מפרט ארכיטקטוני

הפרויקט מחולק לשני חלקים- צד השרת וצד הלקוח:
צד שרת שנכתב בשפת C# בטכנולוגית Net Core.
צד לקוח שנכתב בשפות Html , css, type script ובטכנולוגית Angular.

צד שרת:
הקוד ב Dot.Net מאורגן בתוך solution שמכיל את כל הקוד של המערכת, ובתוכו אפשר להוסיף פרויקטים רבים. בין הפרויקטים קיים קשר והם יכולים להשתמש זה בזה.
נארגן את הקוד בפרויקטים נפרדים בתוך ה-solution כל פרויקט מכיל קוד עם תפקיד מסוים והוא בעצם שכבה אחת של המערכת. ארגון של הקוד בשכבות נועד כדי לאפשר ניהול טוב יותר של הקוד, יכולת לעדכן חלקים ממנו בלי שכל העדכון ישפיע על כל הקוד כולו ועוד.

שם ה-solution הוא שם הפרויקט HMO
בתוך ה-solution יש לנו את הפרויקטים הבאים:

1. HMO.API – מסוג Web Api ובו ה Api שחשוף החוצה ומספק לצד לקוח גישה לפונקציות של שליפת נתונים, עדכון, הוספה ומחיקה וכל מה שנדרש.
2. HMO.Core - מסוג Class libery ובו מודלים וממשקים של כלל המערכת.
3. HMO.Data – מסוג Class libery שמתקשר עם מקור הנתונים, מחזיק את הנתונים ומטפל בפעולות שליפה, עדכון, הוספה ומחיקה.
4. HMO.Services - מסוג Class libery ובו הלוגיקה העסקית.

הערה:
בפרויקט זה התלות היא מחוץ לפנים, כלומר: שכבת ה API-תלויה ב Service וב Data שכבות IService ו-Data- מקבילות זו לזו ואינן תלויות אחת בשניה.
בנוסף, כל השכבות תלויות ב Core-
שכבת ה Core היא ליבת המערכת. בה נמצאות הגדרות מבני הנתונים והגדרות הפונקציונליות.
שכבת ה Core לא מכילה מימוש כלל.
כל השכבות מכירות את שכבת ה Core- והן אלו שמממשות בפועל את הממשקים וההתנהגות שהיא מגדירה.

פירוט:

HMO.WebAPI

פרויקט הWeb Api הוא השכבה שחשופה החוצה ומקבלת את בקשות צד הלקוח.
נארגן את הקוד בכמה תיקיות:

<u>controllers</u>
בתיקיה זו יש class ים מסוג Controller שהם בעצם הendPoint ים שחשופים החוצה וניתן לשלוח אליו קריאה באמצעות ניתוב ייחודי משלו.

<u>Models</u>
בתיקיה זו יהיו את המודלים המיועדים לקריאות post ו- put בלבד.

<u>Mapping</u>
מחלקה זו אחראית על ההמרות של אובייקט ממחלקה "רגילה"- תחת entities שב- Core לאובייקט ממחלקה של Models וכן להפך (בעזרת AutoMapper).

HMO.Core

בפרויקט Core ננהל את המודלים והממשקים של כלל המערכת.
נארגן את הקוד בכמה תיקיות:

Entities
בתיקיה זו יהיו ה-CLASSים שמחזיקים את הנתונים. כל Class ממופה לטבלה אחרת
ובו properties שתואמים לשדות הטבלה.
ה Entities הם מחלקות פשוטות שמגדירות את מבני הנתונים בלבד ולכן יש בהם רק properties
הפונקציונאליות של קישורים ממסד נתונים מוגדרים במחלקות נפרדות שנקראות Repository.

DTOs
בתיקיה זו יהיו ה-CLASSים המחזיקים את הנתונים, אך לא עם כל השדות כמו שיש ב-CLASSים שב-
Entities אלא רק עם השדות שארצה שיוצגו למשתמש.

Mapping
מחלקה זו אחראית על ההמרות של אובייקט ממחלקה "רגילה"- תחת entities לאובייקט ממחלקה של
DTO וכן להפך (בעזרת AutoMapper).

Interfaces:
בשביל שנוכל להשתמש בהזרקות קוד לצורך גמישות המערכת אנו יוצרים לכל מחלקה פונקציונלית
ממשק ובמהלך כל הקוד נשתמש בטיפוסי ממשק, בבנית האפליקציה נזריק את הסרברים המתאימים.
בתיקיה זו שמורים הממשקים של המחלקות Services, Repository :

Repository
Interface ובהם כותרות של פונקציות לקישור עם מסד נתונים כמו שליפה, מחיקה , עדכון, הוספה וכו'.

Services
Interface ובהם כותרות של פונקציות האחראיות על הלוגיקה העסקית.

HMO.Data

שכבה זו אחראית על התקשורת עם מקור הנתונים,
וכן יש בה תיקיית Repository ובה מחלקות הממשות את הפונקציות שב Interface שבשכבת ה
Core(בתיקית Repository).

HMO.Services

אחראית על הלוגיקה העסקית.
בשכבה זו קיימות מחלקות פונקציונאליות שמבצעות את הלוגיקה העסקית בפרויקט.

בפרויקט זה הקוד ממומש בצורה אסינכרונית.