**הגדרות דרישות ותיאור כללי**

**מטרת המערכת:**

מטרת המערכת היא להפוך, בלחיצה אחת, כל מתכון או תפריט לרשימת קניות חכמה.

המערכת מאפשרת למשתמש לשמור מתכונים או להוסיף אותם לתפריט המזון השבועי שלו ויוצרת באופן מידי רשימת קניות במכולת.

**היקף העבודה:**

מספר השעות שהוקדשו עבור פרויקט זה עומד על כ- 750 שעות.

**תיאור חומרת המערכת:**

מחשב PC / מכשיר נייד בעל דפדפן וחיבור לרשת.

**תיאור תוכנת המערכת:**

צד שרת נעשה בשפת #C ,שפה מתקדמת ומלאת פונקציונאליות.

צד לקוח נעשה בשפת Angular8 ,שפה פופולארית הנוחה מאוד למתכנת ולמשתמש.

מסד הנתונים נעשה באמצעות Server SQL בטכנולוגית Framework Entity.

**מדריך למתכנת:**

**תיאור המערכת:**

אתר המאפשר למשתמש המעוניין לצאת לקניות במכולת...

**ממשק המערכת:**

**באתר נבנה כמה וכמה ממשקים:**

1. ממשק ראשי הכולל:

* אודות המערכת ?
* מעלות המערכת ?
* מתכונים מומלצים על בסיס הפופולאריות שלהם.

1. ממשק התחברות:

* כניסה – למשתמש רשום.
* כניסה למשתמש חדש ע"י רישום למערכת.

1. ממשק אזור אישי הכולל:

* מתכונים מועדפים.
* לוח תכנון ארוחות.
* רשימת קניות

1. ממשק חיפוש מתכון/הוספת מתכון:?

נדרש לחפש מתכון במאגר(שנשלף מהAPI ע"י מילת החיפוש שהמשתמש מקליד) ולהגדיר עבורו תדירות ותאריך הכנה ?

1. ממשק כללי:?

* דף מתכון

**תכנות**

**תיאור**

המערכת נכתבה בשפת C#.NET בסביבת Visual Studio 2019 ועושה שימוש בטכנולוגיות מתקדמות, בתוכם: Web API, Angular8, Entity Framework וכן, בספריית Bootstrap.

כאשר צד ה-Server פותח בטכנולוגיית Web API בשיטות המתקדמות ביותר ש-NET מציע וצד ה-Client פותח באמצעות Angular8.

**טכנולוגיות מתקדמות**

1. API Web – ממשק תכנות יישומים לשרתי אינטרנט ו/או לדפדפני אינטרנט. מטרתו "לתווך" בין צד הלקוח(client) לצד שרת(server) על מנת לבצע את הפונקציונליות הנדרשת לצד לקוח. עיקר תפקודו הוא בקבלת נתונים מצד הלקוח והעברתם לשרת ,ולהפך.

קטע קוד לדוגמא: קבלת בקשת GET לקבלת רשימת המתכונים של לקוח מסוים, קבלת בקשת post להוספת מתכון.

1. LINQ – שפת צד שרת, שיטת כתיבה חדשנית מובנית ב-NET , המיועדת לריצה על lists מערכים של אובייקטים וחיפוש בתוכם בצורה פשטנית, חכמה וקלילה.

קטע קוד לדוגמא: שליפת מתכון מסוים ע"י שפת הlinq בצורה ישירה וקלילה, ללא משפטי תנאי מיותרים.

1. Entity Framework – טכנולוגית עבודה מתקדמת מול DB .יצירת אובייקטים ומיפוי מול טבלאות הDB ,כאשר אופן העבודה ועיבוד הנתונים בשפת LINQ כמוזכר לעיל.
2. Angular8 – טכנולוגיה מתקדמת המאפשרת פיתוח מלא מקצה לקצה בצד לקוח, כלומר Angular אספה התנהגויות שונות שעד היום נכתבו בצד השרת ואפשרה אותם בצד הלקוח (services, ingectable ועוד).

Angular תומכת באפשרות ליצור אפליקציות של S.P.A – Single Page Application, כלומר, רק חלק בדף מסוים מתרענן ומתחלף וכל שאר החלקים נשארים אותו הדבר. דבר זה מאפשר גלישה מהירה וחלקה אשר משפרת את חווית המשתמש. בכל קומפוננטה (רכיב המיועד לתצוגה, המכיל את כללי התצוגה ואת הניהול שלה) בה רוצים להציג תוכן כלשהו משתמשים במאפיין – template. מאפיין זה אומר לAngular איזה תוכן HTML להציג עבור קומפוננטה זו, הtemplate מכיר את הקומפוננטה ואת כל השדות והפונקציות הכתובות בה. Angular משתמשת בשיטת ה- tow way data binding,כלומר, כל שינוי הנעשה ב- template משפיע ישירות על הקומפוננטה ולהפך.

1. Typescript – שפה חדשה הדומה לתקן החדש של JavaScript.
2. SweetAlert2, Syncfusion ej2, Angular Material – חבילות components מעוצבות.

**עקרונות התכנות**

1. מבחינה מקצועית – בחירת שפות וטכנולוגיות התואמות את צרכי האפליקציה, התמקדנו בחיפוש שפות וטכנולוגיות מהמתקדמות ביותר, המאפשרות מודולריות ופונקציונליות גבוהה, וכן ברצוננו לרכוש ידע מקצועי מהמובילים בתחום.
2. מבחינת מבנה הקוד – בנינו את הפרויקט בצורה מודולרית ביותר ע"י חלוקה לשכבות (DAL, BL, GUI) בכדי להפוך את הקוד לנהיר ודינאמי כך שמפתחים נוספים יוכלו להבין את הקוד בקלות, להוסיף על הפרויקט, ובמידת הצורך לשנות.
3. מסכי המערכת נכתבו בצורה פשטנית וברורה ביותר לעין, כך שניתנים לשימוש בקלות, ונותנים חווית משתמש .
4. התפיסה הרווחת כיום, לגבי אפליקציות Web היא כמה שיותר קוד בצד ה-client מכיוון שכל פנייה לשרת מכבידה על המערכת וגורמת לחוויית משתמש ירודה. השתדלנו שרוב הנתונים שלא צריכים עיבוד בשרת ימומשו בצד ה-client ,ועל כן, השתמשנו בספריית angular כדי לחסוך פניות מיותרות לשרת.

**חלופות שפות תכנות:**

**שפת** **HTML** -שפה בסיסית שלא מאפשרת הצגת נתונים דינאמיים ומתאימה במיוחד לאתרים פשוטים. השפה מאפשרת למתכנתים לקשר בין מגוון שפות תכנות ולייצר קוד מורכב. בוני אתרים רבים משתמשים בשפה כדי לחתוך ולעצב תבניות ולהציג מידע באינטרנט. כמו כן, ניתן להשתמש בשפה כדי ליצור קישורים לנתונים, כדי להציג טקסטים, תמונות, סרטוני וידיאו ומוסיקה וכדי לבצע מגוון פעולות נוספות.

**שפת Java Script** - הפקודות של JS משפיעות על הדפדפן והמחשב ומקלות על המתכנתים לבנות אלבומי תמונות, קישורים ותפריטים שנפתחים. חשוב לציין כי השפה לא מובנת למנועי החיפוש משום שהיא מתקשרת בין הדפדפן והמחשב ולא עם השרתים, דבר שעשוי לפגוע בתהליך הקידום של האתר.

**MVC** - מקובל לחלק יישום תוכנה למספר שכבות נפרדות: שכבת התצוגה (ממשק משתמש), שכבת התחום העסקי (לעיתים נקראת גם "שכבת הלוגיקה העסקית") ושכבת הגישה לנתונים.

בתבנית MVC שכבת התצוגה מחולקת בנוסף לתצוגה ובקר.

יש המחשיבים את התבנית כתבנית עיצוב, אך בהשוואה לתבניות עיצוב אחרות, MVC עוסקת במבנים בקנה מידה בינוני-גדול ולכן נחשבת גם כתבנית ארכיטקטורה.

**מודל** - המודל הוא ייצוג מסוים, מוכוון תחום עסקי, של המידע עליו פועל היישום. המודל, למרות הדעה הרווחת, אינו שם אחר לשכבת התחום העסקי והוא נפרד ממנה.

תבנית MVC אינה מזכירה במפורש את שכבת הגישה לנתונים, מכיוון ששכבה זו היא מתחת למודל, או נעטפת על ידו .

**תצוגה** – תפקידה להמיר את נתוני המודל לייצוג המאפשר למשתמש לבצע פעולת גומלין כלשהי. לרוב מדובר על המרה לממשק למשתמש כלשהו. תבנית MVC משמשת רבות ביישומי Web ,בהם התצוגה היא דף HTML והקוד אוסף מידע דינאמי לדף.

**בקר** - תפקידו לעבד ולהגיב לאירועים המתרחשים בתצוגה, לרוב, כתגובה לפעולה של המשתמש. בעיבוד האירועים, הבקר עשוי לשנות את המידע במודל, באמצעות תפעול שירותים המוגדרים בו. בקרים מורכבים מתבססים לרוב על יישום של תבנית command.

**לסיכום**:

בחרנו לכתוב את המערכת בטכנולוגיית Angular8 ב- Client side,

C# - Web API ב- Serever side, מול בסיס נתונים Sql Server, בטכנולוגיית Entity Framework.

**בעיות ופתרונן:**

**אבטחה:**

(לא נראה לי שיש לנו בעיות אבטחה בגלל שהנתונים נקלטים מהAPI ואנחנו מעבדים אותם...)

**שרידות המערכת ותיעוד:**

במהלך הפרויקט שמנו דגש על אבטחת הנתונים, לשם כך בחרנו להשתמש בטכנולוגית Web API מכיוון שהשימוש בטכנולוגיה זו מחייב שימוש בפרוטוקול HTTPS .

פרוטוקול זה הינו בטוח לשימוש ומוכר בעולם התכנות.

בשליחת נתונים מצד הלקוח לצד השרת )post )הנתונים מועברים כאובייקט DTO ולא כשרשור גלוי בשורת Url.

**בדיקות תוכנה:**

**טבלת בדיקות מסכמת**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **סוג הבדיקה** | **מס' בדיקה** | **תיאור קצר** | **עבר/נכשל** | **באגים** |
| בדיקת פונקציונאליות | 1 | בדיקה שהמערכת  פועלת כמצופה  מימנה | עבר | 0 |
| בדיקת שימושיות | 2 | בדיקת נוחות  השימוש  וחווית המשתמש | עבר | 0 |
| בדיקת GUI | 3 | בדיקת הזדהות  נכונה וניווט דפים  תקין | עבר | 0 |

**Test case:**

**בדיקות פונקציונאליות:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **בדיקה מס'** | **שלב** | **תוצאה צפויה** | **תוצאה בפועל** | **עבר/נכשל** | **באגים** |
| **1** |  |  |  |  |  |
| **2** |  |  |  |  |  |
| **3** |  |  |  |  |  |

**בדיקות שימושיות:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **בדיקה מס'** | **שלב** | **תוצאה צפויה** | **תוצאה בפועל** | **עבר/נכשל** | **באגים** |
| **1** |  |  |  |  |  |
| **2** |  |  |  |  |  |
| **3** |  |  |  |  |  |

**בדיקות שימושיות:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **בדיקה מס'** | **שלב** | **תוצאה צפויה** | **תוצאה בפועל** | **עבר/נכשל** | **באגים** |
| **1** |  |  |  |  |  |
| **2** |  |  |  |  |  |
| **3** |  |  |  |  |  |