**MVP**

**ארכיטקטורה לפיתוח אפליקציות**

פיתוח תוכנה היא משימה מאתגרת הדורשת מחקר, תכנון קפדני, קידוד, הרצות ניסיון, בדיקת שגיאות ועוד.

כידוע דרישות התכנה משתנות עם הזמן ודורשות התאמות לצרכים החדשים כמו גם טכנולוגיות חדשות המאפשרים פיתוחים חדשים ומתקדמים.

חלק גדול מהתוכנות מאפשרות ריצה על פלטפורמות שונות, כגון אתרי אינטרנט, טלפונים חכמים, אפליקציות חלונאיות ועוד.

תוכנות מסוימות הדורשות שימוש במסדי נתונים חייבות תחוקה של מסד הנתונים ויתכן גם שינוי מסד הנתונים עקב כמות המידע הגדלה וזמני עיבוד מתארכים.

אם נבחן לעומק את הנאמר עד כה ניתן לתמצת באופן מאוד פשטני כי תוכנה עשויה לרוץ על פלטפורמות שונות הדורשות התייחסות שונה לאופן הצגת הנתונים והאינטראקציה עם המשתמש וכן שימוש במסדי נתונים שונים עם התפתחות התוכנה.

נשאלת השאלה מה לגבי המידע שהתוכנה עושה בו שימוש ומה לגבי הלוגיקה שלה.

האם אלו חלקים המשתנים עם החלפת האמצעי לאינטראקציה עם המשתמש או שינוי מסד הנתונים ?

ניקח דוגמא פשוטה להבהרת העניין: תוכנה לניהול חשבונות בנק.

לכל בנק יש כיום תוכנה לניהול חשבונות הבנק של לקוחותיו. נתונים בסיסיים כגון פרטי הלקוח, מספר חשבון הבנק, פעולות שונות המתבצעות בחשבון והיתרה הקיימת בחשבון הן הבסיס שעל פיו פועלת הלוגיקה של התוכנה.

בסניף הבנק הפקיד מבצע פעולות ומסתכל על הנתונים באפליקציה חלונאית או אינטרנט פנים בנקאי. הלקוח יכול להסתכל על הנתונים שלו וביצוע חלק מהפעולות באופן עצמאי באמצעות אתר האינטרנט של הבנק וכשהו בדרך באמצעות הטלפון החכם.

אנו רואים כי הנתונים והלוגיקה של התוכנה נשארת אותו דבר ואיננה תלויה באמצי האינטראקציה עם המשתמש.

ברבות השנים הבנק גדל וכן גם הפעילות העסקית ואמצעי אחסון הנתונים לא עומדים בעומס דבר הדורש את החלפת מסד הנתונים.

שוב גם כאן, החלפת מסד הנתונים איננה משפיעה על הלוגיקה של התוכנה והנתונים עליה היא מתבססת.

מכאן אנו למדים כי תוכנה בנויה מחלקים שונים המהווים פיתרון לבעיה מסוימת, במקרה שלנו ניהול חשבון בנק. אנו מבינים שחלקים אלה אינם תלויים בחלקים אחרים לצורך פעולתם והקשר היחיד ביניהם נועד לפתור את הבעיה הכוללת.

הדרך הנכונה לכתיבת תוכנה היא לדאוג לכך שכל חלק יהיה עצמאי לחלוטין שאיננו תלוי בחלק אחר – הפרדה מוחלטת בין החלקם הנ"ל.

חלוקה זו עשויה להקל על כתיבת התוכנה, עשוי להקל מאוד בבדיקת החלקים השונים שלה התוכנה, תיקון שגיאות ותחזוקה שוטפת לאורך השנים.

עם השנים התפתחו ארכיטקטורות שונות לפיתוח תוכנה כך שיביא לתוצאות הטובות ביותר בפיתוח תוכנה לסוגי תוכנות שונות. כולן באות להפריד את חלקי התוכנה השונים.

המוכרים ביותר כיום:

MVC – Model View Controller.

MVP – Model View Presenter.

MVVM – Model View ViewModel.

הארכיטקטורה העדיפה כיום לפיתוח אפליקציות באנדרואיד הינה MVP.

**צמידות** – מושג המבטא את דרגת הקשר בין חלקי תוכנה שונים.

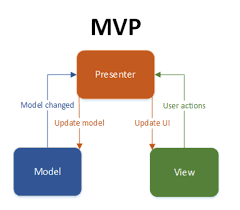
ככל שרמת הצמידות נמוכה יותר כך התוכנה כתובה בצורה טובה יותר.

**Model View Presenter**

**Model** – נתוני המערכת (כולל נתונים במסדי נתונים) והלוגיקה של המערכת.

**View** – תצוגת המערכת (באנדרואיד קבצי XML ו-Activities). מנתב אירועים (כגון לחיצה על כפתור) ל-Presenter לביצוע מול ה-Model. מציג נתונים שהגיעו מה-Model.

**Presenter** – מקשר בין ה-Model ל-View.

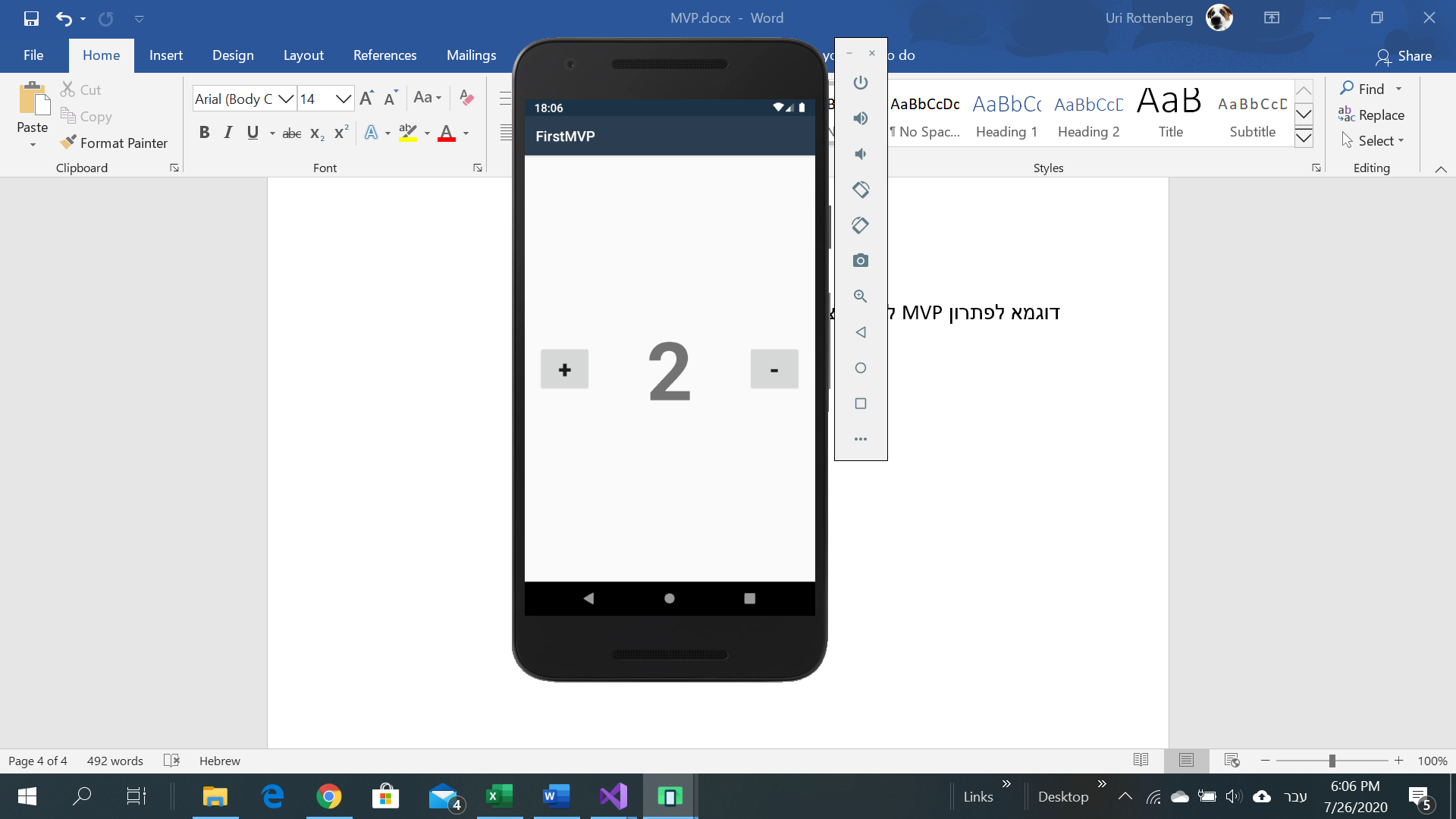


על מנת לממש את ה-MVP הצורה מיטבית כך שלא יהיה כל קשר בין ה-View

ל-Model נשתמש בתכונות OOP באופן העמוק ביותר (שימוש במחלקות וממשקים) כדי להגיע לרמת צמידות נמוכה.

**דוגמא פשוטה**

דוגמא לפתרון MVP לאפליקציה פשטוה המעלה ומורידה מספר.



IPresenter

Increment()

Decrement()

IView

DisplayNumber()

Class Number

Num

Increment()

Decrement()

MainActivity

Presenter

btnIncrement.Click()

btnDecrement.Click()

DisplayNumber()

Class Presenter

Number

View

Increment()

Decrement()

**1**

**2**

**3**

**4**

**5**

המטודה Increment במחלקה Presenter

public void Increment()

{

number.Increment();

view.DisplayNumber(number.Num);

}

**הסבר**

1. המשתמש לוחץ על הכפתור "+".

המטודה btnIncrement.Click() מופעלת.

המטודה Increment במחלקה Presenter מזומנת.

1. Presenter->Increment()

המטודה Increment של המחלקה Number מזומנת.

1. הערך של num במחלקה Number עולה ב-1.
2. (המשך מסעיף 2)

המטודה DisplayNumber מזומנת (באמצעות המימשק) ומועבר הערך של num.

1. המטודה DisplayNumber ב-MainActivity

מציגה את הערך החדש.

**מבנה פרויקט**

פרויקט ב- Visual Studio ניקרא Solution – פתרון לבעיה.

Solution מורכב מאוסף פרויקטים Project.

קיים קשר בין הפרויקטים (Reference).

מבנה זה מתאים לשיטת הצמידות הנמוכה.

מבנה זה מאפשר להחליף בקלות את רכיב התצוגה ורכיב שכבת הנתונים (מסד נתונים).

**שכבת העזרים**

**(מטודות שרות)**

**שכבת הלוגיקה**

**(מחלקות)**

**שכבת הניהול**

**)Model <-> View(**

**שכבת התצוגה**

**(Activities + XML)**

**שכבת הנתונים**

**(מסד נתונים)**

**HELPER**

**VIEW**

**PRESENTER**

**MODEL**

**DAL**