_Tbl.Uid = Usr_Tbl.ID")
    {
        List<Entity> cl = base.Collect(cmdTxt);
        foreach (Person p in cl) { if (p._absId == id) return p; }
        return new Person();
    }

    public override async Task<int> Insert(Entity e)
    {
        Person prsn;
        if (e is Person) prsn = e as Person;
        else return -1;

        int records = 0;

        string sqlStr = string.Format("INSERT INTO Personal_Tbl (Uid, FirstName, LastName,
Corp) "
        + "VALUES (?, ?, ?, ?);");
        CommandSet(sqlStr);
        command.Parameters.AddWithValue("?", prsn._absId);
        command.Parameters.AddWithValue("?", prsn._fName);
        command.Parameters.AddWithValue("?", prsn._lName);
        command.Parameters.AddWithValue("?", prsn._cid);
        return Edit().Result;
    }

    public override async Task<int> Remove(Entity e) { return -16; } // empty
    public override async Task<int> Update(Entity e0, Entity e1) { return -16; } // empty
}
}

```

ReminderDB.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection.PortableExecutable;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using Model_;

namespace VModel_
{
    public class ReminderDB : BaseDB
    {
        public ReminderDB() : base() { }

        public List<Reminder> SelectPerId(User user, string cmdTxt = "SELECT * FROM
Reminder_Tbl")
        {
            int id = user._absId;
            List<Reminder> rl = new List<Reminder>();
            List<Entity> el = base.Collect(cmdTxt);
            foreach (Reminder r in el)
            { if (r._uid == id) rl.Add(r); }
            return rl;
        }

        public override void CreateModel(Entity e)
        {
            Reminder r = e as Reminder;
            r._uid = int.Parse(reader["Uid"].ToString());
            r._text = reader["Content"].ToString();
            r._cid = reader["Corp"].ToString();
        }

        protected override Entity EGen()
        {
            return new Reminder();
        }

        public override List<Entity> SelectAll(string cmdTxt = "SELECT * FROM Reminder_Tbl")
        {
            return base.Collect(cmdTxt);
        }
    }
}

```

```

    }

    public override Entity TargetSelect(int id, string cmdTxt = "SELECT * FROM
Reminder_Tbl")
    {
        return new Reminder(); // unused
    }

    public override async Task<int> Insert(Entity e)
    {
        Reminder r;
        if (e is Reminder) r = e as Reminder;
        else return -1;
        int records = 0;

        // later fix sql injection (fixed)
        string sqlStr = string.Format("INSERT INTO Reminder_Tbl (Uid, Content, Corp) " +
"VALUES (?, ?, ?);");
        CommandSet(sqlStr);
        command.Parameters.AddWithValue("?", r._uid);
        command.Parameters.AddWithValue("?", r._text);
        command.Parameters.AddWithValue("?", r._cid);
        return Edit().Result;
    }

    public override async Task<int> Remove(Entity e)
    {
        Reminder r;
        if (e is Reminder) r = e as Reminder;
        else return -1;
        int records = 0;

        // deletes all same instances (it is now a feature)
        string sqlStr = $"DELETE FROM Reminder_Tbl WHERE (Uid = ? AND Content = ?)";
        CommandSet(sqlStr);
        command.Parameters.AddWithValue("?", r._uid);
        command.Parameters.AddWithValue("?", r._text);
        return Edit().Result;
    }

    public override async Task<int> Update(Entity e0, Entity e1) { return -16; } // empty
}
}

```


UserDB.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics.Metrics;
using System.Linq;
using System.Net;
using System.Reflection.PortableExecutable;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using Model_;

namespace VModel_
{
    public class UserDB : BaseDB
    {
        //"SELECT * FROM Usr_Tbl"

        public UserDB() : base() { }

        protected override Entity EGen()
        {
            return new User();
        }

        public override Entity TargetSelect(int id, string cmdTxt = "SELECT *, Usr_Tbl.ID AS id
FROM (Usr_Tbl INNER JOIN TypeUser_Tbl ON Usr_Tbl.ID = TypeUser_Tbl.Id)")
        {
            List<Entity> cl = base.Collect(cmdTxt);
            foreach (User u in cl)
            { if (u._absId == id) return u; }
            return new User();
        }

        public override void CreateModel(Entity e)
        {
            User u = (User)e;
            u._usrName = reader["UserName"].ToString();
            u._pwd = reader["PassCode"].ToString();
            u._absId = int.Parse(reader["ID"].ToString());
            u._type = int.Parse(reader["Type"].ToString());
        }
    }
}
```

```

    public override List<Entity> SelectAll(string cmdTxt = "SELECT *, Usr_Tbl.ID AS id FROM
(Usr_Tbl INNER JOIN TypeUser_Tbl ON Usr_Tbl.ID = TypeUser_Tbl.Id)")
    {
        return base.Collect(cmdTxt);
    }

    public override async Task<int> Insert(Entity e)
    {
        User usr;
        if (e is User) usr = e as User;
        else return -1;

        User uLast = (User)SelectAll().Last(); // not perfect
        int lid = uLast._absId + 1;

        int records = 0; // affected lines

        string sqlStr = string.Format("INSERT INTO Usr_Tbl (UserName, PassCode) "
            + "VALUES (?, ?);");
        //do not insert autonumber, and do not use 'Password' fields (syntax error)
        CommandSet(sqlStr);
        command.Parameters.AddWithValue("?", usr._usrName);
        command.Parameters.AddWithValue("?", usr._pwd);
        await Edit();

        sqlStr = $"INSERT INTO TypeUser_Tbl (ID, Type) "
            + $"VALUES ({lid}, {usr._type});";
        CommandSet(sqlStr);
        //command.Parameters.AddWithValue("?", lid);
        //command.Parameters.AddWithValue("?", usr._type); // isnt working
        return Edit().Result;
    }

    public override async Task<int> Remove(Entity e) { return -16; } // empty
    public override async Task<int> Update(Entity e0, Entity e1) { return -16; } // empty

    public int FindID(string uname, string cmdTxt = "SELECT *, Usr_Tbl.ID AS id FROM
(Usr_Tbl INNER JOIN TypeUser_Tbl ON Usr_Tbl.ID = TypeUser_Tbl.Id)")
    {
        List<Entity> ul = SelectAll();
    }

```

```

        int found = -1;
        foreach (User uu in ul)
        {
            if (uu._usrName == uname) found = uu._absId;
        }

        return found;
    }
}

```

EventService.cs

```

using Model_;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static System.Runtime.InteropServices.JavaScript.JSType;
using System.Xml.Linq;

namespace VModel_
{
    public class EventService
    {
        public static ObservableCollection<Event> Load(User usr)
        {
            return new CaleDb().SelectPerId(usr);
        }

        public static Event GetEvent(ObservableCollection<Event> l, int id)
        {
            for (int i = 0; i < l.Count; i++)
            {
                if (l[i]._ID == id) return l[i];
            }
            return null;
        }

        public static ObservableCollection<Event> Storm(ObservableCollection<Event> l, string d)
        // an unfitting name ik, but I like it (returns all event._date == d for user)
        {

```

```

ObservableCollection<Event> strm = new ObservableCollection<Event>();
foreach (Event ev in l)
{
    if (ev._date == d) strm.Add(ev);
}
return strm;
}
public static ObservableCollection<Event> Vortex(User u, string d) // the upgrade of storm;
d is unspaced date
{
    ObservableCollection<Event> events = new CaleDb().SelectPerId(u);
    ObservableCollection<Event> ell = new ObservableCollection<Event>();
    foreach (Event e in events)
    {
        if (e._date == d) ell.Add(e);
    }
    return ell;
}

public static int CreateNewID()
{
    return new CaleDb().ReturnNextID();
}

public static Event Find(string name, User u, string date) => new
CaleDb().FindEvent(name, u, date);

public static void Tear(Event ev)
{
    new CaleDb().Remove(ev);
}
public static void Write(Event ev)
{
    new CaleDb().Insert(ev);
}
public static void ReWrite(Event Old, Event New)
{
    new CaleDb().Update(Old, New);
}

public static List<string> ActiveDays(int id, string month, string year) => new
CaleDb().DaysActive(id, month, year, 0);

```

```

    public static List<string> SuperActiveDays(int id, string month, string year) => new
    CaleDb().DaysActive(id, month, year, 1);
    public delegate List<string> ActivityDays(int id, string month, string year);

    // full writing mechanism, command template function
    public static void WriteEclipse(User u, Event target, string Name_, string Text_, string
    _date_, string cid = "--")
    {
        if (target == null) // add new
        {
            target = new Event();
            target._cid = cid; // its already "--", but per corp write it sets it accordingly (only for new
            events ofc)

            // set info
            target._name = Name_;
            target._description = Text_;

            //target._ID = CreateNewID(); // not needed, id is autonumber

            target._date = _date_.Replace(" ", ""); // in case of
            target._uid = u._absId; // set id to user

            Write(target); // call insert
        }
        else // edit selected
        {
            Event tor = target.Copy(); // all properties besides main info set

            target._name = Name_;
            target._description = Text_;
            // no need to change anything else, we have add + delete for that

            Rewrite(tor, target); // call update
        }
    }
}

```

NoteService.cs

```

using Model_;
using System;

```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace VModel_
{
    public class NoteService
    {
        public static List<NoteItem> GetNotes(User user)
        {
            List<NoteItem> list = new NoteDB().SelectPerId(user);
            return list;
        }

        public static void WriteNote(NoteItem n)
        {
            new NoteDB().Insert(n);
        }

        public static void DeleteNote(NoteItem n)
        {
            new NoteDB().Remove(n);
        }

        public static void UpdateNote(NoteItem n, string name, string txt)
        {
            new NoteDB().Update(n, new NoteItem(name, txt, n._uid));
        }

        public static NoteItem Find(User u, string name)
        {
            List<Entity> nl = new NoteDB().SelectAll();
            NoteItem nn = new NoteItem();
            foreach (NoteItem n in nl) { if (n._uid == u._absId && n._name == name) nn = n; }
            return nn;
        }

        public static void ScrapWrite(string Name, string Text, int Uid, string Cid = "--")
        {
            NoteItem nt = new NoteItem(Name, Text, Uid, Cid);
            WriteNote(nt);
        }
    }
}

```

```
}
```

ReminderService.cs

```
using Model_;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace VModel_
{
    public class ReminderService
    {
        public static List<Reminder> GetReminders(User user)
        {
            List<Reminder> l = new ReminderDB().SelectPerId(user);
            return l;
        }

        public static void ListReminder(Reminder r)
        {
            new ReminderDB().Insert(r);
        }

        public static void DropReminder(Reminder r)
        {
            new ReminderDB().Remove(r);
        }

        public static void ListNewReminder(int id, string text, string cid = "--") => ListReminder(new
Reminder(id, text, cid));
    }
}
```

UserService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```

using System.Data.OleDb;
using Model_;

namespace VModel_
{
    public class UserService
    {
        // general login control on base form
        public static bool LoginAtt(string username, string pass, int type_ = 1)
        {
            User usr = FindUser(username);

            if (usr._usrName == username && usr._pwd == pass && (usr._type == type_ || usr._type
== 3)) return true;
            return false;
        }

        public static User FindUser(string usrName) => (User)(new UserDB().TargetSelect(new
UserDB().FindID(usrName)));

        public static int FindUserID(string usrName) => FindUser(usrName)._absId;

        public static User Get(int id = 2) // get user with id
        {
            UserDB usrdb = new UserDB();
            User usr = (User)(usrdb.TargetSelect(id));
            if (usr._absId == id) return usr;
            else return null;
        }

        public static User MakeUser(string usrN, string pass, int ty)
        {
            User uu = new User(usrN, pass, NextId(), ty);

            new UserDB().Insert(uu);

            return uu;
        }

        public static int NextId()
        {
            List<Entity> ul = new UserDB().SelectAll();
            int iid = 0;
            foreach (User u in ul)

```



```
{
    if (iid <= u._absId) iid = u._absId;
}
iid++;
return iid;
}
```

```
public static bool Verify(User u)
{
    if (u._absId > 0) return true; else return false;
}
```

```
public static Person Persona(User u) => (Person)(new
PersonalDB().TargetSelect(u._absId)); // get (person)u raw
```

```
public static Corp LaEmpressa(User u) => (Corp)(new CorpDB().TargetSelect(u._absId)); //
get (corp)u raw
```

```
public static Person LaPersona(User u, Person p = null) // returns the full (besides type)
person of user if valid | (Person)u
{
    Person pu = Persona(u);
    if (Verify(pu)) return pu;
    else return p;
}
```

```
public static Corp Corporative(User u, Corp cc = null) // returns if the user is corp the
corporation | (Corp)u
{
    Corp cu = LaEmpressa(u);
    if (Verify(cu)) return cu;
    else return cc;
}
```

```
public static Corp UnGroupe(Person p) => new CorpDB().Call(p._cid); // returns the corp
containing user if is
```

```
public static List<Person> LaCampanella(Corp c) // returns all people in a corp
```

```
{
    List<Entity> el = new PersonalDB().SelectAll();
    List<Person> people = new List<Person>();
    foreach (Person p in el) { if (p._cid == c._cid) people.Add(p); }
    return people;
}
```

```
public static string CorpNameID(string cid) => new CorpDB().Call(cid)._cName; // get name
per id
```

```
public static void CreatePerson(User u, string f, string l, string c)
{
    Person p = new Person(u, f, l, c);
    new PersonalDB().Insert(p);
}

public static void CreateCorp(User u, string cid, string cn)
{
    Corp cc = new Corp(u, cid, cn);
    new CorpDB().Insert(cc);
}
}
```

appsettings.json (VWService)

```
{
  "Urls": "http://localhost:5000;https://localhost:5001",
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

Host.cs

```
using CoreWCF;
```

```

using CoreWCF.Configuration;
using CoreWCF.Description;
//using CoreWCFDemoServer;

namespace VWS
{
    public class Host
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            // Add WSDL support
            builder.Services.AddServiceModelServices().AddServiceModelMetadata();
            builder.Services.AddSingleton<IServiceBehavior,
            UseRequestHeadersForMetadataAddressBehavior>();

            var app = builder.Build();

            // Configure an explicit none credential type for WSHttpBinding as it defaults to Windows
            which requires extra configuration in ASP.NET
            var myWSHttpBinding = new WSHttpBinding(SecurityMode.Transport);
            myWSHttpBinding.Security.Transport.ClientCredentialType =
            HttpClientCredentialType.None;

            app.UseServiceModel(builder =>
            {
                builder.AddService<VisionService>((serviceOptions) => { })
                // Add a BasicHttpBinding at a specific endpoint
                .AddServiceEndpoint<VisionService, IVisionService>(new BasicHttpBinding(),
            "/VisionService/basichttp")
                // Add a WSHttpBinding with Transport Security for TLS
                .AddServiceEndpoint<VisionService, IVisionService>(myWSHttpBinding,
            "/VisionService/WSHttps");
            });

            var serviceMetadataBehavior =
            app.Services.GetRequiredService<CoreWCF.Description.ServiceMetadataBehavior>();
            serviceMetadataBehavior.HttpGetEnabled = true;

            app.Run();
        }
    }
}

```

IVisionService.cs

```
using CoreWCF;
using Model_;

namespace VWS
{
    [ServiceContract]
    public interface IVisionService
    {
        [OperationContract]
        string Status();

        [OperationContract]
        bool Login(string Uname, string Upass, int T);

        [OperationContract]
        void CreatePerson(string cid, string fName, string lName, string un, string pd, int id);

        [OperationContract]
        List<string> RemindersString(int id);

        [OperationContract]
        List<Reminder> Reminders(int id);

        [OperationContract]
        int GetID(string n);

        [OperationContract]
        List<string> EventsString(int id);

        [OperationContract]
        List<Event> Events(int id);

        [OperationContract]
        List<string> NotesString(int id);

        [OperationContract]
        List<NoteItem> Notes(int id);

        [OperationContract]
        List<Person> CorpUsers(int id);

        [OperationContract]
        List<string> CorpUsersString(int id);
    }
}
```

```
}
}
```

VisionService.cs

```
using CoreWCF;
using VModel_;
using Model_;

namespace VWS
{
    public class VisionService : IVisionService
    {
        public string Status() => "Running";

        public bool Login(string Un, string Up, int T) => UserService.LoginAtt(Un, Up, T);

        public int GetID(string uName) => UserService.FindUserID(uName);

        public void CreatePerson(string cid, string fName, string lName, string un, string pd, int id = 0)
        {
            Person p = new Person(cid, fName, lName, un, pd, id);
            User u = (User)p;
            User uu = UserService.MakeUser(u._usrName, u._pwd, 1);
            UserService.CreatePerson(uu, p._fName, p._lName, "-");
        }

        public List<string> RemindersString(int id)
        {
            List<Reminder> l = Reminders(id);
            List<string> result = new List<string>();
            foreach (Reminder r in l) { result.Add(r._text); }
            return result;
        }

        public List<Reminder> Reminders(int id)
        {
            User u = UserService.Get(id);
            List<Reminder> l = ReminderService.GetReminders(u);
            return l;
        }

        public List<Event> Events(int id)
```

```

{
    User u = UserService.Get(id);
    var ll = EventService.Load(u);
    List<Event> l = new();
    foreach (var i in ll) { l.Add(i); }
    return l;
}

public List<string> EventsString(int id)
{
    var lst = Events(id);
    List<string> result = new();
    foreach (var i in lst) { result.Add(i._name); }
    return result;
}

public List<NoteItem> Notes(int id)
{
    User u = UserService.Get(id);
    var ll = NoteService.GetNotes(u);
    List<NoteItem> l = new();
    foreach (var i in ll) { l.Add(i); }
    return l;
}

public List<string> NotesString(int id)
{
    var lst = Notes(id);
    List<string> result = new();
    foreach (var i in lst) { result.Add(i._name); }
    return result;
}

public List<Person> CorpUsers(int id)
{
    User u = UserService.Get(id);
    Corp c = UserService.Corporative(u);
    var l = UserService.LaCampanella(c);
    return l;
}

public List<string> CorpUsersString(int id)
{
    var lst = CorpUsers(id);

```

```

        List<string> result = new();
        foreach(var i in lst) { result.Add($"({i._fName} {i._lName}, {i._absId})"); }
        return result;
    }
}

```

launchSettings.json (VisiOnStarlight)

```

{
  "$schema": "http://json.schemastore.org/launchsettings.json",
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:35808",
      "sslPort": 44359
    }
  },
  "profiles": {
    "http": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "applicationUrl": "http://localhost:5251",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "https": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "applicationUrl": "https://localhost:7777;http://localhost:5251",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}

```

```

    }
  }
}

```

app.css

```

html, body {
  font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif;
}

a, .btn-link {
  color: #006bb7;
}

.btn-primary {
  color: #fff;
  background-color: #1b6ec2;
  border-color: #1861ac;
}

.btn:focus, .btn:active:focus, .btn-link.nav-link:focus, .form-control:focus, .form-check-input:focus
{
  box-shadow: 0 0 0 0.1rem white, 0 0 0 0.25rem #258cfb;
}

.content {
  padding-top: 1.1rem;
}

h1:focus {
  outline: none;
}

.valid.modified:not([type=checkbox]) {
  outline: 1px solid #26b050;
}

.invalid {
  outline: 1px solid #e50000;
}

.validation-message {
  color: #e50000;
}

```



```
.blazor-error-boundary {  
    background:  
  
url(data:image/svg+xml;base64,PHN2ZyB3aWR0aD0iNTYlIGhlaWdodD0iNDkiIHhtbG5zPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwL3N2ZylgeG1sbmM6eGxpbnMs9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkveGxpbnmsilG92ZXJmbG93PSJoaWRkZW4iPjxkZWZzPjxbGlwUGF0aCBpZD0iY2xpcDAiPjxyZWN0IHg9IjlzNSlgeT0iNTEiIHdpZHRoPSI1NiIGA GVpZ2h0PSI0OSlvPjwwY2xpcFBhdGg+PC9kZWZzPjxnIGNsaXA t cGF0aD0idXJsKCNjbGlwMCkilHRYYW5zZm9ybT0idHJhbnNsYXRiKC0yMzMzUgLTUxKSIPHBhdGggZD0iTTI2My41MDYgNTFDMjY0Ljc xNyA1MSAyNjUuODEzIDUxLjQ4Mzc gMjY2LjYwNiA1Mi4yNjU4TDI2Ny4wNTIgNTluNzk4NyAyNjcuNTM5IDUzLjYyODMgMjk wLjE4NSA5Mi4xODMxIDI5MC41NDUgOTluNzk1IDI5MC42NTYgOTluOTk2QzI5MC44Nzc gOTMuNTEzIDI5MSA5NC4wODE1IDI5MSA5NC42Nzg yIDI5MSA5Ny4wNjUxIDI4OS4wMzg gOTkgMjY2LjYxNyA5OUwyND AuMzg zIdk5QzIzNy45NjMgOTkgMjY2IDk3LjA2NTEgMjY2IDk0LjY3ODIgMjY2IDk0LjM3OTkgMjY2LjAzMSA5NC4wODg2IDlzNi4wODkgOTMuODA3Mkw yMzYuMz M4IDkzLjAxNjlgMjY2Ljg1OCA5Mi4xMzE0IDI1OS40NzMgNTMuNjI5NCAyNTkuOTYxIDUyLjY3ODUgMjYwLjQwNyA1Mi4yNjU4QzI2MS4yIDUxLjQ4Mzc gMjYyLjI5NiA1MSAyNjMuNTA2IDUxW k0yNjMuNTg2IDY2LjAxODNDMjYwLjczNyA2Ni4wMTgzIDI1OS4zMTMgNjcuMTI0NSAy NTkuMz EzIDY5LjMzNyAyNTkuMzEzIDY5LjYxMDIgMjY5LjMzMiA2OS44NjA4IDI1OS4zNzEgNzAuMDg4N0wyNjEuNzk1IDg0LjAxNjEgMjY1LjM4IDg0LjAxNjEgMjY3LjgyMSA2OS43NDc1QzI2Ny44NiA2OS43MzA5IDI2Ny44NzkgNjkuNTg3NyAyNjcuODc5IDY5LjMxNzkgMjY3Ljg3OSA2Ny4xMTg yID I2Ni40NDggNjYuMDE4MyAyNjMuNTg2IDY2LjAxODNaTTI2My41NzYgODYuMDU0N0MyNjEuMDQ5IDg2LjA1NDcgMjY5LjC4NiA4Ny4zM DA1IDI1OS43ODYgODkuNzkyMSAyNTkuNzg2IDkyLjI4Mzc gMjYxLjA0OSA5My41Mjk1IDI2My41NzYgOTMuNTI5NSAyNjYuMTE2IDkzLjUyOTUgMjY3LjM4NyA5Mi4yODM3IDI2Ny4zODcgODkuNzkyMSAyNjcuMzg3IDg3LjMwMDUgMjY2LjExNiA4Ni4wNTQ3IDI2My41NzYgODYuMDU0N1oiIGZpbGw9InGRKU1MDAiIGZpbGwtcnVsZT0iZXZlbm9kZCIvPjwwZz48L3N2Zz4=) no-repeat 1rem/1.8rem, #b32121;  
    padding: 1rem 1rem 1rem 3.7rem;  
    color: white;  
}  
  
.blazor-error-boundary::after {  
    content: "An error has occurred."  
}  
  
.darker-border-checkbox.form-check-input {  
    border-color: #929292;  
}
```

MainLayout.razor

@inherits LayoutComponentBase

```
<div class="page">
  <div class="sidebar">
```

```

        <NavMenu />
    </div>

    <main>
        <div class="top-row">
            <!--a href="https://learn.microsoft.com/aspnet/core/" target="_blank">About</a-->
            <p class="text-black-50">by Daniel Sermyagin</p>
        </div>

        <article class="content px-4">
            @Body
        </article>
    </main>
</div>

<div id="blazor-error-ui">
    An unhandled error has occurred.
    <a href="" class="reload">Reload</a>
    <a class="dismiss">□</a>
</div>

```

NavMenu.razor

```

<div class="top-row ps-3 navbar navbar-dark">
    <div class="container-fluid">
        <a class="navbar-brand" href="">Visi0n Starlight</a>
    </div>
</div>

<input type="checkbox" title="Navigation menu" class="navbar-toggler" />

<div class="nav-scrollable" onclick="document.querySelector('.navbar-toggler').click()">
    <nav class="flex-column">
        <div class="nav-item px-3">
            <NavLink class="nav-link" href="" Match="NavLinkMatch.All">
                <span class="bi bi-house-door-fill-nav-menu" aria-hidden="true"></span> Home
            </NavLink>
        </div>

        <!--div class="nav-item px-3">
            <NavLink class="nav-link" href="counter">
                <span class="bi bi-list-nested-nav-menu" aria-hidden="true"></span> Counter
            </NavLink>
        </div-->
    </nav>
</div>

```

```

<div class="nav-item px-3">
  <NavLink class="nav-link" href="page/1">
    <span class="bi bi-list-nested-nav-menu" aria-hidden="true"></span> Personal
  </NavLink>
</div>

<div class="nav-item px-3">
  <NavLink class="nav-link" href="page/2">
    <span class="bi bi-list-nested-nav-menu" aria-hidden="true"></span> Company
  </NavLink>
</div>

<div class="nav-item px-3">
  <NavLink class="nav-link" href="create">
    <span class="bi bi-plus-square-fill-nav-menu" aria-hidden="true"></span> Create
  </NavLink>
</div>
</nav>
</div>

```

Company.razor

```

@page "/corp/{Value}"
@rendermode InteractiveServer
@using SRv1;

```

```

<h2>Company</h2>
<p> </p> <p> </p>
<h4>Linked Users</h4>
<p>@users</p>

```

```

@code {
  [Parameter]
  public string Value { get; set; }

  string users = "";

  int id;

  protected override async void OnInitialized()
  {

```

```

        base.OnInitialized();
        id = int.Parse(Value);
        VisionServiceClient client = new
VisionServiceClient(VisionServiceClient.EndpointConfiguration.BasicHttpBinding_IVisionService
, "http://localhost:5000/VisionService/basichttp");
        users = await LoadUsers(id, client);
        StateHasChanged();
    }

```

```

async Task<string> LoadUsers(int id, VisionServiceClient client)
{
    var strings = await client.CorpUsersStringAsync(id); string ls = "";
    foreach (var s in strings) { ls += "" + s + "; "; }
    return ls;
}
}

```

Home.razor

@page "/"

<PageTitle>Home</PageTitle>

<h1>The Web Based Visi0n User Interface</h1>

```

<style>
    .border {
        border-radius: 8px;
        max-width: 100%;
        height: auto;
    }
</style>

```

```

<ul class="list-group list-group-flush">
    <li class="list-group-item">view company members & personal details</li>
    <li class="list-group-item">generate new users (personal)</li>
    <!--li class="list-group-item">use the provided tools and other features</li-->
</ul>
<p> </p>
<p>Enjoy this picture for the while</p>


```

LoginPage.razor

```

@page "/page/{value}"
@rendermode InteractiveServer
@inject NavigationManager Navigation
@using SRv1;

<PageTitle>Login</PageTitle>

<h1>Login Page</h1>

<input @bind-value="_name" />
<input @bind-value="_text" />
<p>Checked: @_inputfull</p>
<button class="btn btn-primary" @onclick="LogAtt">Click me</button>

<p role="status">Login: @result</p>

@code {

    [Parameter]
    public string value { get; set; }

    private int currentCount = 0;
    string _name = "Username";
    string _text = "Password";
    string _inputfull = "";

    bool Submit;
    string result = "";

    private async void LogAtt()
    {
        currentCount++;
        result = "trying";
        _inputfull = _name + " " + _text;
        Submit = await Call(_name, _text, int.Parse(value));
        if (Submit)
        {
            result = "logged";
        }
        else
    }

```

```

    {
        result = "failed";
    }
    StateHasChanged();
    await Task.Delay(300);
    VisionServiceClient client = new
VisionServiceClient(VisionServiceClient.EndpointConfiguration.BasicHttpBinding_IVisionService
, "http://localhost:5000/VisionService/basichttp");
    int idd = await client.GetIDAsync(_name);
    if (Submit)
    {
        if (int.Parse(value) == 1)
        {
            Navigate("self/" + idd);
        }
        else if (int.Parse(value) == 2)
            Navigate("corp/" + idd);
    }
}

public async Task<bool> Call(string un, string pw, int tp)
{
    var client = new
VisionServiceClient(VisionServiceClient.EndpointConfiguration.BasicHttpBinding_IVisionService
, "http://localhost:5000/VisionService/basichttp");
    bool fl = await client.LoginAsync(un, pw, tp);
    if (fl) return true;
    else return false;
}

void Reload() { } // just run a command on main to load unloaded data (for async functions),
or use StateHasChanged

void Navigate(string name)
{
    Navigation.NavigateTo("/") + name);
}
}

```

Personal.razor

```

@page "/self/{Value}"
@rendermode InteractiveServer
@using SRv1;

```

```
<h2>Personal</h2>
<p> </p> <p> </p>
<h4>Calendar Events</h4>
<p>@calEves</p>
<h4>Notes Index</h4>
<p>@notes</p>
<h4>Reminders</h4>
<p>@reminders</p>
```

```
@code {
    [Parameter]
    public string Value { get; set; }

    string calEves = "";
    string notes = "";
    string reminders = "";

    int id;

    protected override async void OnInitialized()
    {
        base.OnInitialized();
        id = int.Parse(Value);
        VisionServiceClient client = new
VisionServiceClient(VisionServiceClient.EndpointConfiguration.BasicHttpBinding_IVisionService
, "http://localhost:5000/VisionService/basichttp");
        reminders = await LoadReminders(id, client);
        notes = await LoadNotes(id, client);
        calEves = await LoadEvents(id, client);
        StateHasChanged();
    }

    async Task<string> LoadReminders(int id, VisionServiceClient client)
    {
        var strings = await client.RemindersStringAsync(id); string ls = "";
        foreach(var s in strings) { ls += "" + s + "; "; }
        return ls;
    }

    async Task<string> LoadNotes(int id, VisionServiceClient client)
    {
        var strings = await client.NotesStringAsync(id); string ls = "";
```

```

        foreach (var s in strings) { ls += "" + s + "; "; }
        return ls;
    }

    async Task<string> LoadEvents(int id, VisionServiceClient client)
    {
        var strings = await client.EventsStringAsync(id); string ls = "";
        foreach (var s in strings) { ls += "" + s + "; "; }
        return ls;
    }
}

```

UserGen.razor

```

@page "/create"
@inject NavigationManager Navigation
@rendermode InteractiveServer
@using Newtonsoft.Json;
@using SRv1;

```

```

<div class="text-center">
    <h2>Random User Api for user creation</h2>
    <p class="text-black">Use it to generate new users (personal) for thr Visi0n project</p>
    <p>Or make a new user manually</p>
</div>

```

```

<div class="text-center">
    <button class="btn btn-primary" @onclick="GenUser">Generate</button>

    <p> </p>
    <p>
        <i> ( First Name, Last Name, Username, Password ) </i>
    </p>

    <input type="text" id="I1" name="FN" @bind-value="Fname" />
    <input type="text" id="I2" name="LN" @bind-value="Lname" />
    <input type="text" id="I3" name="US" @bind-value="uName" />
    <input type="text" id="I4" name="PW" @bind-value="pass" />
    <p> </p>
    <p> <i>all users created this way are not linked to any corporation</i></p>
    <p> </p>
    <button class="btn btn-primary" @onclick="Act">Register</button>

```



```
<!--button class="btn btn-primary" @onclick="Navigate">Redirect</button-->
</div>
```

```
@code {
    public string InputText;
    public string Result;

    public string uName;
    public string pass;
    public string FName;
    public string Lname;

    // register
    public async void Act()
    {
        var client = new
VisionServiceClient(VisionServiceClient.EndpointConfiguration.BasicHttpBinding_IVisionService
, "http://localhost:5000/VisionService/basichttp");
        await client.CreatePersonAsync("-", FName, Lname, uName, pass, 0);
        GenUser();
    }

    // load data into the page
    public async void GenUser()
    {
        uName = await GetUName();
        pass = await GetPCode();
        FName = await GetFName();
        Lname = await GetLName();
        StateHasChanged();
    }

    // returns a list (async) of parts of a user
    public async Task<List<string>> Create()
    {
        string apiUrl = "https://randomuser.me/api/?results=1&nat=us"; // 1 person (from us)
        string fn = "";
        string ln = "";
        string u = "";
        string p = "";
        List<string> list = new List<string>();
```

```

using (HttpClient client = new HttpClient())
{
    HttpResponseMessage response = await client.GetAsync(apiUrl);
    if (response.IsSuccessStatusCode) // if there is a result
    {
        string jsonResponse = await response.Content.ReadAsStringAsync();

        // Parse
        RootObject0j rootObject =
        JsonConvert.DeserializeObject<RootObject0j>(jsonResponse);

        // loop for the case of more than 1 person
        foreach (var result in rootObject.results)
        {
            //ILI += ("Name: {result.name.first} {result.name.last}; {result.login.username}
            {result.login.password}");
            fn += result.name.first; ln += result.name.last; u += result.login.username; p +=
            result.login.password;
            // since there is only 1 result there is no meaning in changing it
        }
        list.Add(fn); list.Add(ln); list.Add(u); list.Add(p);
    }
    else
    {
        Console.WriteLine("Error fetching data.");
    }
}

return list;
}

// individual parts, might improve
public async Task<string> GetFName()
{
    List<string> list = await Create();
    return list[0];
}
public async Task<string> GetLName()
{
    List<string> list = await Create();
    return list[1];
}
public async Task<string> GetUName()

```

```

    {
        List<string> list = await Create();
        return list[2];
    }
    public async Task<string> GetPCode()
    {
        List<string> list = await Create();
        return list[3];
    }
    // result object model
    public class RootObject0j
    {
        public List<Result0j> results { get; set; }
    }
    public class Result0j
    {
        public Name0j name { get; set; }
        public string email { get; set; }
        public Login0j login { get; set; }
    }
    public class Name0j
    {
        public string title { get; set; }
        public string first { get; set; }
        public string last { get; set; }
    }
    public class Login0j
    {
        public string username { get; set; }
        public string password { get; set; }
    }

    void Navigate()
    {
        Navigation.NavigateTo("/counter");
    }

    void Ref() { }
}

<!--button @onclick="Ref">Refresh<!--button-->

```