

Peace of mind

פרויקט גמר סדנא בתכנות מונחה עצמים 20586



****

המנחה דני כלפון – סמסטר 2024 ב

האונברסיטה הפתוחה

עטי גינבורג ואילה שימל

Contents

[מסמך רקע ואפיון עבור מערכת לניהול קליניקה פרטית Peace of Mind 5](#_Toc175262458)

[מבוא 5](#_Toc175262459)

[דרישות המערכתהמערכת תהיה זמינה למטפל – מנהל הקלניקה הפרטית באמצעות תוכנה שולחנית. 6](#_Toc175262460)

[פונקצינליות עיקריות 6](#_Toc175262461)

[התחברות למערכת 6](#_Toc175262462)

[פגישות טיפוליות 6](#_Toc175262463)

[לקוחות: 6](#_Toc175262464)

[גורמים מממנים: 7](#_Toc175262465)

[תשלומים 7](#_Toc175262466)

[אלמנטים במערכת 7](#_Toc175262467)

[מנהל הקליניקה – מטפל ראשי 7](#_Toc175262468)

[דרישות לתשלום 7](#_Toc175262469)

[גורם מממן 8](#_Toc175262470)

[לקוחות 8](#_Toc175262471)

[דיאגרמות 9](#_Toc175262472)

[דיאגרמת התחברות ראשונית למערכת 9](#_Toc175262473)

[פגישות 10](#_Toc175262474)

[לקוחות 11](#_Toc175262475)

[גורמים מממנים 12](#_Toc175262476)

[דרישה לתשלום 13](#_Toc175262477)

[מסמך תכנון ואפיון 14](#_Toc175262478)

[1. תאור כללי של המערכת 14](#_Toc175262479)

[1.1 יתרונות הגישהMVVM 14](#_Toc175262480)

[1.2 דגשים והנחיות בפיתוח התוכנית 14](#_Toc175262481)

[2.1 מוסכמות רישום 15](#_Toc175262482)

[3. שכבת בסיס הנתונים (namespce - loginDb.Repositories ) 15](#_Toc175262483)

[3.1 תאור המחלקות המשמשות להתקשרות 15](#_Toc175262484)

[3.2 התחברות למסד הנתונים EntityFramwork 16](#_Toc175262485)

[3.3 תבניות עיצוב המשמשות להתחברות אל מסד הנתונים 16](#_Toc175262486)

[3.4 תאור מסד הנתונים: 18](#_Toc175262487)

[3.4.1 קשרים לוגיים בין אוביקטים 18](#_Toc175262488)

[3.4.2 טבלת חשבון משתמש UserAccount 18](#_Toc175262489)

[3.4.3 טבלת מטפל User 19](#_Toc175262490)

[3.4.4 טבלת לקוח Client 19](#_Toc175262491)

[3.4.5 טבלת גורמים ממממנים Payers 19](#_Toc175262492)

[3.4.6 טבלת תשלומים Payment 20](#_Toc175262493)

[3.4.6 טבלת פגישות Meetings 20](#_Toc175262494)

[3.5 דיאגרמת מסד הנתונים 21](#_Toc175262495)

[4. שכבת הנתונים 22](#_Toc175262496)

[4.1 מחלקת user 22](#_Toc175262497)

[4.2 מחלקת user account 22](#_Toc175262498)

[4.3 מחלקת client 23](#_Toc175262499)

[4..4 מחלקת payers 23](#_Toc175262500)

[4..5 מחלקת Meeting 24](#_Toc175262501)

[.4.6 מחלקת payment 24](#_Toc175262502)

[4.7 Status.cs 25](#_Toc175262503)

[5. מחלקות עזר convectors –loginDb.Converters 26](#_Toc175262504)

[6. שכבת הלוגיקה view model 28](#_Toc175262505)

[6.1 ViewModels/ViewModelBase.cs 28](#_Toc175262506)

[6.2 ViewModels/ViewModelCommand.cs 28](#_Toc175262507)

[6.3 ViewModels/LoginViewModel.cs 29](#_Toc175262508)

[6.4 ViewModels/HomeViewModel.cs 30](#_Toc175262509)

[6.5 ViewModels/MainViewModel.cs 30](#_Toc175262510)

[6.6 ViewModels/ReportsViewModel.cs 31](#_Toc175262511)

[6.7 ViewModels/ClientsViewModel.cs 32](#_Toc175262512)

[6.8 ViewModels/MeetingsViewModel.cs 32](#_Toc175262513)

[6.9 ViewModels/PayersViewModel.cs 33](#_Toc175262514)

[6.10 ViewModels/PaymentsViewModel.cs 34](#_Toc175262515)

[6.11 ViewModels/AddOrEditClientViewModel.cs 35](#_Toc175262516)

[6.12 ViewModels/AddOrEditPaymentViewModel.cs 36](#_Toc175262517)

[6.13 ViewModels/AddOrEditPayerViewModel.cs 37](#_Toc175262518)

[6.14 ViewModels/AddOrEditMeetingViewModel.cs 37](#_Toc175262519)

[7. שכבת התצוגה 39](#_Toc175262520)

[7.1 Custom Control 40](#_Toc175262521)

[7.2 Styles 40](#_Toc175262522)

[7.3 טפסים במערכת 40](#_Toc175262523)

[7.3.1 View/LoginView.xaml.cs 40](#_Toc175262524)

[7.3.2 View/HomeView.xaml.cs 41](#_Toc175262525)

[7.3.3 View/MainView.xaml.cs 41](#_Toc175262526)

[7.3.4 View/ClientsView.xaml.cs 41](#_Toc175262527)

[7.3.5 View/MeetingsView.xaml.cs 42](#_Toc175262528)

[7.3.6 View/PayersView.xaml.cs 42](#_Toc175262529)

[7.3.7 View/PaymentsView.xaml.cs 43](#_Toc175262530)

[7.3.8 View/ReportsView.xaml.cs 43](#_Toc175262531)

[7.3.9 View/AddOrEditClientView.xaml 44](#_Toc175262532)

[7.3.10 View/AddOrEditMeetingView.xaml.cs 44](#_Toc175262533)

[7.3.11 View/AddOrEditPayerView.xaml.cs 44](#_Toc175262534)

[7.3.12 View/AddOrEditPaymentView.xaml.cs 45](#_Toc175262535)

[8. דיאגרמת ישויות קשרים 46](#_Toc175262536)

[9. שינויים עתידיים בתוכנה 47](#_Toc175262537)

[9.1 עדכון תמונת פרופיל יחודית לכל משתמש 47](#_Toc175262538)

[9.2 הוספת מטפלים ואנשי צוות נוספים 47](#_Toc175262539)

[9.3 עדכון תעריף טיפול 47](#_Toc175262540)

[10. מדריך למשתמש 48](#_Toc175262541)

[10.1 רישום משתמש חדש למערכת 49](#_Toc175262542)

[10.2 התחברות משתמש קיים למערכת 50](#_Toc175262543)

[10.3 מסך הבית של התוכנה 50](#_Toc175262544)

[10.4 תצוגת לקוחות במערכת 51](#_Toc175262545)

[10.4.1 עריכת פרטי לקוח קיים 51](#_Toc175262546)

[10.4.2 הוספת לקוח חדש למערכת 52](#_Toc175262547)

[10.5 תצוגת פגישות בקליניקה 52](#_Toc175262548)

[10.5.1 עריכת פרטי פגישה 53](#_Toc175262549)

[10.5.2 הוספת פגישה חדשה 53](#_Toc175262550)

[10.6 תצוגת גורמים מממנים 54](#_Toc175262551)

[10.6.1 עריכת פרטי גורם מממן 54](#_Toc175262552)

[10.6.2 הוספת גורם מממן חדש 55](#_Toc175262553)

[10.7 תצוגת דרישות לתשלום 56](#_Toc175262554)

[10.7.1 הוספת דרישה חדשה לתשלום 56](#_Toc175262555)

[10.8 תצוגת דו"חות 58](#_Toc175262556)

[10.8.1 סיכום כללי אודות הקליניקה 58](#_Toc175262557)

[10.8.2 דו"ח מצבת לקוחות 59](#_Toc175262558)

[10.8.3 דו"ח גורמים מממנים 59](#_Toc175262559)

# מסמך רקע ואפיון עבור מערכת לניהול קליניקה פרטית Peace of Mind

## מבוא

מטפלים עצמאיים ניצבים בפני אתגרים רבים בניהול יעיל של קליניקה פרטית. ניהול ידני של משימות כמו קביעת תורים, מעקב אחר תשלומים מכמה גורמים, ניהול תיקי מטופלים ותיעוד טיפולים יכול להיות גוזל זמן ומייגע, תוך גרימת טעויות פוטנציאליות.

מערכת ניהול קליניקה אוטומטית נועדה לסייע למטפלים להתגבר על אתגרים אלו על ידי אוטומציה של משימות ניהוליות רבות. כתוצאה מכך, מטפלים יכולים להקדיש יותר זמן למטופלים שלהם ולשפר את איכות הטיפול.

המערכת שפתחנו יוצרת כעין "יומן חכם" עבור המטפל מנהל הקליניקה הן מבחינת ניהול מצבת הפגישות והן מצד ניהול כספי של עלויות הפגישות.

המערכת יודעת לנהל עבור מנהל הקליניקה בקלות, במהירות את המטופלים, הפגישות, יוצרת דרישות לתשלום לגורמים הרלוונטים, מסכמת כמה כל לקוח חייב לקליניקה עבור הטיפולים ועוד.

וגורמת בכך למנהל הקליניקה להשקיע בתחומים בהם הוא הכי טוב, ללא כאבי ראש מיותרים.

כיום המערכת מתוכננת כמערכת לניהול קליניקה פרטית של מטפל יחיד , כאשר הוא מטפל במספר מטופלים בו זמנית. יחד עם זאת בתכנון המערכת הושקע מחשבה כיצד ניתן להרחיבה למערכת לניהול קליניקה בעלת מספר מטופלים ומטפל ראשי.

שימוש במערכת ניהול קליניקה אוטומטית מציע מספר יתרונות משמעותיים למנהלי הקליניקה:

* **חיסכון בזמן:** אוטומציה של משימות ידניות חוזרות ונשנות משחררת זמן יקר שניתן להקדיש למטופלים ולפיתוח העסק
* **שיפור היעילות:** ניהול יעיל יותר של תורים, תשלומים ותיקי מטופלים מוביל לשיפור היעילות והרווחיות של הקליניקה
* **הפחתת טעויות:** מערכות אוטומטיות מפחיתות את הסיכוי לטעויות אנושיות, כגון תזמון יתר של תורים או חיוב שגוי של מטופלים
* **שיפור שביעות רצון המטופלים:** שירות יעיל , זמין ומהיר יותר תורם לשביעות רצון גבוהה יותר של המטופלים
* **גישה למידע:** המערכת מציעה גישה נוחה למידע על מטופלים, תשלומים וטיפולים ובכל זמן.

המערכת מפותחת כאפליקצית desktop ומיועדת להיות מופעלת על מחשבי הקליניקה.

המערכת פותחה בצורה גנרית , כך שתוכל להתאים למגוון בעלי קלינקות פרטיות עצמאיות , ללא קשר לתחום הטיפול בו הם עוסקים.

המערכת מתאימה לניהול מגוון קלינקות קטנות בהם מנהל הקליניקה הוא המטפל העיקרי בה, כגון קלינאי תקשרת, מרפאים בעיסוק, תרפיה בבעלי חיים, תרפיה רגשית, פסיכודרמה ועוד.

מלבד פיתוח המערכת הגנרית אנו נדגים שימוש במערכת עבור "קליניקה לטיפול רגשי" peace of mind

הקליניקה מנוהלת ומתוחזקת על ידי משתמש אחד – מנהל המערכת שהוא גם המטפל הראשי.

דרישות המערכתהמערכת תהיה זמינה למטפל – מנהל הקלניקה הפרטית באמצעות תוכנה שולחנית.

המערכת כוללת אפליקציית מחשב למטפל, המפותחת תוך הקפדה על מראה עיצובי אחיד ונקי.

המערכת תשתמש בבסיס נתונים שולחני וכך תשמור על גיבוי וסנכרון נתונים.

# פונקצינליות עיקריות

### התחברות למערכת

* מנהל התוכנה יוצר בעת התחברות ראשונית למערכת שם משתמש וסיסמא למנהל המערכת
* מנהל המערכת (המטפל הראשי בקליניקה) מתחבר למערכת באמצעות שם משתמש וסיסמא

### פגישות טיפוליות

* מנהל המערכת יכול לראות כמה פגישות מתוכננות לאותו יום (במסך הבית)
* מנהל המערכת יכול לראות את מצבת הפגישות הקיימות במערכת.
* מנהל המערכת יכול לראות כמה פגישות התקיימו עד כה.
* מנהל המערכת יכול לראות כמה פגישות מתוכננות עבורו להיום
* מנהל המערכת יכול לראות מיהם המטופלים היום ובאלו שעות הטיפולים.
* מנהל המערכת יכול לראות כמה פגישות מתוכננות בסך הכל לעתיד
* מנהל המערכת יכול לקבוע פגישה ללקוח.
* מנהל המערכת יכול למחוק פגישה ללקוח. מחיקת הפגישה אפשרית רק אם היא פגישה מתוכננת – עתידית, ושאין אחריה עוד פגישות מתוזמנות לאותו לקוח , זוהי פגישה אחרונה בסדרה.
* מנהל המערכת יכול לעדכן פרטי פגישה ללקוח קיים.
* מנהל המערכת יכול לחפש פגישה קיימת במערכת.

### לקוחות:

* מנהל המערכת יכול לראות את מצבת הלקוחות הרשומים במערכת.
* מנהל המערכת יכול להוסיף לקוח חדש למערכת
* מנהל המערכת יכול למחוק לקוח מהעסק. מחיקה של לקוח קיים מתאפשרת אך ורק במידה שלאותו לקוח אין חובות למערכת ואין עוד פגישות מתוכננות עבורו במערכת
* מנהל המערכת יכול לעדכן פרטי לקוח קיים
* מנהל המערכת יכול לחפש לקוח קיים במערכת.

### גורמים מממנים:

* מנהל המערכת יכול לראות את מצבת הגורמים המממנים הרשומים במערכת.
* מנהל המערכת יכול להוסיף גורם מממן למערכת
* מנהל המערכת יכול למחוק גורם מממן מהעסק. מחיקה של גורם מממן קיים מתאפשרת אך ורק במידה שלאותו גורם מממן אין חובות לקליניקה ואין עוד לקוחות שממומנים על ידו במערכת.
* מנהל המערכת יכול לעדכן פרטי גורם מממן קיים.
* מנהל המערכת יכול לחפש גורם מממן קיים במערכת.
* מנהל המערכת יכול לראות את סכום החוב של גורם מממן למערכת

### תשלומים

* מנהל המערכת יכול עקוב אחרי תשלומים שהתקבלו לקליניקה.
* המערכת תשלח אימייל אוטמטי עם דרישה לתשלום לנושה עבורו הופקה הדרישה בעת יצירת דרישה חדשה לתשלום.
* מנהל המערכת יכול ליצור דרישה לתשלום תשלומים מגורמים ממממנים או מלקוח.
* מנהל המערכת יכול לסגור דרישה קיימת לתשלום (כלומר – שולם) ומאז הבקשה אינה ניתנת לעריכה.
* מנהל המערכת יכול לראות כמה הכנסות היו לקליניקה עד כה.
* מנהל המערכת יכול לראות כמה חובות יש למטופלים/ גורמים ממממנים לקליניקה.

# אלמנטים במערכת

### מנהל הקליניקה – מטפל ראשי

מנהל הקליניקה הפרטית הוא אחראי על התוכנה , ומפעיל את הקליניקה הפרטית שלו.

ההתחברות למערכת נעשית על ידי שם משתמש וסיסמא כפי שהוגדו על ידי מנהל הקליניקה הפרטית – מנהל המערכת בעת החיבור הראשוני והתקנת התוכנה.

למנהל הקליניקה הינו כיום הישות היחידה היכולה לבצע פעולות במערכת כפי שפורט למעלה.

יחד עם זאת תנתן אופציה להוספת מטפלים נוספים בקליניקה בהמשך במידת הצורך.

### דרישות לתשלום

דרישה לתשלום הינה ישות במערכת המתארת חוב הנוצר כתוצאה מטיפולים בקליניקה.

דרישה לתשלום יכולה להיות או דרישה לתשלום מלקוח במערכת, או דרישה לתשלום מגורם מממן

עבור כל דרישה לתשלום הקיימת במערכת ניתן לשלוח אימייל לאיש הקשר המתאים הכולל את פרטי החוב ואמצעי התשלום המקובלים.

עבור כל דרישה לתשלום יוצג:

* שם החייב
* סכום החוב.

## גורם מממן

גורם מממן , משמעו גורם אשר מסייע למטופלים לממן את טיפוליהם בקליניקה הפרטית. מאחר ותחום הטיפול הפרטי , ובייחוד תחום הטיפול הפרא רפואי הינו יקר מאד.

גורם מממן יכול להיות ביטוח משלים של קופת החולים השונות – כללית, מאוחדת, לאומית.

או ביטוח משלים רפואי פרטי – מגדל , כלל וכדו,

או ארגון חסד אשר מסייע לאנשים מעוטי יכולת לטפל בצרכיהם בקליניקה פרטית מקצועית, כמו ועד הרבנים , קופת צדקה שכונתית, יד אליעזר וכדו.

ישנם מטופלים אשר ממממנים באופן עצמאי את טיפוליהם בקליניקה ללא אף גורם חיצוני כלשהוא.

הטיפול בקליניקה יכול להיות ממומן במלואו בידי הגורם המממן , או ממומן באופן עצמאי ללא סיוע מאף גורם מממן, אך במרבית המקרים עלות הטיפול מתחלקת ביחסיות קבועה מראש , בין גורם מממן לבין המטופל עצמו. במקרה כזה התוכנה תדע לתת דוחות לתשלום הן, עבור הגורם המממן ועבור המשתשמש.

גורם מממן יכול לממן מטופלים רבים במערכת.

לכל מטופל רשאי להיות גורם מממן , אשר מסייע במידה קבועה מראש עבור מימון טיפוליו בקליניקה.

לכל גורם מממן קיימת דרישת לתשלום אחת במערכת

עבור כל גורם מממן יוצגו:

* שם הגורם המממן
* שם איש קשר
* אימייל איש קשר בארגון
* סכום השתתפות לפגישה

## לקוחות

לקוח הינו מטופל אשר בוחר לקבל שירותי טיפול פרטיים ברמה מקצועית במערכת. לכל לקוח נקבעת סדרת טיפולים בקליניקה כפי הצורך המקצועי.

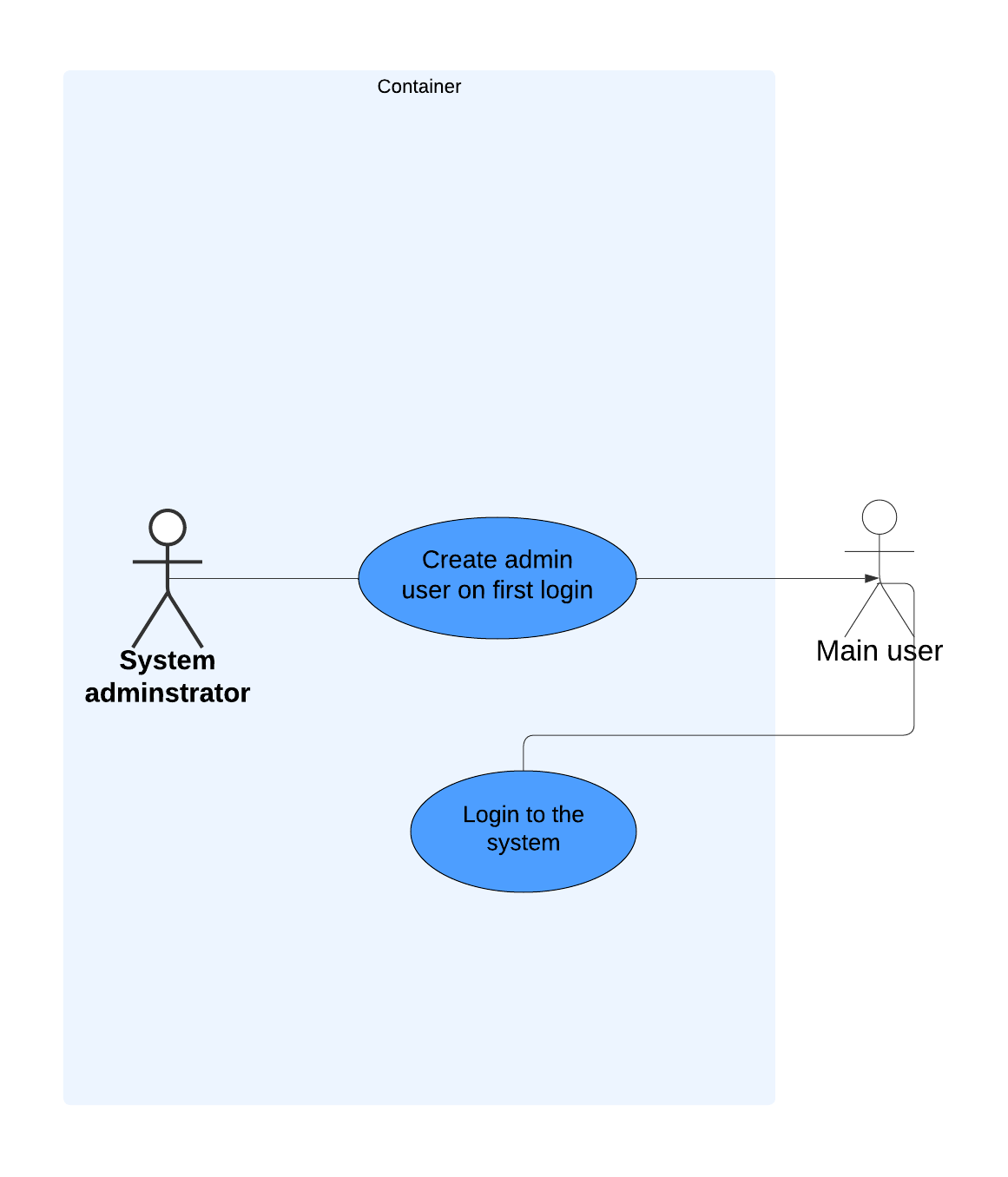
לקוח ראשי להעזר במימון הפגישה בגורם מממן, כאשר הגורם מממן משתתף בסכום קבוע מראש ושאר עלות הפגישה היא על הלקוח. במידה וללקוח אין גורם מממן, כל עלות הפגישה היא עליו.

עבור כל לקוח יוצגו הפרטים הבאים:

* שם לקוח
* כתובת אימייל
* טלפון
* גיל
* גורם מממן לטיפולים

# דיאגרמות

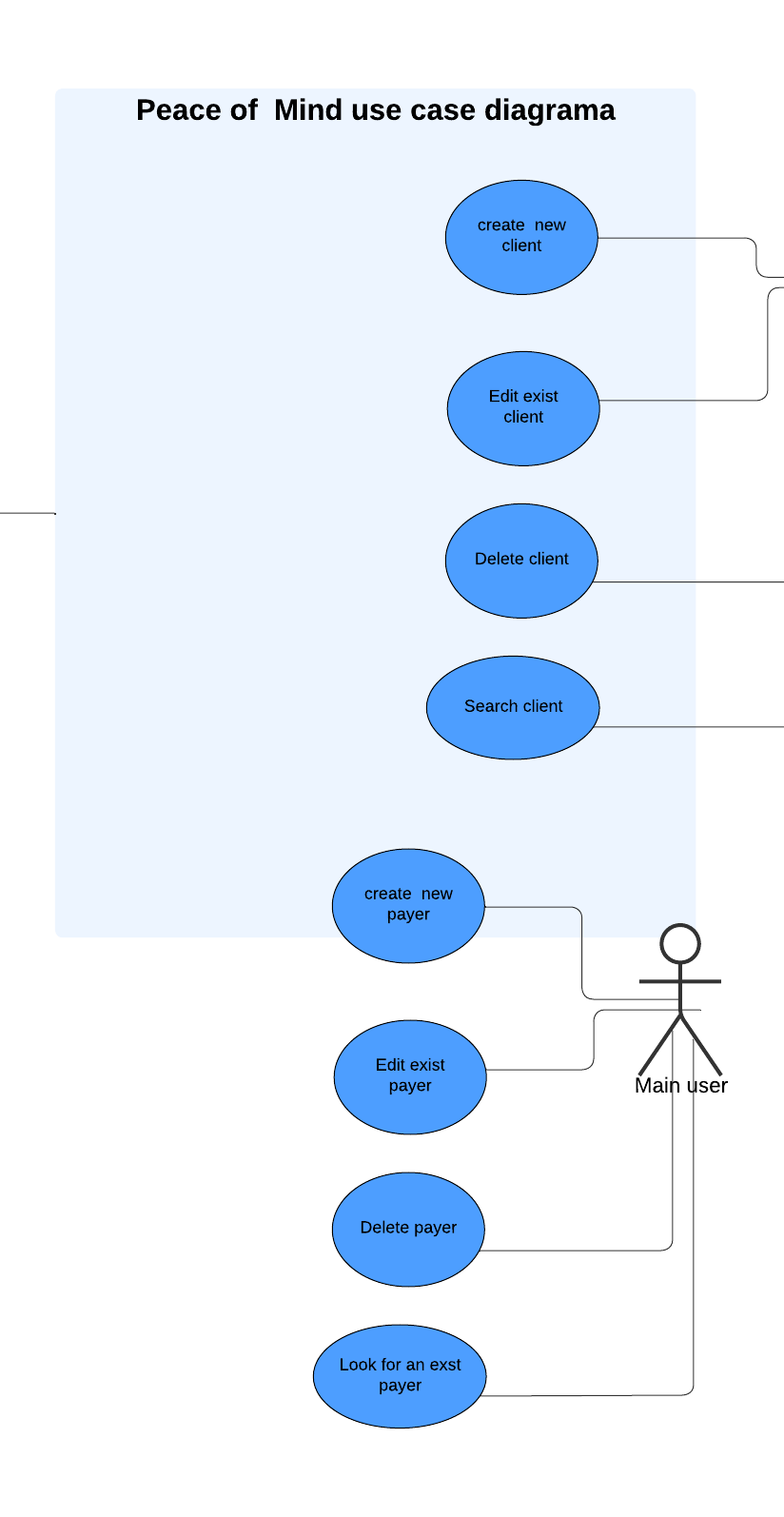
### דיאגרמת התחברות ראשונית למערכת



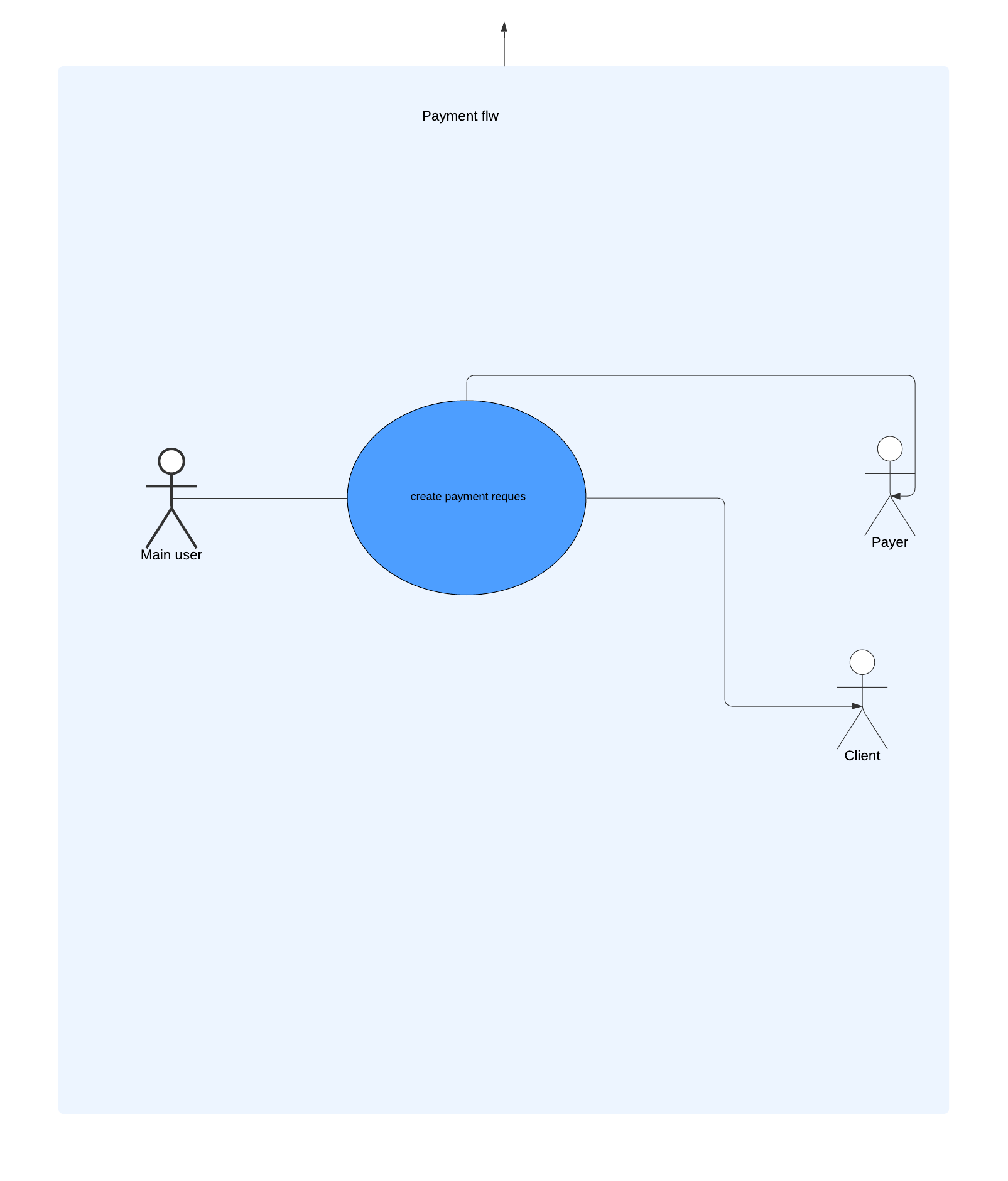
### פגישותA diagram of a case diagram Description automatically generated

### לקוחות

גורמים מממנים



### דרישה לתשלום



# מסמך תכנון ואפיון

## 1. תאור כללי של המערכת

בעלת שלוש שכבות MVVM בנויה בשיטת Peace of mind המערכת

1. **VIEW** - שכבת ממשק המשתמש בה המשתמשים מקיימים אינטרקציה עם המערכת, זהו החלק הנראה לעין של המערכת. השכבה הזו בנויה באמצעות טכנולגיה של WPF (Windows Presentation   
    Foundation).
2. שכבת **ViewModel**: שכבה זו משמשת כ"דבק" בין שכבת הממשק לשכבת הנתונים. היא מחזיקה את הלוגיקה של המערכת ומאפשרת לעדכן את הממשק בתגובה לשינויים בנתונים, ולהיפך.
3. שכבת **הModel** , שכבת נתונים: השכבה המחוברת ישירות למסד הנתונים SQL Server. היא אחראית על אחסון, גישה ושליפת נתונים.

**אופן פעולה בסיסי:**

המשתמש מקיים אינטראקציה עם הממשק: למשל, המשתמש מוסיף לקוח חדש למערכת ולוחץ על כפתור.

ViewModel השינוי שנעשה בממשק מועבר לשכבת ה-

שכבה זו מבצעת בדיקות תקינות על הנתונים, יכולה בצע חישובים כמו גיל, ולאחר מכן מעבירה את הנתונים לשכבת המודל (הנתונים) ע"י עדכון, כתיבה או מחיקה של הנתונים אל מסד הנתונים, וכ להפך כאשר יש שינויים במסד הנתונים הנתונים הללו מתגלגלים בשכבות ועוברים אל המודל – שכבת התצוגה.

### 1.1 יתרונות הגישהMVVM

הפרדת תפקידים: כל שכבה אחראית על חלק ספציפי בפונקציונליות של המערכת, מה שמגדיל את הניתנות לתחזוקה והקלות בהבנת הקוד.

קוד נקי יותר: הפרדת הלוגיקה העסקית מהממשק מאפשרת לכתוב קוד ברור יותר וקל יותר לתחזוקה.

ניתנות לבדיקה גבוהה יותר: ניתן לבדוק כל שכבה בנפרד, מה שמקל על איתור באגים ותיקונם.

. קיימים כלים רבים לתמיכה,MVVM: ישנם כלים רבים הזמינים לפיתוח אפליקציות מבוססות MVVM

מה שמפשט את תהליך הפיתוח.

המערכת היא מערכת טיפוסית מבוססת MVVM, המשלבת את היתרונות של WPF ןMVVM , ומבנה זה מאפשר פיתוח אפליקציות מודרניות וניתנות להרחבה.

## דגשים והנחיות בפיתוח התוכנית

* בכל רגע נתון רק משתמש אחד מחובר למערכת
* המערכת תומכת במחיקת לקוחות או גורמים ממממנים אך ורק בתנאי שלא מקושרים אליו אף אלמנטים פעילים. לדוגמא: לא ניתן למחוק מישהו שיש לו חוב למערכת, לא ניתן למחוק פגישה כאשר יש אחריה פגישות נוספות בסדרה, ניתן למחוק רק את הפגישה האחרונה בסדרה, ורק פגישה מתוכננת – לא לשנות את ההיסטוריה ולשבש תשלומים וחשבונות.
* בכל הדו"חות הקיימים במערכת ובכל טבלאות הנתונים ברירת המחדל הינה להציג את כל הנתונים הקיימים במערכת. ניתן להעזר בשדה החיפוש על מנת לצמצם את הנתונים המוצגים.
* דו"חות המערכת מאפשרים להציג סכום של הכנסות עבר והכנסות עתידיות מהקליניקה. יחד עם זאת היא לא תציג את הרווח שהעסק הרוויח עבור אותן פגישות. הסיבה לכך היא שנושא הנהלת החשבונות וחישוב רווחים והפסדים הינו מחוץ לתחום של המערכת.
* כל החלונות המוצגים בתוכנה ניתנים לשינוי גודל, הגדלה, הקטנה ומזעור.   
  תוכן החלון המוצג יהיה תואם , מיושר ומותאם לגודל החלון הקיים באותו זמן.
* במהלך פיתוח התוכנית שם המשתמש UserID של המטפל המחובר בתוכנה נשמר בקובץ לצורך שימוש במהלך כל התוכנית. הקובץ נשמר לתייקית מסמכים של המחשב נוצרת שם תיקיית Pece Of Mind ושם יוצר קובץ TEMP הקובץ נמחק אחרי שימוש, אחרי התנתקות משתמש או סגירת התוכנה.

### 2.1 מוסכמות רישום

* כל שם מחלקה/ פונקציה/ שדה שכוללת יותר ממילה אחת נכתבת עם אות גדולה בתחילת כל מילה נוספת.
* כל מחלקה מתחילה באות גדולה. לממשקים נוספה האות I לפני שם המחלקה.
* כל שדה פרטי במחלקה מתחיל ב '\_'
* שם מחלקה נגמר במילה המתארת את השכבה שלו Model/ ViewModel / View

# 3. שכבת בסיס הנתונים (namespce - loginDb.Repositories )

### 3.1 תאור המחלקות המשמשות להתקשרות

שכבה זו אחראית על התקשרות מול מסד הנתונים , ובצוע פעולות שוטפות מולו כמו מחיקה, שינוי, הוספה ושליפת נתונים. המערכת עושה שימוש במסד הנתונים של Microsoft SQL server Management.

שכבה זו מכילה מחלקה בשם **RepositoryBase** מחלקה זו הינה אחראית על יצירת ההתקשרות מול מסד הנתונים ויוצרת SQLConnection , מחלקה זו הינה אבסטרקטית, דבר המבטיח חיבור יחיד למסד הנתונים (יפורט בהמשך).

את החיבור למסד הנתונים ניתן לקבל באמצעות הפונקציה protected GetConnection.

protected SqlConnection GetConnection()

{

return new SqlConnection(\_connectionString);

}

שכבה זו מכילה מחלקה נוספת בשם **UserRepository** היא יורשת ממחלקת RepositoryBase ומממשת את הממשק IUserRepository .

מחלקה זו כוללת פונקציות רבות של התקשרות למסד הנתונים , כאשר החיבור למסד הנתונים נעשה ע"י המתודה GetConnection שתוארה לעיל , בינהן קימות גנריות להוספת אלמנטים כלליים (כל סוג אלמנט שהוא) למסד הנתונים:

public void Add<T>(T entity) where T : class

משמשת להוספת אוביקט אל מסד הנתונים

public void Edit<T>(T entity) where T : class

משמשת לעריכת אוביקט קיים במסד הנתונים

public IEnumerable<T> GetWhere<T>(Expression<Func<T, bool>> predicate) where T : class

קבלת נתונים ממסד הנתונים בתנאי מסוים

public IEnumerable<T> GetAll<T>() where T : class

קבלת כל הנתונים ממסד הנתונים

public void Remove<TEntity>(TEntity entity, string property) where TEntity : class

הסרת אלמנט

public async Task<(int NumOfClients, int NumOfMeetings, int Revenue, int Receivable)> LoadAllAsync(int UserId)

טעינת כל האלמנטים ממסד הנתונים

public int GetDebtById(int id, bool isClient, int Uid)

קבלת חוב עבור משתמש מסוים – לקוח או גורם מממן

public bool AuthenticateUser(NetworkCredential credential)

זיהוי פרטי משתמש – האם שם המשתמש והסיסמא תקינים

### 3.2 התחברות למסד הנתונים EntityFramwork

WPF משתמש בדרך כלל במסגרת Entity Framework כדי להתחבר ל-SQL Server. Entity Framework מאפשרת מניפולציה וולידציה של נתונים באמצעות מחלקות הדומות לטבלאות במסד נתונים.

תבנית המאגר מספקת שכבת ביניים המסתירה את אופן הגישה לנתונים, ומשפרת את הגמישות והתחזוקה של קוד המקור.

### 3.3 תבניות עיצוב המשמשות להתחברות אל מסד הנתונים

* **Repository Pattern:**
  + מספק שכבת הפשטה לניהול גישה לנתונים.
  + מגדיר פונקציות CRUD (יצירה, קריאה, עדכון ומחיקה) גנריות עבור ישויות (Entities).
  + מבודד את ההיגיון העסקי מהפרטים הספציפיים של מקור הנתונים (כגון SQL Server).
* **Generic:**
  + מאפשר פונקציה יחידה <T> Add להוסיף ישויות מסוג T כלשהו.
  + משפר את ההסתגלות והשימוש החוזר של הקוד עבור טיפוסי ישויות שונים.
* **Singelton**

כאשר מסד נתונים מיושם באמצעות תבנית הסינגלטון, אנו מבטיחים כי תהיה רק מופע אחד של החיבור למסד הנתונים בכל היישום. זאת אומרת, לא משנה כמה פעמים תנסו ליצור חיבור חדש, תמיד תקבלו את אותו החיבור הקיים.

* **ניהול משאבים:** חיבור למסד נתונים הוא משאב יקר. שימוש בסינגלטון מאפשר לנו לנהל את החיבור הזה בצורה יעילה, למנוע פתיחה של חיבורים מרובים ולהקטין את העומס על מסד הנתונים.
* **נקודת כניסה יחידה:** כל אינטראקציה עם מסד הנתונים מתבצעת דרך אותו מופע יחיד, מה שמפשט את הקוד ומאפשר בקרה מרכזית על הגישה לנתונים.
* **מצב:** אם יש צורך לשמור מידע על מצב החיבור (למשל, האם החיבור פתוח או סגור), סינגלטון מאפשר לנו לאחסן את המידע הזה במקום מרוכז.

**יתרונות**

* **קוד גמיש:** מאפשר הוספת ישויות מסוגים שונים באמצעות פונקציה אחת.
* **תחזוקה קלה:** השימוש בגנריות וב-Entity Framework מקל על הבנה ותחזוקה של הקוד.
* **הפרדת שכבות:** תבנית ה-Repository Pattern מבטיחה הפרדה ברורה בין שכבות האפליקציה.

לדגמא:

פונקציה זו משמשת ליצירת חיבור עם מסד הנתונים , כאשר אוביקט החיבור הינה אוביקט יחיד

public **abstract** class RepositoryBase

{

private readonly string \_connectionString;

public RepositoryBase()

{

\_connectionString = "Server=(local); Database=POMdb; Integrated Security=true";

}

protected SqlConnection GetConnection()

{

return new SqlConnection(\_connectionString);

}

}

פונקציה זו משמשת להוספת אוביקט למסד הנתונים (האוביקט יכול להיות, מטופל, מטפל, פגישה, גורם מממן ועוד)

הקוד לדוגמה, <T>Add, מתאר פונקציה גנרית המשתמשת ב-Repository Pattern כדי להוסיף ישות מסוג T לבסיס הנתונים.

public void Add<T>(T entity) where T : class

{

using (var db = new POMdbEntities())

{

var dbSet = db.Set<T>();

dbSet.Add(entity);

db.SaveChanges();

}

}

## 3.4 תאור מסד הנתונים:

## 3.4.1 קשרים לוגיים בין אוביקטים

* מטופל יכול לקבוע מספר פגישות.
* כל פגישה קשורה למטופל אחד ולמטפל אחד.
* לכל מטפל יכול להיות מטופלים רבים.
* לכל מטופל יכול להיות מטפל אחד.
* לכל מטופל יכולות להיות מספר רב של פגישות.
* לכל מטופל יכול להיות רק גורם מממן אחד ( זהו לא קשר חובה, לקוח יכול לממן את טיפוליו עצמאית).
* גורם מממן יכול לממן מספר רב של לקוחות בו זמנית.
* דרישה לתשלום יכולה להיות בכל פעם לאוביקט אחד – לקוח או גורם מממן
* לכל לקוח או גורם מממן יכולות להיות מספר דרישות לתשלום, אך תמיד רק דרישה אחת לתשלום תהיה פתוחה

## 3.4.2 טבלת חשבון משתמש UserAccount

טבלה זו מייצגת שם משתמש של מטפל במערכת.

טבלה זו מיועדת להרחבה עתידית של התוכנה שתכלול הוספת מטפלים נוספים, מזכירה וכדו

|  |  |  |
| --- | --- | --- |
| **שדה** | **טיפוס** | **הסבר** |
| UserName | מחרוזת | שם משתמש מקוצר, מפתח ראשי, מפתח זר |
| DisplayName | מחרוזת | שם המשתמש המשמש לתצוגה |
| ProfilePic | מחרוזת | נתיב לתמונת פרופיל \* |

נתיב לתמונת פרופיל מיועד להרחבה עתידית של התוכנה בה לכל משתמש תוצג התמונה שלו

## 3.4.3 טבלת מטפל User

טבלה זו מייצגת מטפל במערכת, כרגע ישנו רק מטפל אחד במערכת שהוא מנהל הקליניקה, אך ניתן להרחיב את התוכנה ולהוסיף עוד מטפלים.

השדה UserName הוא מפתח זר לטבלה UserAccount.

|  |  |  |
| --- | --- | --- |
| **שדה** | **טיפוס** | **הסבר** |
| ID | מספר שלם | מזהה משתמש- מטפל, מפתח ראשי, מפתח זר |
| UserName | מחרוזת | שם משתמש מקוצר, מפתח זר |
| Password | מחרוזת | סיסמת המשתמש |
| Name | מחרוזת | שם פרטי |
| LastName | מחרוזת | שם משפחה |
| Email | מחרוזת | אימייל מטפל – נערכת בדיקת תקינות על השדה, משמש לצורך שליחת סיכומים, תשלומים וכו |
| Price | מספר שלם | תעריף לפגישה, בכדי שיהיה ניתן לשנות את התעריף לפגישה שהמטפל לוקח |

## 3.4.4 טבלת לקוח Client

טבלה זו מייצגת מטופל בקליניקה, כרגע ישנו רק מטפל אחד במערכת שהוא מנהל הקליניקה, אך ניתן להרחיב את התוכנה ולהוסיף עוד מטפלים, ולכן לכל מטופל יש שדה המציין מיהו המטפל שלו.

השדה payerID מקושר בקשר של יחיד לרבים – לכל לקוח גורם מממן אחד, ולכל גורם מממן יכולים להיות לקוחות רבים.

|  |  |  |
| --- | --- | --- |
| **שדה** | **טיפוס** | **הסבר** |
| ID | מספר שלם | מזהה משתמש- מטופל, מפתח ראשי, מפתח זר |
| CName | מחרוזת | שם פרטי ושם משפחה של הלקוח |
| Phone | מחרוזת | שפ משפחה |
| Email | מחרוזת | אימייל מטופל – נערכת בדיקת תקינות על השדה, משמש לצורך שליחת דרישה לתשלום |
| BirthDate | DateTime | משמש לצורך חישוב גילו של המטופל נעשה ע"י convertor |
| PayerID | מספר שלם | מפתח זר, מייצג את הגורם המממן של הלקוח, יכול להיות NULL במידה והלקוח משלם עצמאית על כל הטיפולים. |

## 3.4.5 טבלת גורמים ממממנים Payers

טבלה זו מייצגת גורם המממן טיפולים בקליניקה, כמו קופות חולים, קופות צדקה וכו.

טבלה זו מקושרת לטבלת תשלומים בקשר של רבים לרבים – לכל גורם מממן יכולות להיות כמה דרישות לתשלום (הסתייגות חשובה היא שרק הדרישה האחרונה לתשלום יכולה להיות פעילה)

טבלה זו מקושרת לטבלת לקוחות בקשר של רבים ליחיד לכל לקוח יכול להיות גורם מממן אחד , כל גורם מממן יכול לממן 0 או יותר לקוחות.

|  |  |  |
| --- | --- | --- |
| **שדה** | **טיפוס** | **הסבר** |
| ID | מספר שלם | מזהה משתמש- גורם מממן, מפתח ראשי, מפתח זר |
| PName | מחרוזת | שם הגורם המממן |
| ContactName | מחרוזת | שם איש קשר |
| ContactEmail | מחרוזת | אימייל איש קשר – נערכת בדיקת תקינות על השדה, משמש לצורך שליחת דרישה לתשלום |
| TotalPayment | מספר שלם קצר short | משמש לצורך שמירה של סכום התשלומים הנדרש מגורם זה לקליניקה |

## 3.4.6 טבלת תשלומים Payment

טבלה זו מייצגת דרישה לתשלום עבור טיפולים בקליניקה.

המפתח הראשי של טבלה זו מיוצג ע"י מספר סידורי של הדרישה לתשלום , מזהה גורם מממן ומזהה מטופל.

הטבלה מקושרת לטבלאות גורמים מממנים ולקוחות בקשר של רבים לרבים.

לכל לקוח או גורם מממן יכולות להיות מספר דרישות לתשלום אך רק אחת מהן יכולה להיות פתוחה.

|  |  |  |
| --- | --- | --- |
| **שדה** | **טיפוס** | **הסבר** |
| ID | מספר שלם | מזהה בקשה, מפתח ראשי |
| CID | מספר שלם | מזהה לקוח , מפתח זר |
| PId | מספר שלם | מזהה גורם מממן, מפתח זר |
| Debt | מספר שלם | חוב בגין פגישה זו |
| IsOpen | משתנה בוליאני | משמש לצורך הגדרה האם הפגישה התקיימה או לא, תצוגת שדה זה נעשית ע"י convertor |

## 3.4.6 טבלת פגישות Meeting

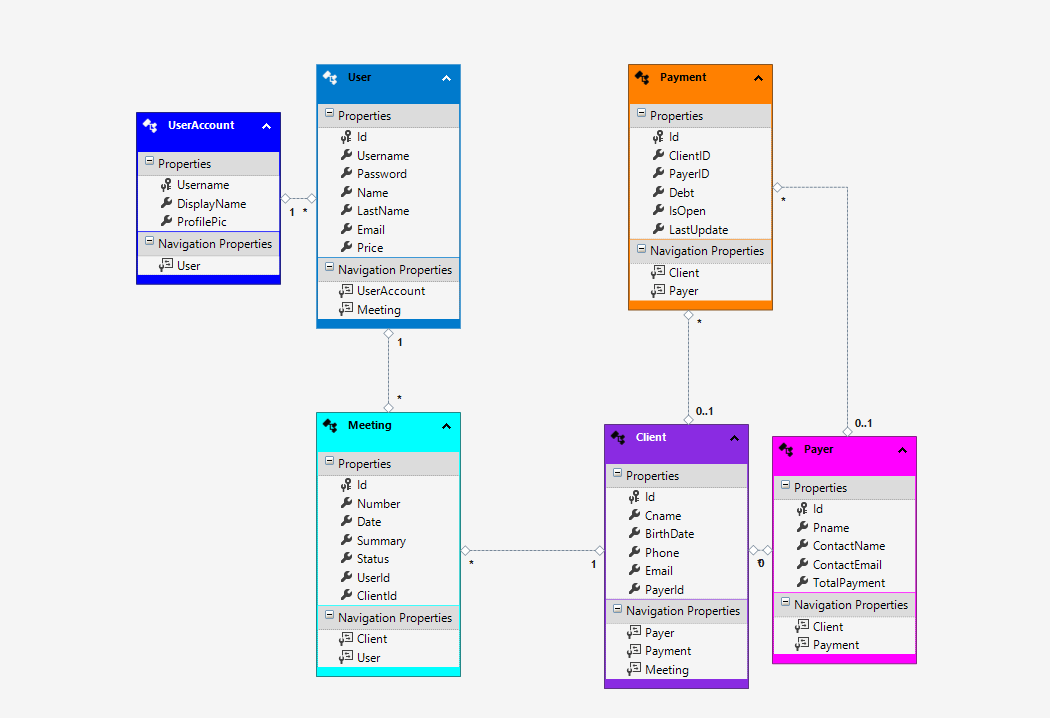
טבלה זו מייצגת פגישה בקליניקה, כרגע ישנו רק מטפל אחד במערכת שהוא מנהל הקליניקה, אך ניתן להרחיב את התוכנה ולהוסיף עוד מטפלים, ולכן לכל פגישה יש שדה המציין מיהו המטפל שלו.

לכל לקוח ולכל מטפל יכולות להיות פגישות רבות, אך בכל פגישה יכול להיות רק מטפל אחד ומטופל אחד.

המפתח הראשי של טבלה זו מיוצג ע"י מספר סידורי של הפגישה , מזהה מטפל ומזהה מטופל

|  |  |  |
| --- | --- | --- |
| **שדה** | **טיפוס** | **הסבר** |
| ID | מספר שלם | מזהה משתמש- מטופל, מפתח ראשי, מפתח זר |
| ClientID | מספר שלם | מזהה לקוח, מפתח זר |
| UserID | מספר שלם | מזהה מטפל, מפתח זר |
| Date | DateTime | תאריך ושעה של הפגישה |
| Status | Enum Status | משמש לצורך הגדרה האם הפגישה התקיימה, עתידית, מצב תשלום. תצוגת שדה זה נעשית ע"י convertor |
| Number | מספר שלם | מספר פגישה בסדרה עבור לקוח זה |
| Summery | מחרוזת | תעוד של הפגישה |

### 3.5 דיאגרמת מסד הנתונים



# שכבת הנתונים

המחלקות בשכבת המודל , מקושרות ישירות לאובייקטים ממסד הנתונים של sql server management

האוביקטים במחלקות אלו מקושרים ע"י Desirable לטבלאות המתאימות במסד הנתונים כפי שפורטו למעלה.

הדבר מיוצג על ידי תגיות מיוחדות מעל שם המחלקה

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

## 4.1 מחלקת user

מחלקה זו מייצגת אוביקט של מטפל בקליניקה. בעלת התכונות הבאות:

public int Id { get; set; }

public string Username { get; set; }

public string Password { get; set; }

public string Name { get; set; }

public string LastName { get; set; }

public string Email { get; set; }

public int Price { get; set; }

לכל מטפל במערכת קיימים שם משתמש וסיסמא יחידים הדבר מבוטא בקשר בין המחלקות

public virtual UserAccount UserAccount { get; set; }

לכל משתמש קיימת רשימה של פגישות. רשימת הפגישות מאותחלת לרשימה ריקה בעת יצירת הConstructor של המחלקה

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Meeting> Meeting { get; set; }

## 4.2 מחלקת user account

מחלקה זו משמשת ליצירת שם משתמש וסיסמא הנדרשים לצורך התחברות למערכת. מחלקה זו מכילה את השדות הבאים:

public string Username { get; set; }

public string DisplayName { get; set; }

public string ProfilePic { get; set; }

כל שם משתמש מקושר למשתמש במערכת (מטפל)

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual User User { get; set; }

## 4.3 מחלקת client

מחלקה זו משמשת לייצג את הלקוח בקליניקה . כאשר לכל לקוח יכולות להיות מספר פגישות (יחיד לרבים) וכן לכל לקוח יכול להיות רשימה של דרישות לתשלום (יחיד לרבים).

public int Id { get; set; }

public string Cname { get; set; }

public System.DateTime BirthDate { get; set; }

public string Phone { get; set; }

public string Email { get; set; }

public Nullable<int> PayerId { get; set; }

public virtual Payer Payer { get; set; }

לכל לקוח קיימת רשימה של דרישות לתשלום. רשימת הדרישות לתשלום מאותחלת לרשימה ריקה בעת יצירת הConstructor של המחלקה

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Payment> Payment { get; set; }

לכל לקוח קיימת רשימה של פגישות. רשימת הפגישות מאותחלת לרשימה ריקה בעת יצירת הConstructor של המחלקה

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Meeting> Meeting { get; set; }

## 4..4 מחלקת payers

מחלקה זו מייצגת גורם מממן , גורם המממן את טיפולי הלקוח בקליניקה. למחלקה זו השדות הבאים.

public int Id { get; set; }

public string Pname { get; set; }

public string ContactName { get; set; }

public string ContactEmail { get; set; }

public short TotalPayment { get; set; }

לכל גורם מממן קיימת רשימה של לקוחות בהם הוא תומך. רשימת הלקוחות מאותחלת לרשימה ריקה בעת יצירת הConstructor של המחלקה.

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Client> Client { get; set; }

לכל גורם מממן קיימת רשימה של דרישות לתשלום. רשימת הדרישות לתשלום מאותחלת לרשימה ריקה בעת יצירת הConstructor של המחלקה

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Payment> Payment { get; set; }

## 4..5 מחלקת Meeting

מחלקה זו מייצגת אוביקט של פגישה בקליניקה לכל פגישה קיימים המאפיינים הבאים:

public int Id { get; set; }

public int Number { get; set; }

public System.DateTime Date { get; set; }

public string Summary { get; set; }

public Status Status { get; set; }

public int UserId { get; set; }

public int ClientId { get; set; }

מייצג את הלקוח הנפגש בפגישה קשר של יחיד ליחיד

public virtual Client Client { get; set; }

מייצג את המטפל בפגישה – קשר של יחיד ליחיד

public virtual User User { get; set; }

## .4.6 מחלקת payment

מייצגת אוביקט של דרישה לתשלום של חוב לקליניקה – דרישה לתשלום יכולה להיות או מלקוח או מגורם מממן

public int Id { get; set; }

public Nullable<int> ClientID { get; set; }

public Nullable<int> PayerID { get; set; }

public int Debt { get; set; }

public bool IsOpen { get; set; }

public System.DateTime LastUpdate { get; set; }

מייצג את הלקוח שעבורו הוצאה הדרישה לתשלום קשר של יחיד ליחיד

public virtual Client Client { get; set; }

מייצג את הגורם המממן שעבורו הוצאה הדרישה לתשלום קשר של יחיד ליחיד

public virtual Payer Payer { get; set; }

## 4.7 Status.cs

מחלקת זו הינה מחלקת עזר המגדירה ENUM של סטטוס לפגישה

האם הפגישה שולמה/ לא שולמה/ שולמה ע"י לקוח/ שולמה ע"י גורם מממן

public enum Status : int

{

planned = 0,

unpaid = 1,

clientPaid = 2,

payerPaid = 3,

paid = 4

}

# מחלקות עזר convectors –loginDb.Converters

**תפקידם של converters ב- WPF**

ממירים (Converters) ב- WPF משמשים להמרת ערכים בין סוגי נתונים שונים לפני שהם מוקצים לתכונות של פקד או אובייקט אחר ב- XAML

לדוגמא, אתה יכול להשתמש בממיר כדי להמיר ערך בוליאני (true/false) ל רשימה המתארת ENUM עבור תווית WPF זה מאפשר לך לכתוב קוד XAML קריא ולתאר את הנתונים בצורה טבעית יותר עבור התצוגה.

יתרונות נוספים של ממירים כוללים:

* היכולת לשרשר ממירים מרובים יחד לביצוע המרות מורכבות יותר
* הפרדת לוגיקת ההמרה מקוד ה-XAML, מה שמוביל לקוד נקי יותר וניתן לתחזוקה

כל הממשקים יורשים מהממשק IValueConverter

**BooleanToStatusConverter**

משמש להצגה האם דרישה לתשלום הינה פתוחה או סגורה ממיר את השדה IsOpen לססטוס “open”/ “close”:

**DateTimeToStringConverter**

ממיר שדה מסוג תאריך למחרוזת בפורמט הרצוי לצורך הצגה לפגישה.

**DateToAgeConverter**

ממיר את התאריך לידה שמזינים בעת הוספת לקוח חדש לגיל שוצג בטבלת הלקוחות

**EnumToBooleanConverter**

משמש לטבלת פגישה מחזיר את הסטטוס של פגישה ובודק מה לסמן בסטטוס – מתוכנן/שולם / לא שולם...

**IdToNameConverter**

משמש להחליף את המספר זיהוי של הלקוח לשמו של הלקוח בטבלאות מקושרות כמו דרישות לתשלום, פגישות.

**ModeConverter,** ModeToVisibilityConverter

משמש להחליף בין תצוגה של edit / add אוביקאט חדש במסד הנתונים ( נעשה בכדי לחסוך בחלונות נוספים בעלי תפקיד דומה)

**ModeToVisibilityConverter**

**בתצוגת החלון הראשי** ניתן להציג חלון של הוספת או עריכה של אוביקט, הממיר הזה משמש בכדי להגדיר האם אנחנו רוצים להראות חלון מסוים או לא.

**TextToFontSizeConverter**

משמש בתצוגת הדוחות בא נכתב הסכום של החוב הקיים או העתידי לקליניקה. הסכום יכול להיות בעל מספר משתנה של ספרות 1-9999 , בכדי שתצוגת הספרות תכנס תמיד בחלון פותח ממיר זה.

**ToTargetConverter**

משמש בתצוגת דרישות לתשלום יכול להיות דרישה או מלקוח או מגורם מממן. יש trigger שאומר שאחד מהם חייב להיות ממומש והשני חייב להיות NULL

ממיר זה משמש להמיר את המספר הזה של לקוח או גורם מממן לשמו בכדי להציגו בתצוגת הפגישות.

# שכבת הלוגיקה view model

שכבת הלוגיקה view model (ViewModel) ב- MVVM אחראית על תיווך בין שכבת התצוגה (View) ושכבת המודל (Model). היא בעצם שכבה אמצעית שמספקת את הנתונים בצורה המתאימה לתצוגה ומטפלת באינטראקציות של המשתמש מהתצוגה אל המודל העסקי.

עקרונות מנחים בפיתוח שכבה זו:

* **ניתוק:** ה- ViewModel לא צריך להיות תלוי ישירות במימוש של שכבת התצוגה. זה מאפשר גמישות רבה יותר ויכולת שימוש חוזר ב- ViewModel בתצוגות שונות
* **IDataNotifyPropertyChanged:** ממשק זה מאפשר ל- ViewModel ליידע את התצוגה על שינויים בנתונים שלו, כך שהתצוגה יכולה להתעדכן אוטומטית.
* **Command Pattern:** דפוס תכנון זה מאפשר ל- ViewModel להגדיר פקודות שניתן לבצע על ידי המשתמש מהתצוגה. הפקודות יכולות לבצע פעולות שונות ב- ViewModel או ב Model

## 6.1 ViewModels/ViewModelBase.cs

מחלקה זו הינה מחלקה אבסטרקטית אשר ממנה יורשים add / edit הViewModel בפרויקט זו.   
במחלקה זו קיים מאפיין מסוג enum המציין האם אנחנו עכשיו בתצוגת הוספה או עריכה של אוביקט

וכן מחלקת זו נרשמה לארוע OnPropertyChanged בכדי לדעת על ארועים שיתרחשו בתוכנה (לחיצה, מעבר על אלמנטים וכדו) ולהגיב בהתאם

מחלקה זו מכילה פונקציה **ReadUserIdFromFile** public int ReadUserIdFromFile()

הפונקציה קוראת מספר זיהוי (ID) של משתמש מקובץ טקסט.

* הפונקציה תחילה בודקת האם הקובץ קיים.
* אם הקובץ לא קיים, היא זורקת חריג מסוג FileNotFoundException (שמטופלת בסוף ואז מוצגת למשתמש המלצה למחוק את התיקייה שבה א מור להשמר הקובץ)
* אם הקובץ קיים, הפונקציה קוראת את התוכן שלו בעזרת StreamReader ומחזירה את המספר שהומר שלם (int)

הפונקציה הזו מטרתה לאפשר הדבר נעשה לצורך התחברות קבועה למערכת,

USER ID נצרך לאורך כל חיי התוכנית, אך רק במסך של ההתחברות זה קיים - כי אז זה נקרא מהDB. ואין דרך להעביר את זה כפרמטר או משהו דומה לכל שאר המחלקות, היות שזה PARTIAL והכל מסונכרן עם הXAML הקריאות מתבצעות אוטומטית, ולא הצלחנו למצוא דרך להעביר את הUSERID לשאר המחלקות.   
הפתרון שנמצא הינו כאשר מתחברים לתוכנה נוצר הקובץ הזה, וכל פעם שצריך קוראים מהקובץ את התז, וכשסוגרים את התוכנה /מתנתקים, הקובץ נמחק.

## 6.2 ViewModels/ViewModelCommand.cs

מחלקת ViewModelCommand זו מייצגת מימוש של ממשק ICommand עבור שימוש בפקדים ב- WPF .

מחלקה זו הינה בשימוש בפקדים המוגדרים לאורך כל התוכנית ובכך מונעת שכפול קוד.

המחלקה מאפשרת הגדרת פקודות בצורה נוחה עבור ViewModel תוך שימוש בפרדיקט לבדיקת תנאי ההפעלה של הפקודה (CanExecute) ופונקציה לביצוע הפקודה (Execute).

המחלקה תומכת גם ב אירוע CanExecuteChanged המאפשר לעדכן את יכולת ההפעלה של הפקד. עדכון זה מתבצע דרך מנהל הפקודות של WPF (CommandManager).

## 6.3 ViewModels/LoginViewModel.cs

מחלקה זו, LoginViewModel, מייצגת ViewModel עבור תהליך רישום משתמש אדמינסטרטור לתוכנה. היא מכילה מאפיינים עבור פרטי המשתמש כגון שם משתמש, סיסמה, שם פרטי, שם משפחה, דואר אלקטרוני וכו'

. עבוד כל שדה ישנם getter and setter בהם נעשה רישום לOnPropertyChange עבור שדה זה

private string \_username;

private SecureString \_sPassword;

private string \_password;

private int \_id;

private string \_email;

private string \_firstName;

private string \_lastName;

private int \_price;

המחלקה מכילה גם פונקציות עבור אימות משתמש, הרשמה ו שחזור סיסמא.

פונקציית CanExecuteLoginCommand private bool CanExecuteLoginCommand(object obj)

מפעילה בידוק פשוט על שם המשתמש והסיסמא כדי לוודא שהם לא ריקים ובעלי אורך מינימלי .

פונקציית ExecuteLoginCommand private void ExecuteLoginCommand(object obj)

מנסה לאמת את המשתמש באמצעות השם והסיסמא מול מאגר נתונים כלשהו , במידה וכן מציגה אותו למשתמש , אחרת מציגה הודעת שגיאה מתאימה למשתמש.

פונקציית CanSignUpCommand - private bool CanSignUpCommand(object obj)

מפעילה בידוק ראשית האם קיים משתמש מחובר למערכת .במידה ולא בודקת על שם המשתמש האם השם משתמש כבר קיים במערכת, מוודאת את אורך הסיסמא, אורך תקין לשם המשתמש (לא יותר מ20 תווים), תקינות המייל לפי REGEX ועוד.

פונקציית ExecuteSignUpCommand private void ExecuteSignUpCommand(object obj)

מכניסה משתמש חדש למאגר הנתונים , כתיבת האוביקט למסד הנתונים נעשית ע"י UserRepository

## 6.4 ViewModels/HomeViewModel.cs

המחלקה `HomeViewModel` מנהלת נתוני משתמשים ומידע על פגישות בתוך האפליקציה. הנתונים יוצגו במסך הבית של התוכנה בפורמט הבא 'שלום \*\*\*\* יש לך היום X פגישות'

הוא מאחזר פרטי משתמש, מאכלס רשימה של פגישות עבור המשתמש הנוכחי בהתבסס על התאריך של היום, וחושף מאפיינים לקשירת נתונים Binding בתצוגה.

Properties:

\* `LstMeetings`: `ObservableCollection` המכיל רשימה של אובייקטי `Meetings` עבור המשתמש הנוכחי המתוכנן להיום

\* `IsViewVisible`: מאפיין בוליאני המציין את מצב הנראות של אלמנט תצוגה.

\* `DisplayName`: מאפיין מחרוזת המציג את שמו המלא של המשתמש שאוחזר ממסד הנתונים.

\* ‘Num’: מאפיין שלם המייצג את מספר הפגישות של המשתמש הנוכחי היום.

פונקציות

LoadMeetings(Expression<Func<Meeting, bool>> predicate) **LoadMeetings** \*

אוספת את כל הפגישות של המשתמש הנוכחי ומסננת אותן לפי התנאי שצוין בפרמטר predicate   
 מסדרת את הפגישות לפי תאריך בסדר יורד ומעדכנת את רשימת הפגישות המוצגת למשתמש, תוך הצגת רק את הפגישות של היום הנוכחי.

## 6.5 ViewModels/MainViewModel.cs

מחלקת MainViewModel משמשת כמודל הנתונים הראשי עבור היישום. היא אחראית על ניהול המידע של המשתמש הנוכחי, על ניווט בין מסכים שונים ביישום ועל ביצוע פעולות בסיסיות כמו כניסה ויציאה מהמערכת.

**מאפיינים:**

* CurrentUserAccount: מאחסן את פרטי המשתמש הנוכחי שנכנס למערכת.
* CurrentChildView: מאפשר ניווט בין מסכים שונים ביישום על ידי החלפת ה-View הנוכחי.
* Caption ו-Icon: משמשים לעדכון כותרת המסך וסמל במסך הראשי בהתאם למסך הנוכחי.
* IsViewVisible: מאפשר שליטה על נראות של חלקים שונים בממשק המשתמש.

**פקודות ניווט בין חלקי התצוגות השונים:**

* ShowHomeViewCommand: מעבירה את היישום למסך הבית הראשי.
* ShowClientsViewCommand: מעבירה את היישום לתצוגת הלקוחות.
* ShowMeetingsViewCommand : מעבירה את היישום לתצוגת פגישות.
* ShowReportsViewCommand: מעבירה את היישום להצגת תצוגת הדו"חות.
* ShowPayersViewCommand: מעבירה את היישום להצגת תצוגת גורמים מממנים.
* ShowPaymentsViewCommand: מעבירה את היישום להצגת תצוגת דרישות לתשלום.
* **פקודת יציאה:** LogoutCommand מבצעת את פעולת היציאה מהמערכת על ידי מחיקת פרטי המשתמש וניווט למסך הכניסה.

**מטודות במחלקה:**

* **LoadCurrentUserData** **-:** private void LoadCurrentUserData()

פונקציה זו אחראית לטעינת פרטי המשתמש הנוכחי מהמאגר ולאכלס אותם בViewModel.

* + היא קוראת את שם המשתמש מה-Identity של המשתמש המחובר.
  + לאחר מכן, היא מנסה לאחזר את פרטי המשתמש מהמאגר לפי שם המשתמש.
  + אם המשתמש נמצא, היא מעדכנת את המאפיינים ב-ViewModel CurrentUserAccount עם השם, שם משפחה ותמונת פרופיל (אם קיימת).
  + היא גם שומרת את מזהה המשתמש.
  + אחרת, היא מציינת שהמשתמש אינו מחובר [מציג הודעה מתאימה]
  + במקרה של חריגה במהלך הטעינה, היא מדפיסה הודעת שגיאה מתאימה.
* CreateUserIdFile פונקציה זו שומרת את שם הuser ID של המשתמש בתוך קובץ, הקובץ נשמר בניב קבוע בתוך התיקיות של הפרויקט. שם המשתמש נקרא במהלך התוכנית פעמים רבות (במהלך הקוד אין דרך לדעת מיהו הuse ID לכן נדרשת שמירה שלו בקובץ )

## 6.6 ViewModels/ReportsViewModel.cs

מחלקה זו ב- WPF ViewModelReports מספקת את הנתונים עבור תצוגת הדוחות של המשתמש המחובר.

היא אוספת נתונים אודות לקוחות וגורמים מממנים, מחשבת ערכים נוספים כגון מספר פגישות וחוב עבור כל פגישה, ולבסוף יוצרת אוספים מסוננים עבור התצוגה.

**properties**

* + NumOfClients (מספר לקוחות)
  + NumOfMeetings (מספר פגישות
  + Revenue (הכנסה)
  + Receivable (תקבולים – הכנסה צפויה)
  + FilteredClients (אוסף לקוחות מסוננים מחזירה אובייקט מסוג ObservableCollection FilteredPayers (אוסף משלמים מסוננים) מחזירה אובייקט מסוג ObservableCollection
  + IsViewVisible (קובע האם התצוגה גלויה)
  + ErrorMessage ( הודעת שגיאה)

**מטודה במחלקה:**

* : **LoadAll** : private async void LoadAll()

פונקציה זו אסינכרונית (async) ומטעינה את כל הנתונים הדרושים עבור תצוגת הדוחות של המשתמש המחובר בהתבסס על מידע מ-UserRepository

היא מתחילה בטעינת רשימות של לקוחות ומשלמים וקוראת נתונים מצטברים דרך המשתמש המחובר.

לאחר מכן עוברת ברשימות הלקוחות וגורמים מממנים ומחשבת עבור כל פריט נתונים נוספים כמו מספר פגישות וחוב.

לבסוף היא יוצרת אוספים מסוננים עבור לקוחות וגורמים מממנים המכילים שם, מספר פגישות וחוב.

פונקציה זו מניחה שהמשתמש כבר עבר אימות

## 6.7 ViewModels/ClientsViewModel.cs

מחלקה זו ClientsViewModel מאפשרת ניהול של רשימת לקוחות. בקליניקה.

המחלקה מחזיקה אוסף מסונן מסוג ObservableCollection של לקוחות ומאפשרת חיפוש, מחיקה וניהול של תצוגת הלקוחות.

Properties

* LstClients - אוסף מסונן של כל הלקוחות
* FilteredClients - אוסף מסונן של לקוחות לפי קריטריון חיפוש מחזירה אובייקט מסוג ObservableCollection
* SearchText - מחרוזת החיפוש של הלקוחות
* IsViewVisible - קובע האם התצוגה גלויה
* ErrorMessage - הודעת שגיאה.

**פונקציות ניווט בין חלקי התצוגה השונים:**

* ShowMeetingsCommand: מעבירה את היישום להצגת תצוגת פגישות.
* ShowEditCommand : מעבירה את היישום להצגת תצוגת עריכת לקוח קיים.
* ShowAddCommand: מעבירה את היישום להצגת תצוגת הוספת לקוח חדש.
* SearchCommand : מעבירה את היישום לתצוגה חלקית של לקוחות לפי תנאי החיפוש
* DeleteCommand: מעבירה את הלקוח למחיקת לקוח , המחיקה מתאפשרת בתנאי שהלקוח אינו פעיל – אין לו אף פגישה, חוב וכו.

**LoadClients - private void LoadClients(Expression<Func<Client, bool>> predicate)**

פונקציה זו אחראית לטעינת רשימת הלקוחות ולעדכון אוסף הלקוחות המסוננים. פונקציית המיון predicate מאפשרת סינון גמיש של הלקוחות לפי מאפיינים שונים. לדוגמא, ניתן לסנן את הלקוחות לפי שם הלקוח

שימוש בבדיקת null על פרמטר predicate מונעת קריסת התוכנית במקרה בו לא הועבר פרמטר מיון לפונקציה

## 6.8 ViewModels/MeetingsViewModel.cs

מחלקה זו MeetingsViewModel מאפשרת ניהול של רשימת פגישות בקליניקה.

המחלקה מחזיקה אוסף מסונן מסוג ObservableCollection של פגישות ומאפשרת חיפוש, מחיקה וניהול של תצוגת הפגישות.

Properties

* lstMeeting - אוסף מסונן של כל הפגישות
* FilteredMeetings - אוסף מסונן של פגישות לפי קריטריון חיפוש מחזירה אובייקט מסוג ObservableCollection
* SearchText - מחרוזת החיפוש של פגישות
* IsViewVisible - קובע האם התצוגה גלויה
* ErrorMessage - הודעת שגיאה.
* UserID – מזהה משתמש נקרא מהקובץ

**פונקציות ניווט בין חלקי התצוגה השונים:**

* ShowMeetingsCommand: מעבירה את היישום להצגת תצוגת פגישות.
* ShowEditCommand : מעבירה את היישום להצגת תצוגת עריכת פגישה קיימת.
* ShowAddCommand: מעבירה את היישום להצגת תצוגת הוספת פגישה חדשה.
* SearchCommand : מעבירה את היישום לתצוגה חלקית של פגישות לפי תנאי החיפוש
* DeleteCommand: מעבירה את הלקוח למחיקת פגישה , המחיקה מתאפשרת בתנאי שהפגישה הינה אחרונה בסדרה.

**LoadMeetings private void LoadMeetings(Expression<Func<Meeting, bool>> predicate)**

פונקציה זו אחראית לטעינת רשימת הפגישות ולעדכון אוסף הפגישות המסוננות. פונקציית המיון predicate מאפשרת סינון גמיש של הפגישות לפי מאפיינים שונים. לדוגמא, ניתן לסנן את הפגישות לפי שם הלקוח , מספר פגישה וכו

שימוש בבדיקת null על פרמטר predicate מונעת קריסת התוכנית במקרה בו לא הועבר פרמטר מיון לפונקציה

## 6.9 ViewModels/PayersViewModel.cs

מחלקה זו PayersViewModel מאפשרת ניהול של רשימת פגישות בקליניקה.

המחלקה מחזיקה אוסף מסונן מסוג ObservableCollection של גורמים מממנים ומאפשרת חיפוש, מחיקה וניהול של תצוגת הגורמים מממנים.

Properties

* lstPayers - אוסף מסונן של כל הגורמים מממנים
* FilteredPayers - אוסף מסונן של גורמים מממנים לפי קריטריון חיפוש מחזירה אובייקט מסוג ObservableCollection
* SearchText - מחרוזת החיפוש של גורמים מממנים
* IsViewVisible - קובע האם התצוגה גלויה
* ErrorMessage - הודעת שגיאה.
* UserID – מזהה משתמש נקרא מהקובץ

**פונקציות ניווט בין חלקי התצוגה השונים:**

* ShowPayersCommand: מעבירה את היישום להצגת תצוגת גורמים מממנים.
* ShowEditCommand : מעבירה את היישום להצגת תצוגת עריכת גורם מממן קיים.
* ShowAddCommand: מעבירה את היישום להצגת תצוגת הוספת גורם מממן חדש.
* SearchCommand : מעבירה את היישום לתצוגה חלקית של הגורמים המממנים לפי תנאי החיפוש
* DeleteCommand: מעבירה את התצוגה למחיקת הגורמים המממנים, המחיקה מתאפשרת בתנאי שהגורם המממן אינו פעיל – אין לו אף חוב וכו.

**LoadPayers private void LoadPayers(Expression<Func<Payer, bool>> predicate)**

פונקציה זו אחראית לטעינת רשימת הגורמים המממנים ולעדכון אוסף הגורמים הממממנים. פונקציית המיון predicate מאפשרת סינון גמיש של הפגישות לפי מאפיינים שונים. לדוגמא, ניתן לסנן את הפגישות לפי שם הלקוח

שימוש בבדיקת null על פרמטר predicate מונעת קריסת התוכנית במקרה בו לא הועבר פרמטר מיון לפונקציה.

## 6.10 ViewModels/PaymentsViewModel.cs

מחלקה זו PaymentsViewModel מאפשרת ניהול של רשימת דרישות לתשלום הקיימות בקליניקה.

המחלקה מחזיקה אוסף מסונן מסוג ObservableCollection של דרישות לתשלום ומאפשרת חיפוש, מחיקה וניהול של תצוגת הדרישות לתשלום.

חשוב לציין כי מחלקה זו תומכת בשליחת מייל אוטמטית דרך הפרוטוקול של גימייל לכתובת המייל הרשומה במערכת של גורם אליו הופקה הדרישה לתשלום (לקוח או איש הקשר של גורם מממן) בעת הוספת דרישה חדשה לתשלום למערכת.

מטרת המייל האוטמטי הינה לשלוח תזכרת לנישום אודות חובו למערכת

Properties

* lstPayments - אוסף מסונן של כל הדרישות לתשלום
* FilteredPayments - אוסף מסונן של דרישות לתשלום לפי קריטריון חיפוש מחזירה אובייקט מסוג ObservableCollection
* SearchText - מחרוזת החיפוש של דרישות לתשלום
* IsViewVisible - קובע האם התצוגה גלויה
* ErrorMessage - הודעת שגיאה.
* UserID – מזהה משתמש נקרא מהקובץ

**פונקציות ניווט בין חלקי התצוגה השונים:**

* ShowPaymentsCommand: מעבירה את היישום להצגת תצוגת דרישות לתשלום
* ShowEditCommand : מעבירה את היישום להצגת תצוגת עריכת דרישה קיימת לתשלום.
* ShowAddCommand: מעבירה את היישום להצגת תצוגת הוספת דרישה חדשה לתשלום
* SearchCommand : מעבירה את היישום לתצוגה חלקית של דרישות לתשלום לפי תנאי החיפוש
* DeleteCommand: מעבירה את התצוגה למחיקת דרישה לתשלום המחיקה מתאפשרת בתנאי שהדרישה לתשלום אינה פעילה -לא פתוחה.

**LoadPayments private void LoadPayments(Expression<Func<Payment, bool>> predicate)**

פונקציה זו אחראית לטעינת רשימת דרישות לתשלום ולעדכון אוסף דרישות לתשלום. פונקציית המיון predicate מאפשרת סינון גמיש של דרישות לתשלום לפי מאפיינים שונים. לדוגמא, ניתן לסנן את דרישות לתשלום לפי שם הלקוח

שימוש בבדיקת null על פרמטר predicate מונעת קריסת התוכנית במקרה בו לא הועבר פרמטר מיון לפונקציה

**הערה חשובה:** רשימת הViewModel שמוצגים כאן עכשיו משמשים בו זמנית הן לעריכת פרטי אוביקט והן להוספת אוביקט. הדבר נעשה בכדי למנוע שכפול קוד ולתחזוקה נקייה וקלה יותר של הפרויקט.

הבחירה האם אנחנו במצב הוספה או עריכה נשמרת במשתנה CurrentMode.

## 6.11 ViewModels/AddOrEditClientViewModel.cs

מחלקה AddOrEditClientViewModel מאפשרת תצוגה של הוספת לקוח או עריכת פרטי לקוח, זהו חלון המשמש הן לעריכה והן להוספה בהתאם להוראה שנשלחה. עבור תצוגת עריכה יופיעו כל השדות הקיימים של האוביקט הנבחר. עבור תצוגת הוספה יופיעו פקדים ריקים בהם יש למלא את פרטי הלקוח

**Properties**:

Payers: אוסף ObservableCollection המכיל אובייקטים של גורמים מממנים (משמש לבחירת גורם מממן המשויך ללקוח).

CurrentMode: מציין את המצב הנוכחי (הוסף או ערוך) עבור הלקוח

SelectedClient: אובייקט הלקוח שנבחר כעת לעריכה

IsViewVisible: מאפיין בוליאני השולט על הנראות של תצוגת העריכה של פרטי הלקוח.

ErrorMessage: מאחסן את כל הודעות השגיאה שנתקלו במהלך פעולות.

בנוסף קימיים מאפינים של לקוח (כפי שמפורט בטבלה) : ID, שם, תאריך לידה, טלפון ודואר אלקטרוני.

**פונקצית ניווט בין חלקי התצוגה השונים:**

* `AorECommand`: מציג את התצוגה המתאימה הוספה או עריכה של פרטי לקוח על סמך `CurrentMode` Edit/ Add.

**פונקציות:**

* **LoadPayers** שואב א כל נתוני הגורמים המממנים ממסד הנתונים דרך הUserRepostiry לצורך קישור גורם מממן ללקוח פעיל.
* `**CanAorECommand**`: מאמת אם ניתן להוסיף או לערוך לקוח על סמך תנאים שונים כמו תקינות הפרטים שהוכנסו – אימייל , טלפון , ID וכו.
* `**ExecuteOpenCommand**`: מעדכן את מצב `IsOpen` של הלקוח שנבחר או לקוח חדש במצב הוספה.

## 6.12 ViewModels/AddOrEditPaymentViewModel.cs

המחלקה AddOrEditPaymentViewModel מאפשרת תצוגה של הוספת או עריכת דרישה לתשלום, זהו חלון המשמש הן לעריכה והן להוספה בהתאם להוראה שנשלחה. עבור תצוגת עריכה יופיעו כל השדות הקיימים של האוביקט הנבחר. עבור תצוגת הוספה יופיעו פקדים ריקים בהם יש למלא את פרטי הדרישה לתשלום

**Properties:**

* ` Payments `: ObservableCollection של תשלומים לצפיה
* `LstPayers`: אוסף ObservableCollection של `Payer` המקושרים לדרישה לתשלום הנוכחית.
* `LstClients`: אוסף ObservableCollection של `Client` המקושרים לדרישה לתשלום הנוכחית.
* `CurrentMode`: מציין את המצב הנוכחי (הוסף או ערוך) עבור דרישה לתשלום.
* `SelectedPayment`: אובייקט `Payment` שנבחר כעת לעריכה.
* `IsClient`: בוליאנית המציינת אם הדרישה לתשלום משויך ללקוח.
* 'IsPayer': בוליאנית המציינת אם הדרישה לתשלום משויך לגורם מממן.
* `IsOpen`: בוליאנית המציינת אם בקשת התשלום פתוחה (לא שולמה) או סגורה.
* `Debt`: סכום התשלום – כמה הלקוח או הגורם המממן נדרשים לשלם.
* `ErrorMessage`: מאחסן את כל הודעות השגיאה שנתקלו במהלך פעולות.
* `Uid`: מזהה ייחודי עבור המשתמש, נטען מקובץ.
* 'Cid': מפתח זר המתייחס ל'לקוח' המשויך לתשלום (אינו שדה חובה).
* 'Pid': מפתח זר המתייחס ל'Payer' המשויך לתשלום (אינו שדה חובה).

**פונקציות ניווט בין חלקי התצוגה השונים:**

* `AorECommand`: מציג את התצוגה המתאימה הוספה או עריכה של דרישה לתשלום על סמך `CurrentMode` Edit/ Add.
* `OpenCommand`: מחליפה את מצב `IsOpen` של בקשת התשלום.

**פונקציות עזר:**

* 'LoadPayments': שואב את כל נתוני התשלום מ-'userRepository' וממלא את אוסף ה'Payments'.
* 'LoadAll': שואב את כל נתוני הגורם המממן והלקוח מ-'userRepository' ומאכלס את האוספים המתאימים.
* `CanAorECommand`: מאמת אם ניתן להוסיף או לערוך תשלום על סמך תנאים שונים כמו בקשות פתוחות וסכום החוב.
* `ExecuteOpenCommand`: מעדכן את מצב `IsOpen` של התשלום שנבחר או תשלום חדש במצב הוספה.
* `ClosePayment`: מעדכן את סטטוס הפגישות הקשורות לתשלום בהתבסס על המשלם/לקוח ומצב פתוח/סגור.
* `ExecuteAorECommand`: מבצע את ההיגיון המרכזי להוספה או עריכה של תשלום. הוא מאמת את קלט המשתמש, מוסיף/עורך את התשלום למאגר, שולח הודעת דוא"ל (אם רלוונטי), ומטפל בתרחישי הצלחה/שגיאה.
* `GetDebtDetailsForPayer`: מייצר הודעה מפורטת עם סיכום הפגישה והחוב הכולל עבור גורם מממן ספציפי.
* `GetDebtDetailsForClient`: מייצר הודעה מפורטת עם סיכום הפגישה והחוב הכולל עבור לקוח ספציפי.
* `HundleSending`: פונקציה לשליחת הודעת דואר אלקטרוני עם דרישה לתשלום.
* `GetUserCredential` (Async): מאחזר אישורי משתמש לגישה ל-Google Gmail API (עבור שליחת אימיילים).
* `SendEmail` (Async): שולח הודעת דוא"ל באמצעות Gmail API.

## 6.13 ViewModels/AddOrEditPayerViewModel.cs

מחלקה AddOrEditPayerViewModel מאפשרת תצוגה של הוספת או עריכת פרטי גורם מממן, זהו חלון המשמש הן לעריכה והן להוספה בהתאם להוראה שנשלחה. עבור תצוגת עריכה יופיעו כל השדות הקיימים של האוביקט הנבחר. עבור תצוגת הוספה יופיעו פקדים ריקים בהם יש למלא את פרטי הגורם המממן.

**Properties**:

Payers: אוסף ObservableCollection המכיל אובייקטים של גורמים מממנים (משמש לבחירת גורם מממן המשויך ללקוח).

CurrentMode: מציין את המצב הנוכחי (הוסף או ערוך) עבור גורם מממן.

SelectedClient: אובייקט הגורם המממן שנבחר כעת לעריכה

IsViewVisible: מאפיין בוליאני השולט על הנראות של תצוגת העריכה של הגורם המממן.

ErrorMessage: מאחסן את כל הודעות השגיאה שנתקלו במהלך פעולות.

בנוסף קימיים מאפינים של גורם מממן (כפי שמפורט בטבלה) : ID, שם, תאריך לידה, טלפון ודואר אלקטרוני.

**פונקציות ניווט בין חלקי התצוגה השונים:**

AorECommand: בוחר האם לעבוד במצב של הוספה או עריכה של לקוח בהתבסס על CurrentMode.

**פונקציות**:

LoadPayers: שואב את כל נתוני הגורמים המממנים מ-userRepository וממלא את אוסף הגורמים המממנים.

CanAorECommand: מאמת אם ניתן להוסיף או לערוך לקוח על סמך תנאים שונים כמו פורמט מזהה, אורך שם, טווח תאריכי לידה, פורמט מספר טלפון ופורמט דואר אלקטרוני (באמצעות Regex).

ExecuteAorECommand: מבצע את הלוגיקה המרכזית להוספה או עריכה של לקוח. הוא מאמת את קלט המשתמש, מוסיף/עורך את הלקוח למאגר, מטפל בתרחישי הצלחה/שגיאות ומציג בקצרה הודעת אישור לפני סגירת התצוגה.

## 6.14 ViewModels/AddOrEditMeetingViewModel.cs

מחלקה AddOrEditMeetingViewModel מאפשרת תצוגה של הוספת או עריכת פגישה, זהו חלון המשמש הן לעריכה והן להוספה בהתאם להוראה שנשלחה. עבור תצוגת עריכה יופיעו כל השדות הקיימים של האוביקט הנבחר. עבור תצוגת הוספה יופיעו פקדים ריקים בהם יש למלא את פרטי הפגישה.

**Properties**:

Clients: אוסף ObservableCollection המכיל אובייקטים של לקוחות (משמש לבחירת לקוח לפגישה שאותה עורכים או מוסיפים).

CurrentMode: מציין את המצב הנוכחי (הוסף או ערוך) עבור הפגישה

SelectedMeeting: אובייקט הפגישה שנבחר כעת לעריכה

IsViewVisible: מאפיין בוליאני השולט על הנראות של תצוגת העריכה של פרטי הפגישה.

ErrorMessage: מאחסן את כל הודעות השגיאה שנתקלו במהלך פעולות.

בנוסף קימיים מאפינים של פגישה (כפי שמפורט בטבלה) תאריך, שעה, סיכום, סטטוס , מזהה לקוח ומזהה מטפל

**פונקצית ניווט בין חלקי התצוגה השונים:**

* `AorECommand`: מציג את התצוגה המתאימה הוספה או עריכה של פרטי פגישה על סמך `CurrentMode` Edit/ Add.

**פונקציות:**

* **LoadClients** שואב את כל נתוני הלקוחות ממסד הנתונים דרך הUserRepostiry לצורך קישור לקוח פעיל לפגישה הנבחרת
* `**CanAorECommand**`: מאמת אם ניתן להוסיף או לערוך פגישה על סמך תנאים שונים כמו תקינות הפרטים שהוכנסו תאריך , בודק שלא ניתן להוסיף פגישה אם כבר נקבעה אחריה פגישה נוספת.

# 7. שכבת התצוגה

שכבה זו מייצגת את ממשק המשתמש כפי שהוא מוצג לעין ופתוחה בשפת XAML

בשכבת התצוגה הושקע מאמץ רב לעיצוב נקי ואלגנטי.

בטפסים שאינם חלונות צצים popup יעמוד לרשות המשתמש סרגל כלים עליון לביצוע פעולות (כגון חזרה לתפריט הראשי, שינוי משתמש, שליחת אימייל ועוד)

בפיתוח אפליקציות WPF באמצעות תבנית העיצוב MVVM, הקבצים **.xaml** ו-**.xaml.cs** משחקים תפקידים משלימים אך שונים:

* **.xaml:**
  + **הגדרת הממשק הגרפי:** קובץ זה מכיל את ההגדרה הוויזואלית של האלמנטים בממשק המשתמש, כמו כפתורים, תיבות טקסט, רשימות ועוד.
  + **שפת :XAML** הקובץ כתוב בשפת XAML (Extensible Application Markup Language), שהיא שפה דקלרטיבית המשמשת לתיאור הממשק הגרפי בצורה פשוטה וקריאה.
  + **קישור ל :ViewModel-** באמצעות ,XAML ניתן לקשר את האלמנטים הגרפיים לנכסים ופקודות ב-ViewModel מה שמאפשר עדכון דו-כיווני של המידע בין השניים.
* **.xaml.cs:**
  + **קוד מאחורי:** קובץ זה מכיל את הקוד שמתבצע מאחורי הקלעים של הממשק הגרפי.
  + **אירועים:** כאן מטפלים באירועים שונים שמתרחשים בממשק המשתמש, כמו לחיצה על כפתור או שינוי טקסט בתיבה.
  + **לוגיקה פשוטה:** לפעמים מבצעים כאן פעולות פשוטות שלא דורשות יצירת פקודה ב-ViewModel.
  + **קישור ל-ViewModel:** בדרך כלל, הקובץ הזה משמש לגישה ל-ViewModel ולהגדרת הקשר בין ה-View ל-ViewModel.

**עקרונות מנחים בשכבת התצוגה:**

* + ממשק המשתמש יהיה נוח ופשוט לתחזוק ושימוש.
  + נעשה שימוש בקבצי style חיצוניים לצורך שמירה על אחידות התצוגה (יפורטו בהמשך)
  + שימוש בuserControl במידה ואפשר עבור חלקים המשותפים לטפסים רבים בכדי למנוע שכפול קוד ועל מנת לאפשר שימוש חוזר.
  + שימוש בDATA GRID בקבצי הXAML בכדי לאפשר הגדלה והקטנה של החלונות לגודל גדול ולגודל קטן.
  + שימוש בBINDING בכדי לקשר בין שכבת התצוגה לשכבת ViewModel.
  + שימוש בדפוסי עיצוב Design Pattern במידת האפשר.

## 7.1 Custom Control

לצורך שמירה על עיצוב נעים ונקי כאשר לא היה פקד המתאים לייצוג סיסמא בצורה מתקדמת יצרנו פקד כזה. זהו פקד עיצוב מיוחד המוגדר תחת **customControl**

CustomControls/BindablePasswordBox.xaml

CustomControls/BindablePasswordBox.xaml.cs

משמש להצגת הסיסמא במסך החיבור בתצוגה של \*\*\*\* , ומטפל בקישור של הסיסמה כSecureString.

## 7.2 Styles

בכדי לשמור על עיצוב אחיד לאורך כל החלונות הרבים הקיימים בתוכנה נעשה שימוש בקבצים חיצוניים המכילים את העיצוב לכל החלונות הדבר תורם לשמירה על מראה עיצובי אחיד , מקל על עדכון התצוגה ומונע קוד כפול לאורך כל התוכנה

**בקובץ Styles/UIColors.xaml**

מכיל את צבעי רקע, צבעי פקדים, צבעי פונטים, צבעי פנלים לשימוש חוזר.

**Styles/ButtonStyles.xaml**

מכיל את תבניות העיצוב לכפתורים הרבים הקיימים בתוכנה- צבעים, גדלים, טריגרים ואירועים בזמן לחיצה וכו'.

## 7.3 טפסים במערכת

### 7.3.1 View/LoginView.xaml.cs

מחלקה זו מייצגת את חלון ההתחברות הראשונית של משתמש לתוכנה.

באמצעות חלון זה ניתן לבצע אחת מן הפעולות הבאות:

* + 1. כניסת משתמש קיים הרשום למערכת.
    2. רישום ראשוני למשתמש חדש למערכת.

האופציה הראשונה של כניסת משתמש רשום לתוכנה דורשת הקשת שם משתמש וסיסמא.

האופציה שניה של רישום משתמש חדש למערכת דורשת הכנסת פרטים מזהים הכוללים:

שם משתמש, סיסמא קבועה, שם פרטי, שם משפחה, תעודת זהות , כתובת איימיל ותעריף המטפל לשעה.

### 7.3.2 View/HomeView.xaml.cs

מחלקה זו מייצגת את דף הבית של התוכנה זהו החלון הראשון שנפתח לאחר התחברות המטפל לתוכנה

בחלון זה מוצג למטפל שהתחבר למערכת כמה פגישות צפויות לו היום

"שלום \*\*\* , יש לך היום X פגישות "

בנוסף מוצגת טבלה המציגה עבור כל פגישה את מספר הפגישה, שם המטופל ושעת הפגישה.

חלון זה יכול לגדול ולקטון ועיצוב התוכן ישתנה בהתאמה – לשמור על עיצוב נקי ונעים לעין בכל מצב.

### 7.3.3 View/MainView.xaml.cs

מחלקה זו מייצגת את מסך הבית של התוכנה.

מסך הבית מורכב משני חלקים, תפריט ניווט המכיל מסגרת של סרגלי ניווט עליון וצידי לנוחות המשתמש, ואזור של תוכן משתנה לפי הבחירה בסרגלי הכלים

סרגל הכלים העליון. היא מאפשרת הצגת שם המשתמש ותמונת פרופיל שלו ( כרגע זוהי תמונת לוגו ברירת מחדל, ניתן לשינוי בעתיד).

בסרגל זה ישנם כפתורים להגדלת החלון , הקטנת החלון וליציאה מהתוכנה , הסרגל הכלים העליון הינו קבוע ומופיע בכל החלונות בתוכנה למעט בחלונות הPOPUP

תפריט הניווט נמצא בצד שמאל של החלון ומשתמש בלחצנים כדי לנווט בין תצוגות שונות של היישום לכל כפתור סמל ותווית טקסט מתאימה כדי לציין את הפונקציונליות שלו – הוא מכיל כפתורים לניווט עבור: מסך הבית, פגישות, לקוחות, גורמים מממנים ודוחות.

התוכן הראשי נמצא בצד ימין של החלון ומציג את התצוגה הנוכחית של היישום, המשתנה לפי בחירת המשתמש. אזור תוכן זה מאוכלס על ידי 'ContentControl' שמתחבר למאפיין 'CurrentChildView' במודל התצוגה

בנוסף נעשה שימוש במברשות גבול ושיפועים משמשים לאורך החלון כדי ליצור עיצוב מושך מבחינה ויזואלית

### 7.3.4 View/ClientsView.xaml.cs

מחלקה זו מייצגת קוד האחראי לניהול הלקוחות ופרטיהם האישיים. ומוגדר באמצעות UserControl שניתן לשימוש חוזר עבור יישום WPF המנהל את פרטי הלקוח.   
הקוד חושף פונקציונליות לחיפוש, הוספה, מחיקה ועריכה של נתוני לקוח באמצעות ממשק ידידותי למשתמש.  
בנוסף הקוד מיישם ממירים לעיצוב עמודות רשת נתונים ספציפיות לקריאות משופרת. לדוגמה, 'dateToAgeConverter' הופך תאריכי לידה לגילאים מחושבים, בעוד שה-'IdToNameConverter' מתרגם של מזהה המשתמש ID לשמות ליצירת חווה ותצוגה טובה יותר למשתמשים.

טבלת נתונים מציגה רשימה של לקוחות עם עמודות מידע כמו תעודת זהות, שם, גיל, טלפון, דוא"ל וגורם מממן. עמודה "פעולות" מספקת לחצנים לעריכה ומחיקה של פרטי לקוח, דבר התורם לייעל את ניהול נתוני הלקוח

חשוב לציין כי כל הנתונים כדי מקושרים למסד הנתונים. זה מבטיח שהתצוגה משקפת אוטומטית כל שינוי שנעשה בנתוני הלקוח.

**חיפוש:** שורת חיפוש עם תיבת טקסט וכפתור מאפשרת למשתמשים לסנן את רשימת הלקוחות המוצגת על סמך קריטריונים ספציפיים.

**הוספת לקוח:** כפתור ייעודי מאפשר למשתמשים לנווט בצורה חלקה לתצוגה נפרדת להוספת לקוחות חדשים למערכת.

### 7.3.5 View/MeetingsView.xaml.cs

מחלקה זו מייצגת קוד האחראי לניהול הלקוחות ופרטיהם האישיים. ומוגדר באמצעות UserControl שניתן לשימוש חוזר עבור יישום WPF לניהול פגישות באפליקציה.   
הקוד חושף פונקציונליות לחיפוש, הוספה, מחיקה ועריכה של נתוני הפגישות באמצעות ממשק ידידותי למשתמש.  
טבלת נתונים מקיפה מציגה את פרטי הפגישות בצורה מאורגנת היטב, כולל מספר פגישה, מספר זהות של הלקוח, שם לקוח (המושג באמצעות ממיר IdToNameConverter), זמן פגישה (מושג באמצעות ממיר DateTimeToStringConverter), סטטוס פגישה (שולמה/ מתוכננת וכוד), ושדה טקסטואלי של סיכום הפגישה.  
 כל שורה בטבלה המהווה פגישה מלווה בכפתורי פעולה, המאפשרים למשתמשים לערוך או למחוק פגישות ישירות, ולקדם פעולות ניהול פגישות יעילות

חשוב לציין כי כל הנתונים כדי מקושרים למסד הנתונים. זה מבטיח שהתצוגה משקפת אוטומטית כל שינוי שנעשה בנתוני הפגישות.

**חיפוש:** סרגל חיפוש ייעודי מאפשר סינון פגישות על סמך קריטריונים ספציפיים, ומשפר את הניווט דרך בנתונים שעלולים להיות גדולים

**הוספת פגישה:** כפתור ייעודי מאפשר למשתמשים לנווט בצורה חלקה לתצוגה נפרדת להוספת פגישה חדשה למערכת, דבר המטפח ארגון פגישות יעיל

### 7.3.6 View/PayersView.xaml.cs

מחלקה זו מייצגת קוד האחראי לניהול הגורמים המממנים ופרטיהם האישיים. ומוגדר באמצעות UserControl שניתן לשימוש חוזר עבור יישום WPF לניהול הגורמים המממנים באפליקציה.   
הקוד חושף פונקציונליות לחיפוש, הוספה, מחיקה ועריכה של נתוני הגורמים המממנים באמצעות ממשק ידידותי למשתמש.  
טבלת הגורמים המממנים מציגה את פרטיהם בצורה מאורגנת היטב, כולל מספר גורם מממן (מספר רץ), שם הגורם המממן, שם איש הקשר, אימייל איש הקשר וסכום השתתפות של הגורם המממן בפגישה.  
 כל שורה בטבלה המהווה פגישה מלווה בכפתורי פעולה, המאפשרים למשתמשים לערוך או למחוק גורמים מממנים ישירות, ולקדם פעולות ניהול יעילות

חשוב לציין כי כל הנתונים כדי מקושרים למסד הנתונים. זה מבטיח שהתצוגה משקפת אוטומטית כל שינוי שנעשה בנתוני הפגישות.

**חיפוש:** סרגל חיפוש ייעודי מאפשר סינון גורמים ממממנים על סמך קריטריונים ספציפיים, ומשפר את הניווט דרך בנתונים שעלולים להיות גדולים

**הוספת גורמים מממנים:** כפתור ייעודי מאפשר למשתמשים לנווט בצורה חלקה לתצוגה נפרדת להוספת גורם מממן חדש למערכת.

### 7.3.7 View/PaymentsView.xaml.cs

מחלקה זו מייצגת קוד האחראי לניהול הדרישות לתשלום. ומוגדר באמצעות UserControl שניתן לשימוש חוזר עבור יישום WPF לניהול פגישות באפליקציה.   
הקוד חושף פונקציונליות לחיפוש, הוספה, מחיקה ועריכה של דרישות לתשלום באמצעות ממשק ידידותי למשתמש.  
טבלת נתונים מקיפה מציגה את פרטי הדרישות לתשלום בצורה מאורגנת היטב, כולל יעד הדרישה לתשלום – יכול להיות לקוח או גורם מממן (מושג באמצעות הממיר ToTargetConverter) סטטוס הדרישה לתשלום ( פתוחה או סגורה) סכום החוב, תאריך הפקת הדרישה לתשלום(מושג באמצעות ממיר DateTimeToStringConverter), סטטוס פגישה (שולמה/ מתוכננת וכוד).  
 כל שורה בטבלה המהווה דרישה לתשלום מלווה בכפתורי פעולה, המאפשרים למשתמשים לערוך או למחוק דרישות לתשלום ישירות, ולקדם פעולות ניהול גביה ומעקב אחר התשלומים בצורה יעילה.

חשוב לציין כי כל הנתונים כדי מקושרים למסד הנתונים. זה מבטיח שהתצוגה משקפת אוטומטית כל שינוי שנעשה בנתוני הפגישות.

**חיפוש:** סרגל חיפוש ייעודי מאפשר סינון דרישות לתשלום על סמך קריטריונים ספציפיים, ומשפר את הניווט דרך בנתונים שעלולים להיות גדולים

**הוספת פגישה:** כפתור ייעודי מאפשר למשתמשים לנווט בצורה חלקה לתצוגה נפרדת להוספת דרישה לתשלום חדשה למערכת, דבר המטפח ניהול תשלומים יעיל.

### 7.3.8 View/ReportsView.xaml.cs

מחלקה זו מגדירה UserControl בשם 'ReportsView' עבור יישום פיננסי של WPF. הכוללת מספר דוחות המוצגים בצורת סעיפים הניתנים להרחבה ומשמשים להצגת דוחות כספיים שונים בצורה מקיפה וידידותית למשתמש.  
לצורך תופסת בהירות למשתמש, כל קוביה (דיווח) גדול שעומדים עליו.

הדוחות המוצגים הינם:

1. סיכום כללי של נתונים מספר לקוחות פעילים, מספר פגישות בסך הכל, סכום כספי של הכנסות שהקבלו, סכום כספי של הכנסות עתידיות.
2. דו"ח לקוחות- עבור כל לקוח מוצג שמו, מספר הפגישות הכללי שלו וסכום החוב שלו לקליניקה.
3. דו"ח גורמים מממנים – עבור כל גורם מממן מוצג שמו, מספר הלקוחות הממונים דרכו וסכום החוב.

מצגת דוחות מודולרית: ה-UserControl משתמש במרחיבים כדי לסווג ולהציג דוחות, לקדם ארגון מידע ושיפור הקריאות

Data Binding: רכיבי ממשק המשתמש מתחברים למודל תצוגה, ומאפשרים עדכונים בזמן אמת על סמך הנתונים הפיננסיים הבסיסיים

### 7.3.9 View/AddOrEditClientView.xaml

מחלקה זו מגדירה חלון להוספה או עריכה של לקוח לאפליקציה לניהול הקליניקה.

החלון משתמש באיגוד נתונים וממירים כדי לנהל את רכיבי ממשק המשתמש בהתבסס על המצב הנוכחי (הוספה או עריכה של פרטי לקוח קיים)

החלון מספק ניהול מקיף של נתוני לקוח: כולל שדות ייעודיים להצגה של מידע חיוני על הלקוח, כולל מספר זהות, שם, תאריך לידה, פרטי קשר (טלפון, דוא"ל) ובחירת גורם מממן.  
בקרי ממשק המשתמש משנים את הנראות וההתנהגות שלהם בהתבסס על המצב הנוכחי (הוסף/עריכה) באמצעות איגוד נתונים וממירים. פקדים מסוימים כמו מספר זהות של הלקוח הינם לקריאה בלבד במצב עריכה

בנוסף קיים מנגנון טיפול בשגיאות בקלט הכולל בלוק טקסט ה מציג את הודעות השגיאה שנתקלו במהלך עיבוד הנתונים

הנתונים נשמרים במסד הנתונים בעת לחיצה על לחצן הוספה/עריכה: בהתבסס על המצב הנוכחי (הוסף או עריכה).

### 7.3.10 View/AddOrEditMeetingView.xaml.cs

מחלקה זו מגדירה חלון להוספה או עריכה של פגישה לאפליקציה לניהול הקליניקה

החלון משתמש באיגוד נתונים וממירים כדי לנהל את רכיבי ממשק המשתמש בהתבסס על המצב הנוכחי (הוספה או עריכה של פרטי פגישה קיימת)

החלון מספק ניהול מקיף של נתוני הפגישה: כולל שדות ייעודיים להצגה של מידע על הפגישה, כולל שם לקוח, תאריך הפגישה, שעת הפגישה וסטטוס הפגישה – מתוכננת / שולמה וכדו', ותקציר הפגישה.  
בקרי ממשק המשתמש משנים את הנראות וההתנהגות שלהם בהתבסס על המצב הנוכחי (הוסף/עריכה) באמצעות איגוד נתונים וממירים, פקדים מסוימים כמו מספר מזהה של הפגישה, הינם לקריאה בלבד במצב עריכה ואלו פקדים אחרים כמו סיכום הפגישה הינם לקריאה בלבד במצב הוספה.

בנוסף קיים מנגנון טיפול בשגיאות בקלט הכולל בלוק טקסט המציג את הודעות השגיאה שנתקלו בהן במהלך עיבוד הנתונים

הנתונים נשמרים במסד הנתונים בעת לחיצה על לחצן הוספה/עריכה: בהתבסס על המצב הנוכחי (הוסף או עריכה).

### 7.3.11 View/AddOrEditPayerView.xaml.cs

מחלקה זו מגדירה חלון להוספה או עריכה של גורם מממן לאפליקציה לניהול הקליניקה

החלון משתמש באיגוד נתונים וממירים כדי לנהל את רכיבי ממשק המשתמש בהתבסס על המצב הנוכחי (הוספה או עריכה של פרטי גורם מממן קיים)

החלון מספק ניהול מקיף של נתוני הגורם המממן: כולל שדות ייעודיים להצגה של מידע על הגורם המממן, כולל שם גורם מממן, שם איש הקשר, אימייל איש הקשר וסכום השתתפות לפגישה

בנוסף קיים מנגנון טיפול בשגיאות בקלט הכולל בלוק טקסט המציג את הודעות השגיאה שנתקלו בהן במהלך עיבוד הנתונים.

הנתונים נשמרים במסד הנתונים בעת לחיצה על לחצן הוספה/עריכה: בהתבסס על המצב הנוכחי (הוסף או עריכה).

### 7.3.12 View/AddOrEditPaymentView.xaml.cs

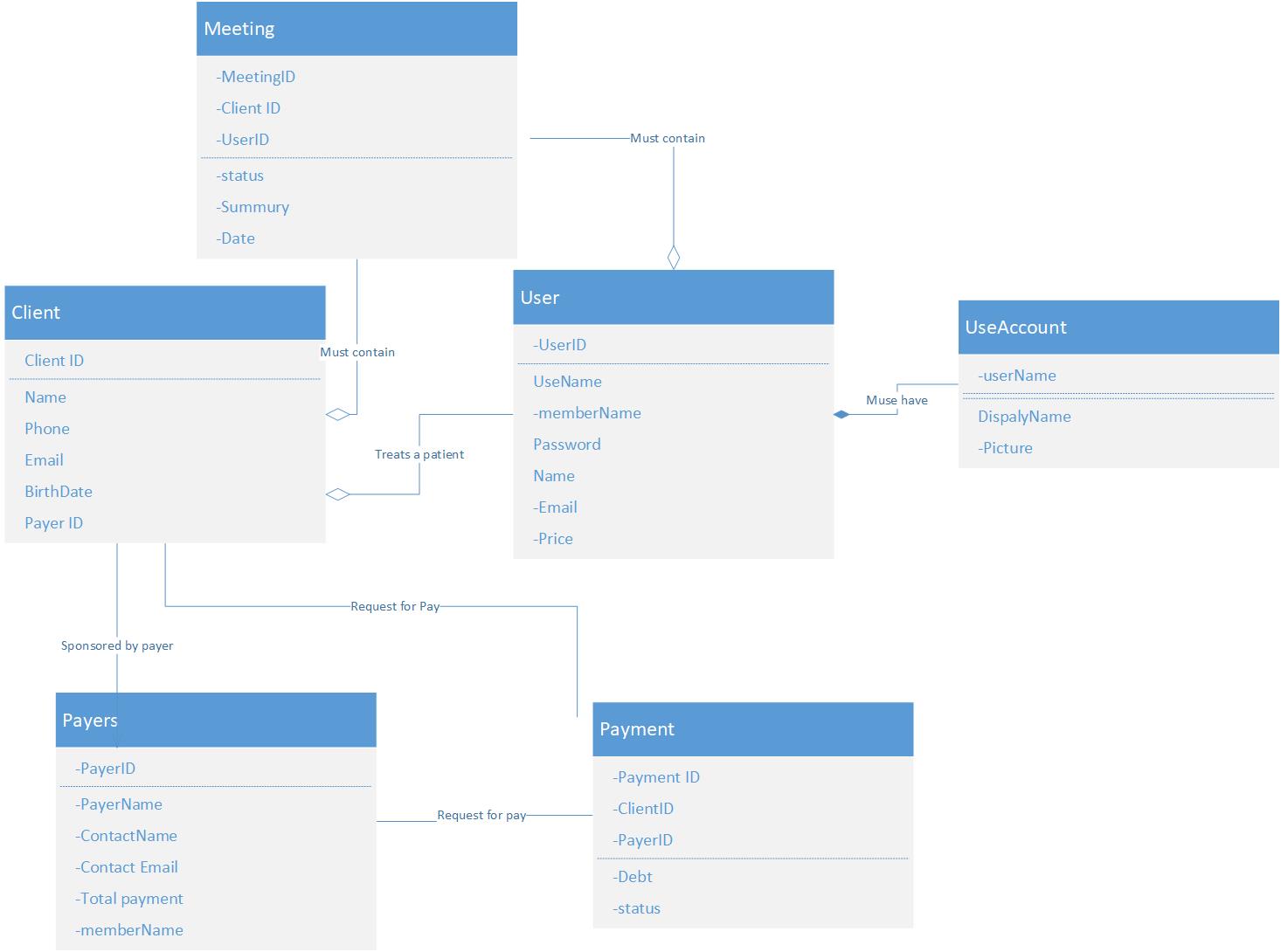
מחלקה זו מגדירה חלון להוספה או עריכה של דרישה לתשלום

החלון משתמש באיגוד נתונים וממירים כדי לנהל את רכיבי ממשק המשתמש בהתבסס על המצב הנוכחי (הוספה או עריכה של פרטי דרישה חדשה תשלום)

החלון מספק ניהול מקיף של נתוני הגורם המממן: כולל שדות ייעודיים להצגה של מידע על הדרישה לתשלום, כולל יעד הדרישה לתשלום – לקוח או גורם מממן – בהתאמה לכך יוצג שמו (מושג באמצעות ממיר ToTargetConverter) , סכום החוב וסטטוס הדרישה לתשלום (פתוחה או סגורה).  
בנוסף קיים מנגנון טיפול בשגיאות בקלט הכולל בלוק טקסט המציג את הודעות השגיאה שנתקלו בהן במהלך עיבוד הנתונים.

הנתונים נשמרים במסד הנתונים בעת לחיצה על לחצן הוספה/עריכה: בהתבסס על המצב הנוכחי (הוסף או עריכה).

## 8. דיאגרמת ישויות קשרים



### 9. שינויים עתידיים בתוכנה

### 9.1 עדכון תמונת פרופיל יחודית לכל משתמש

כיום לכל המשתמשים בברירת מחדל מוצגת תמונת פרופיל של משתמש אנונמי. בכדי ליצור חווית משתמש עתידית יהיה ניתן להרחיב את התוכנה כך שלכל משתמש תוצג תמונותו. לצורך כך בטבלת חשבון משתמש ישנו שדה המייצג תמונת משתמש.

## 9.2 הוספת מטפלים ואנשי צוות נוספים

כאמור לעיל התוכנה פותחה כתוכנה לניהול קליניקה פרטית, מטבע הדברים קליניקה פרטית הינה כוללת מטפל יחיד שהוא גם בעל העסק במרבית המקרים, מטפל בקבלות ובתשלומים.

אם זאת במהלך אפיון ,תכנון ופיתוח התוכנה, הוקדשה מחשבה מרובה בכדי שהמערכת תהיה מותאמת להרחבה בעתיד הן בהוספת מטפלים נוספים, והן בהוספת מזכירה לתפעול העסק.

לצורך כך נוסיף תצוגה של משתמשים במערכת וכן חלון להוספת ועריכה של מטפלים במערכת.

9.2.1. הוספת מטפל נוסף למערכת

שיהיה תחת אחריות מנהל הקליניקה

תכונות מטפל משני במערכת יהיו:

* למטפל המשני יווצר שם משתמש וסיסמא יחודיים לו, על ידי מנהל המערכת
* למטפל המשני לא תהיה הרשאה לראות מטופלים של מטפלים אחרות
* למטפל המשני תהיינה הרשאות לראות את המטופלים שלו בלבד.
* למטפל המשני תהיה הרשאה לראות את הפגישו המתוכננות עבורו בלבד
* למטפל המשני לא תהיה גישה לדוחות הכספיים

9.2.2 הוספת מזכירה לניהול הטכני של העסק

* למזכירה יווצרו שם משתמש וסיסמא יחודיים על ידי מנהל המערכת.
* למזכירה לא תהיינה הרשאות לראות את פרטי המטופלים.
* למזכירה תהיינה הרשאות לקבוע, לעדכן ולמחוק פגישות בקליניקה.
* למזכירה תהינה הרשאות לראות ולעדכן את הגורמים מממנים.
* למזכירה תהיינה הרשאות לראות וליצור את הדרישות לתשלום.
* למזכירה תהייה הרשאה לראות את הדוחות הכספיים של הקליניקה.

## 9.3 עדכון תעריף טיפול

בעולם הטיפול הפרטי, המחירים לטיפול פרטי הינם דינמיים ומשתנים. עלות לטיפול היום הינה כפולה לפחות מעלות הטיפול לפני 10-15 שנה. בהתאם לזאת במהלך פיתוח התוכנה הוצרכנו לחשוב על עליית המחירים לטיפול, דבר המצריך עדכון בתשלום הן ללקוח והן לגורם המממן.   
בנוסף יתכנו מטפלים שונים בקליניקה (כפי שפורט בנקודה קודמת) שלכל אחד מהם ישנו תעריף אחר לטיפול, אינו דומה תעריף לטיפול ע"י מטפל מומחה בעל שנים רבות של נסיון לסטאז'ר.   
במהלך תכנון הפרויקט הוספנו לכל מטפל שדה של תעריף (Price) מחיר לטיפול. ע"פ שדה זה מחושב ללקוח עלות הטיפול, כאשר עלות הטיפול ברוב המקרים מתחלקת בין הגורם המממן ללקוח.

במקרה של עליית מחיר הטיפול נצטרך לעדכן את השדה במסד הנתונים , או ליצור חלון חדש של EditUser שבו נעדכן את התעריף למטפל ובכך העדכון יועבר אוטמטית לכל רמות התוכנה – למחיר ללקוח ולגורם המממן.

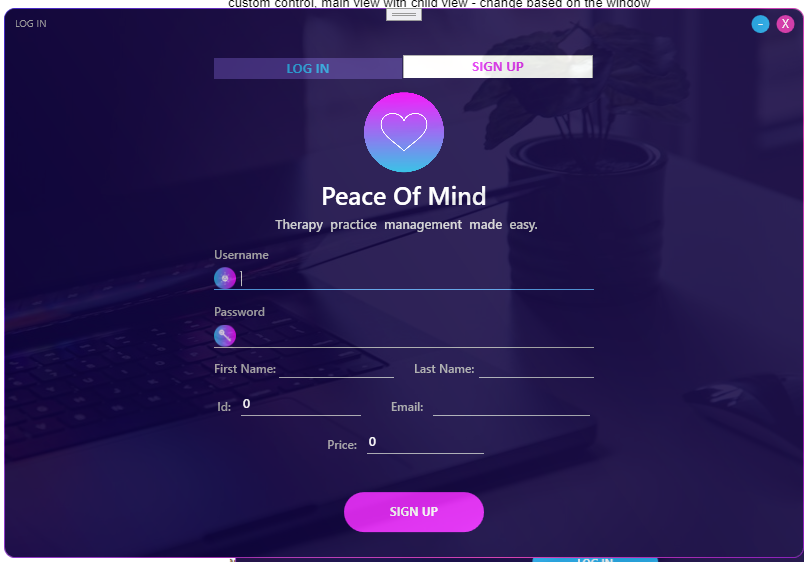
# 10. מדריך למשתמש

בפרק זה נציג את צילומי המסך של החלונות בתוכנה – נתמקד בדוגמאות חיות של לקוחות, גורמים מממנים , פגישות ודוחות להצגת הבנה רחבה של אופן השימוש בתוכנה.

במהלך העבודה על הפרויקט הושקע מאמץ רב ביצירת חווית משתמש איכותית , תוך הקפדה על קו עיצוב נקי ונוחיות המשתמש

בהנאה!

## 10.1 רישום משתמש חדש למערכת

חלון זה מיועד לרישום משתמש – מטפל/ מנהל הקליניקה , מבוצע פעם אחת עבור כל לקוח.

## 10.2 התחברות משתמש קיים למערכת

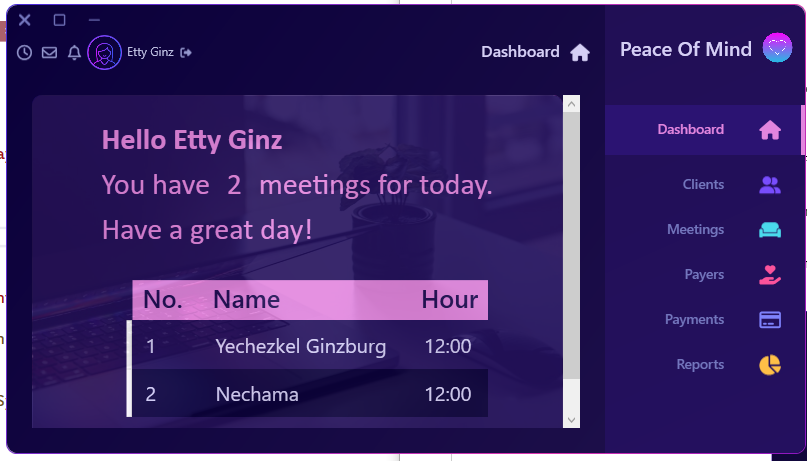
A screenshot of a login screen

Description automatically generated

## 10.3 מסך הבית של התוכנה

מסך זה מוצג לאחר ההתחברות למערכת (התחברות מחדש או רישום ראשוני)

וכן ניתן להציגו בעת לחיצה על מקש dashbord



## 10.4 תצוגת לקוחות במערכת

זוהי טבלה המציגה את פרטי הלקוחות במערכת (במקרה הזה עטי ואילה מפתחות הפרויקט)

A screenshot of a computer

Description automatically generated

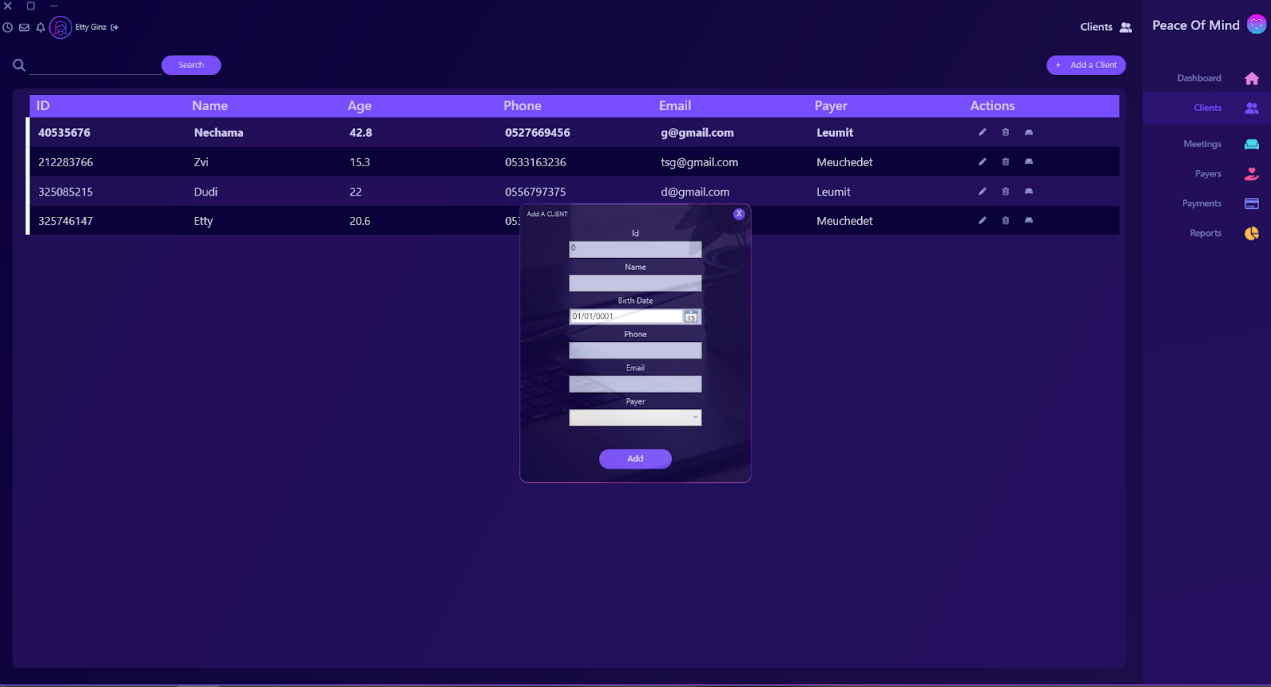
### 10.4.1 עריכת פרטי לקוח קיים

עריכת פרטים אישיים של לקוח קיים, חשוב לשים לב כי חלק מהשדות אינם ניתנים לעריכה כמו שם, ומספר זהות אלא הם לקריאה בלבד.

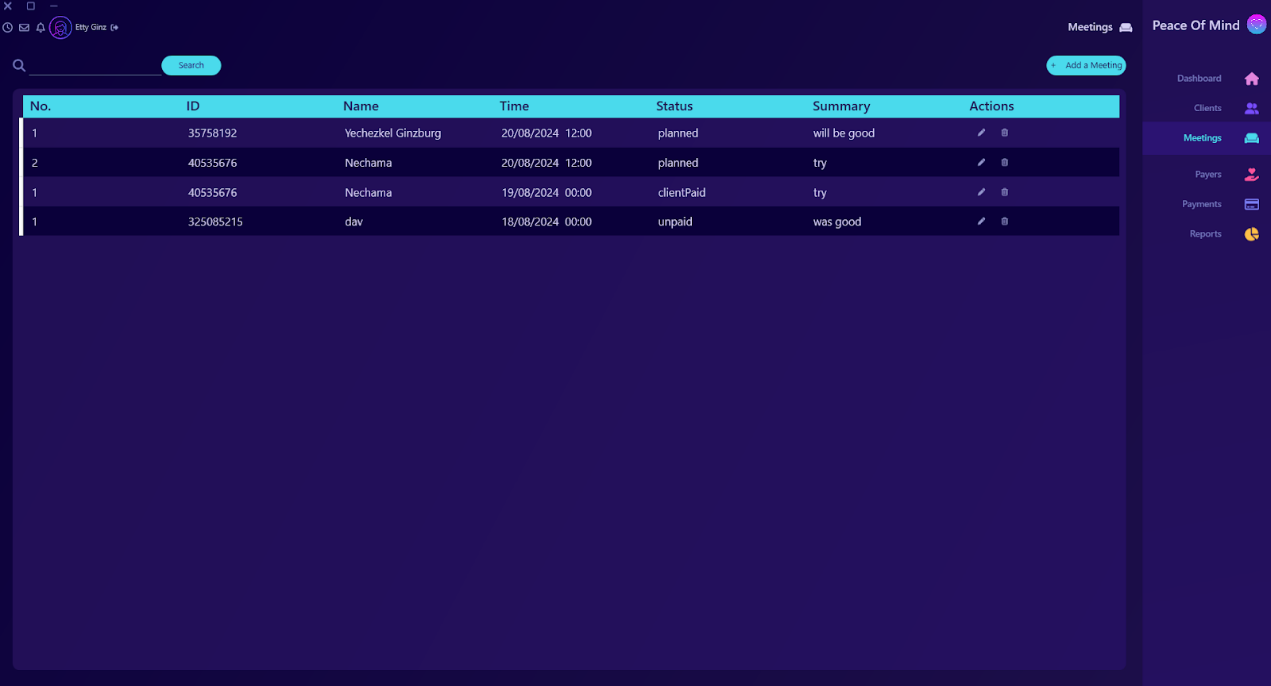
A screenshot of a computer

Description automatically generated

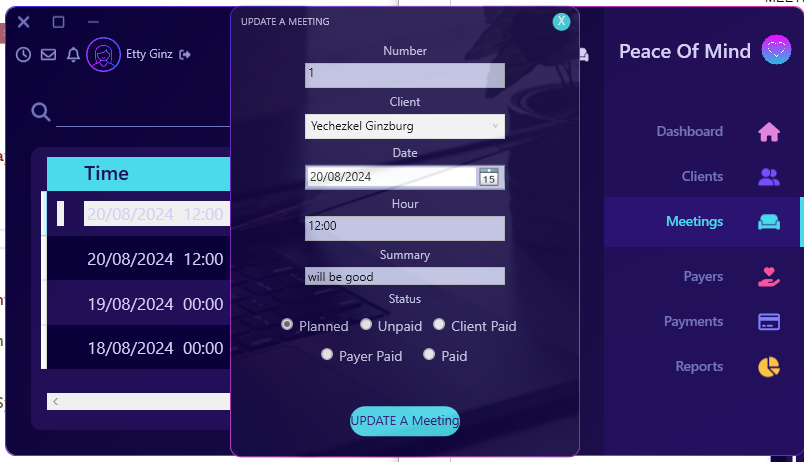
### 10.4.2 הוספת לקוח חדש למערכת



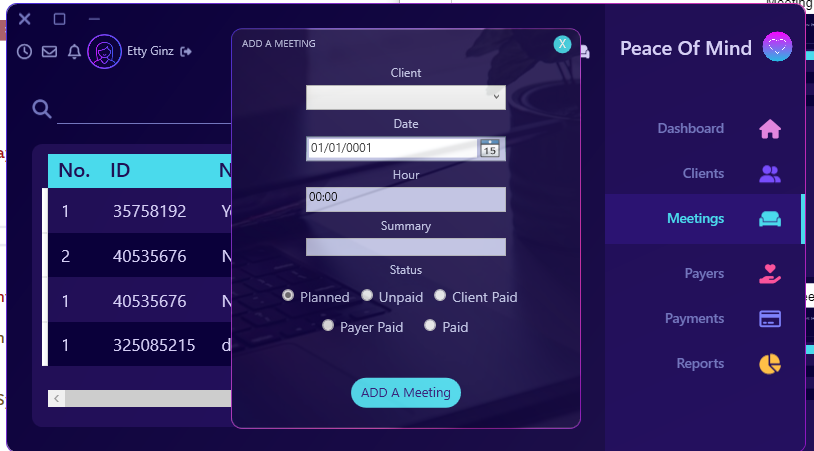
## 10.5 תצוגת פגישות בקליניקה



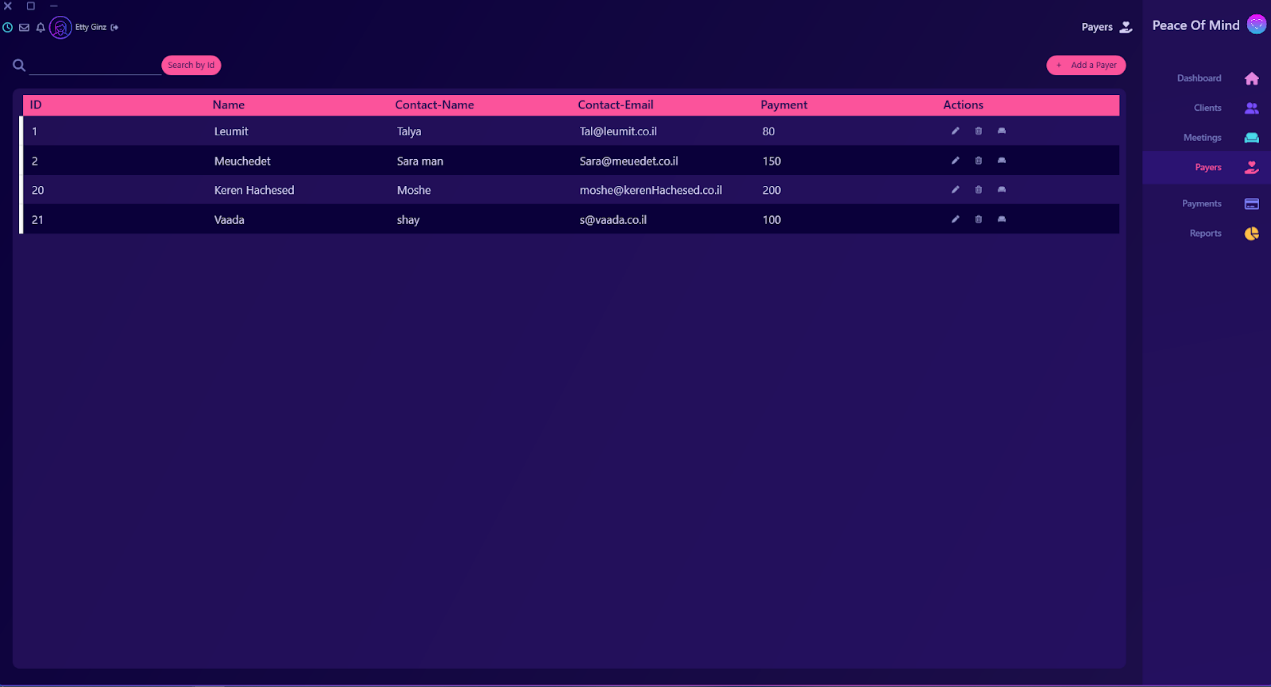
### 10.5.1 עריכת פרטי פגישה



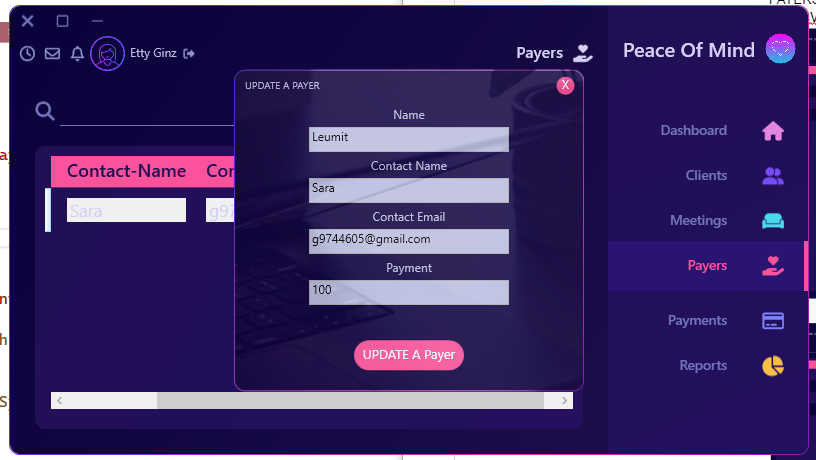
### 10.5.2 הוספת פגישה חדשה



## 10.6 תצוגת גורמים מממנים



### 10.6.1 עריכת פרטי גורם מממן



### 10.6.2 הוספת גורם מממן חדש

A screenshot of a computer

Description automatically generated

## A screenshot of a computer

## 10.7 תצוגת דרישות לתשלום

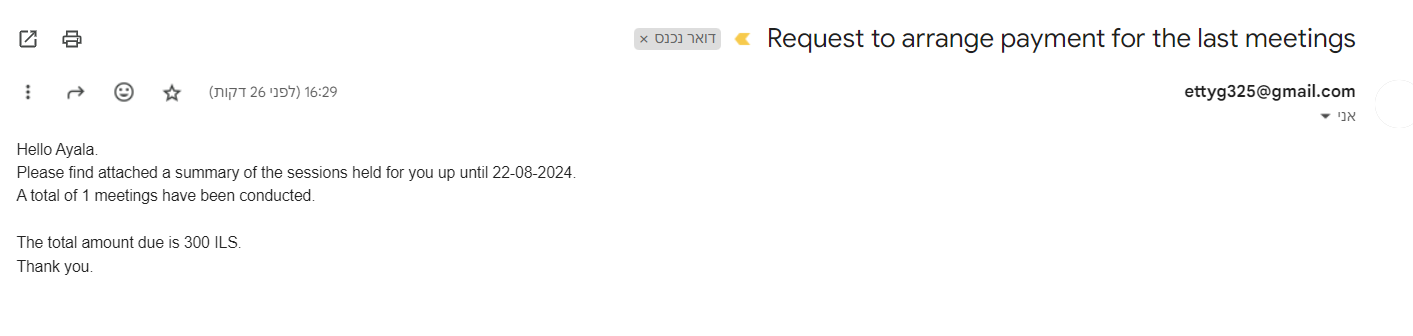
### 10.7.1 הוספת דרישה חדשה לתשלום

כאשר נוספת דרישה לתשלום חדשה למערכת אוטמטית נשלח אימייל ללקוח ובו רשום מהו סכום החוב שלו לקליניקה בגים טיפוליו בה עד כה.

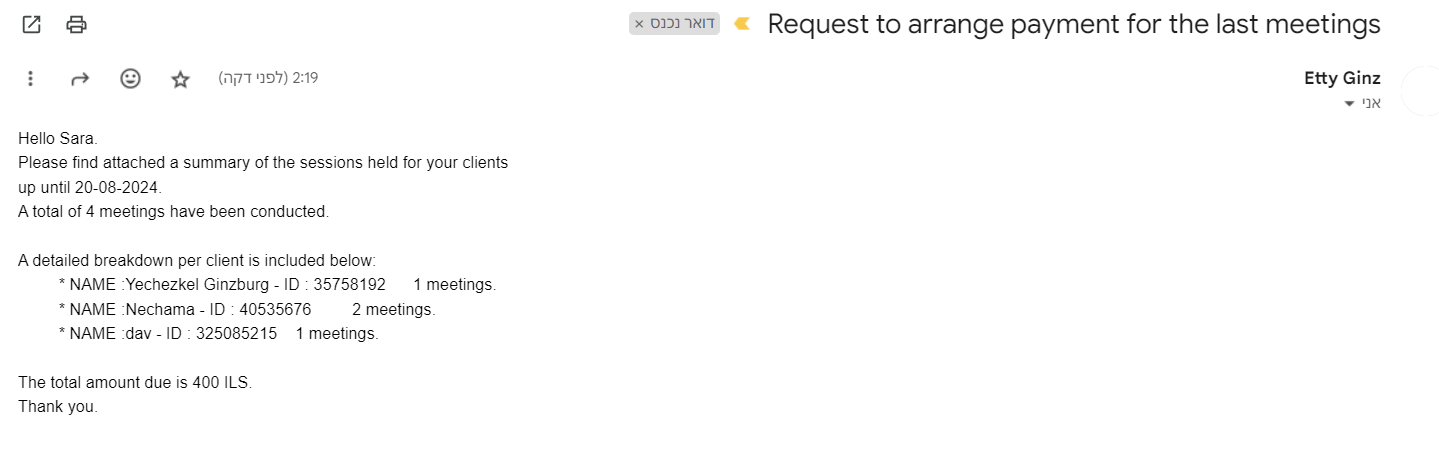
A screenshot of a computer

Description automatically generated

אימייל ללקוח (אילה) המודיע לו על דרישה לתשלום עבור הפגישות בקליניקה. במקרה זה פגישה אחת:



אימייל לאיש הקשר של גורם מממן – שרה אשת הקשר של קופת חולים מאוחדת המודיע לה על דרישה חדשה לתשלום, בפרוט הדרישה לתשלום נכתב מיהם הלקוחות הממומנים ע"י מאוחדת, כמה פגישות כל אחד מהם נפגש, ומהו סכום החוב הכללי של מאוחדת לקליניקה עבור 4 פגישות.



## 10.8 תצוגת דו"חות

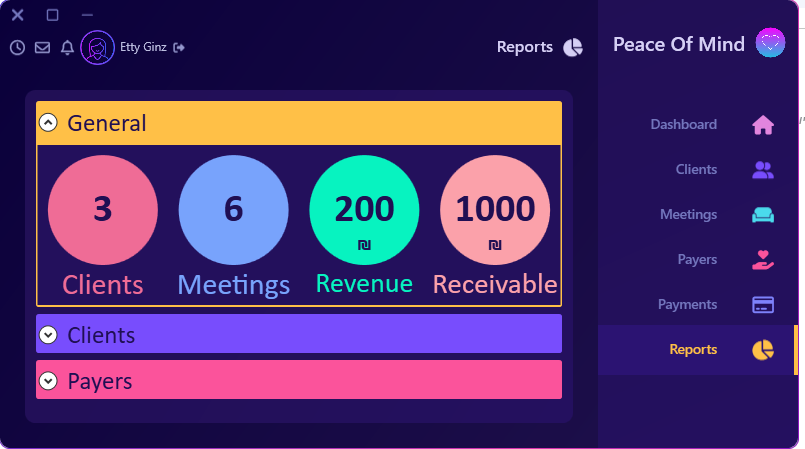
תצוגה זו הינה לצפיה בלבד, לא ניתן לערוך בה פרטים.  
מטרת תצוגה זו הינה לתת למנהל הקליניקה מבט רחב על הכנסות ולקוחות שיש לקליניקה, וסדר – בסוף חודש למי לשלוח בקשת תשלום, כי הצטבר לו חוב. ובכך לגרום לניהול טוב יותר.

### 10.8.1 סיכום כללי אודות הקליניקה

דו"ח זה מציג מצב הכנסות וסיכום לקוחות פעילים ופגישות עבור הקליניקה.

כפי הנראה בדו"ח זה עד כה התקבלו משישה לקוחות תשלומים של 200 שקלים, וצפוים להתקבל עוד תשלומים של 1000 שקלים, כאשר יסודר התשלום לכל הפגישות שנערכו כבר.

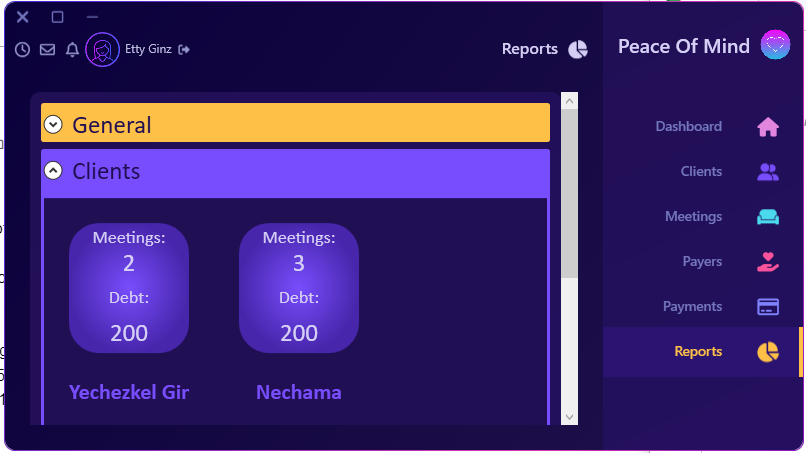
6 פגישות בסך הכל – לא משנה הסטטוס.



### 10.8.2 דו"ח מצבת לקוחות

כפי הנראה בדוח זה ליחזקאל יש 2 פגישות (בכל סטטוס) בקליניקה והחוב שלו הינו 200 שקלים.

ואילו לנחמה יש 3 פגישות במערכת והחוב שלה למערכת הינו 200 שקלים.



### 10.8.3 דו"ח גורמים מממנים

כפי הנראה מדו"ח זה לאומית מממנת טיפולים עבור שלושה לקוחות בקליניקה, כאשר סכום החוב של לאומית לקליניקה בגין השתתפות בטיפוליהם של מבוטחיה הינו 400 שקלים.

