צד לקוח:

1. הצגת כל החברים הרשומים בקופה. (מטבלת חברים).
2. הוספת חבר למערכת. (לטבלת חברים).
3. עדכון חבר.
4. הצגת נתונים של חבר מסוים בלחיצה עליו מהרשימה:

* מועד קבלת חיסוני הקורונה שלו. (מטבלת התחסנות).
* מועד בו היה חולה ומועד החלמתו. (מטבלת חולי).
* אפשרות מחיקה.

צד שרת:

1. חברים:

* ~~הוספה.~~
* מחיקה (סטטוס= שקר).
* Get. (אפשר גם רשומים או מחוקים).
* עדכון.

1. ערים:

* Get. (השאר לא חייב)

1. התחסנות:

* בעקרון רק Get. השאר בווס.

1. חולי:

* בעיקרון רק Get. השאר בונוס.

1. סוג חיסון:

* בעיקרון רק Get.

רציתי שלחבר, לא יהיה מספור אוטומטי כי ת.ז. תהיה מפתח ראשי. הענין הוא, שאנינ רוצה לאפשר עדכון של ת.ז ואם אני אאפשר את זה כך, נתוינם הקשורים לאובייקט שעודכן לא יזוהו, כי עדכון של מפתח ראשי אפשרי על ידי מחיקת הרשומה והוספת חדשה מעודכנת, לא על ידי עדכון רגיל. ועל כן: אני אוסיף עמודה נוספת של קוד וזהו..

**מסד נתונים:**

מסד הנתונים הנבחר הוא SQL SERVER, הישויות בו הן: חברי קופת חולים, תקופות חולי, התחסנויות, סוגי חיסון וערים.

מסד הנתונים נוצר ע"י הרצת שאילתה הבונה מסד נתונים, יוצרת טבלאות לפי שדות ואילוצים מסוימים. לבסוף, אתחול מסד הנתונים בנתונים התחלתיים.

**צד שרת:**

צד השרת נוצר בשפת C# במודל 3 השכבות בentity framework. שלב ההתחברות למסד הנתונים הוא די פשוט: הורדות הספריות SQL SERVER, TOOLS והרצת פקודת התחברות לפי שרת, שם מסד נתונים וכו'. חלקתי את צד השרת לשכבת DAL בה התבצע חיבור למסד נתונים וגישה ישירה אליו. שכבת BLL בה מתרחשות פעולות לוגיות שונות, שכבה זו תלויה בשכבת הDAL. בנוסף, בשכבה זו יצרתי העתק של מסד הנתונים ללא קשרי גומלין לנוחות המשתמש. לפיכך, הצרכתי להשתמש בממיר AUTO MAPPER ההופך את ישויות מסד הנתונים לישויות החדשות ללא קשרי הגומלין. בשכבה זו הוספתי שדות לטבלאות מסוימות כדי שבשליפה אחת לא אצטרך לשלוף טבלה נוספת (לצורך שימוש במפתח זר), והבאתי את הנתונים הבסיסיים לישויות החדשות (גישה למסד נתונים לוקחת זמן רב). שכבת CONTROLLERS התלויה בDAL ובBLL המתקשרת עם המשתמש ומאזינה לבקשות שרת מסוג GET, POST, PUT DELETE.

בפרויקט כולו נעשה שימוש בהזרקת תלויות: במקום להיות תלויים בפונקציות מסוימות תלויים ביכולת מסוימת. מאפשר גמישות ושינוי בקלות יותר בעתיד. בפרויקט כולו הממשקים גנריים, כך שאין קוד כפול, והוא מתאים לכל הישויות.

**צד לקוח:**

ברגע שמוחקים חבר, נמחקת גם הכתובת שלו. הוא עצמו נהפך לFALSEכי המטרה היא לשמור מידע על ההתחסנויות והחולי שלו. היכן גר פחות משנה.

הייתה לי תלות מעגלית:

public class FuncPaddressBLL : IPaddressBLL

public interface IPaddressBLL : ICRUDBLL<PaddressDTO>

public interface ICRUDBLL<T> where T : class

public class FuncPatientBLL : IPatientBLL

public interface IPatientBLL : IRemovableBLL<PatientDTO>

public interface IRemovableBLL<R> : ICRUDBLL<R> where R : class

public interface ICRUDBLL<T> where T : class

הייתה לי תלות: ב FuncPaddressBLL היה לי תלות ב IPatientBLL לצורך List<PatientDTO> p = \_IPatientBLL.GetPatientsByAddress(code);

וב FuncPatientBLL היה לי תלות ב IPaddressBLL לצורך if (!\_IPaddressBLL.RemoveObject(p.AddressCode))במקרה של מחיקת חבר- מוחק את הכתובת שלו.

לפיכך יצרתי ממשק ביניים הפותר את בעיית התלות המעגלית יותר תלות באופן עקיף:

ביטול!!

בעדכון כתובת, אם יש 2 על אותה כתובת, נוצרת כתובת חדשה, חשוב!

מחיקת כתובת מותרת רק אם יש חבר אחד עם הכתובת הזאת. עכשיו בעיקרון מיותר, כי לעולם לא תהיה מחיקת כתובת

בעדכון חבר, גם אם שינה את כתובתו הכתובת הישנה נשארת.

במחיקת חבר, גם הכתובת שלו נשארת.