בס"ד

**מפרט ארכיטקטוני:**

הארכיטקטורה של הפתרון המוצע בפורמט של level Down-Top :

הפרויקט מחולק לשני חלקים- לצד השרת וצד הלקוח:

1.צד שרת שנכתב בשפת #C בטכנולוגית WebApi .

2.צד לקוח שנכתב בשפת react.

**תיאור צד שרת(server):**

צד השרת מחולק כמקובל לשכבות:

שכבת **Core-** .

שכבת **Service**- .

שכבת **API -Presentation-** .

החלוקה לשכבות נועדה להפריד באופן מוחלט בין הלוגיקה של הפרויקט לבין הנתונים עצמם.

הפרדה זו מאפשרת לבצע שינויים בכל אחת מהשכבות בלי תלות ובלי זעזועים בשכבות האחרות.

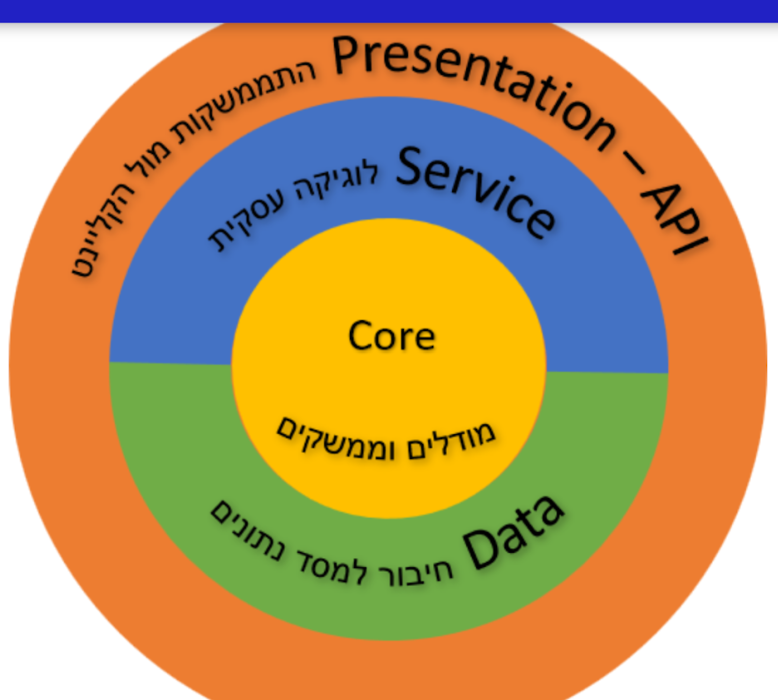
פירוט:

**: Core** שכבת ה **Core** היא ליבת המערכת. בה נמצאות הגדרות מבני הנתונים והגדרות הפונקציונליות שכבת ה Core-לא מכילה מימוש כלל. כל השכבות מכירות בה והן אלו שממשות בפועל את הממשקים וההתנהגויות שהיא מגדירה . פעולות ההתקשרות עם בסיס הנתונים נעשו בטכנולוגית Entityframwork, טכנולוגיה זו היא המקובלת לטיפול בבסיס הנתונים כאשר היא מנתחת את בסיס הנתונים, בונה מחלקות לייצוג הטבלאות ומספקת רשימות מלאות בנתוני בסיס הנתונים.

**Service:** שכבה זו מכילה קוד שמבצע משימות לוגיקה עסקית, כגון כללים עסקיים, חישובים מורכבים, אימות כללי, וכן מתקשר עם שכבת הגישה לנתונים.

**API -Presentation**: שכבה זו מכילה מחלקות המתארות את הנתונים ובמבנה זה מעבירים את הנתונים בין השכבות. מטפלת בכל מה שקשור לתצוגה ולהיפגשות עם המשתמש. כאן ממוקמים האלמנטים.

בזכות ההפרדה בין השכבות, ניהול התלותיות בין הקוד הופך לקל יותר. כל שכבה מוכרת רק לשכבות אחרות ספציפיות, וכך ניתן לנהל את התלותיות בצורה קלה יותר. מודל זה אמנם גורם טרחה טכנית לא מעטה בכתיבת הקוד ובתכנון הפונקציות אבל מספק קוד נקי, קל להבנה ונוח לשינויים ולשדרוגים.

­

**תיאור קישרי גומלין:**

**­**

