design_patterns

:MVVM

העבודה שלנו מחולקת לשלוש שכבות, בשכבה העליונה Presentation Layer, השתמשנו ב-mvvm pattern, ניתן לראות ש"חילקנו" את השכבה העליונה לשלוש תת שכבות. כמו הנלמד בכיתה והשימוש בתבנית.

כלומר חילקנו את ה Presentation Layer, לשכבות כך שכל אחת דומה לשכבה המתאימה לה במודל השכבות.

Model: השכבה "התחתונה", בשכבה זו מימשנו את כל האובייקטים הנדרשים ובעצם את כל הDAL של הPresintation Layer, כך שנוכל לייצר ממשק משתמש שמכיל את כל הפונקציונליות של השכבות מתחתיו. אובייקטים אלו משקפים את האובייקטים הנמצאים בשכבתה Service, בנוסף תיחזקנו controller, שמדבר עם השכבה התחתונה יותר כלומר עם הService, ובכך מעצם מתנהלת כל התקשורת מול השכבה שלמטה.

כמו מודל השכבות, לשכבות העליונות יותר אין שום קשר למימוש ולתחזוקה של שיכבה זו, כלומר שיכבה זו פועלת באופן עצמאי תוך כדי תקשורת עם השכבה מתחתיה.

ViewModel: שכבת ה Business Layer, בשכבה זו נמצאת הפונקציונליות העיקרית שקשורה ל-Presentation Layer, שכבה זו פועלת בתקשורת מול שכבת הModel.

View: השכבה מולה מתממשק המשתמש, בשכבה זו מימשנו את כל ממשק המשתמש. בשכבה זו אין פונקציונליות בכלל והיא רק מקושרת לשכבת הViewModel.

בעצם ע"י פיצול שכבת ה – Presentation Layer, לשלוש תת שכבות יצרנו מערכת עם coupling & cohesion, טובים יותר. הקוד יותר קריא וברור להבנה. ואיכותו ברמה גבוה יותר.

- לאחר התייעצות מרובה עם המתרגלים ועם אחיה הוחלט לא להשתמש ב-singleton Pattern, מכיוון ששימוש בתבנית זו מקבע את המערכת מבחינת שינויים עתידיים.