# 

**המכון הממשלתי להכשרה בטכנולוגיה ובמדע**

# דף כריכה

**פרויקט גמר לתואר הנדסאי מחשבים**

**שם הפרויקט: MediCity**

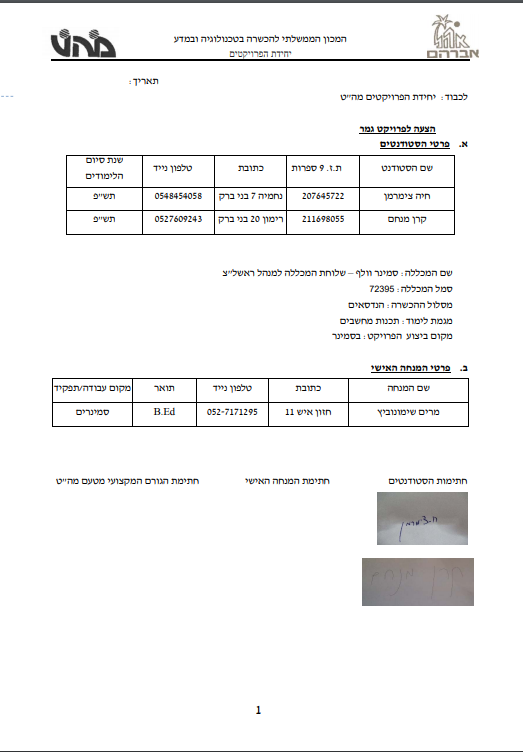
מגישות: חיה צימרמן קרן מנחם

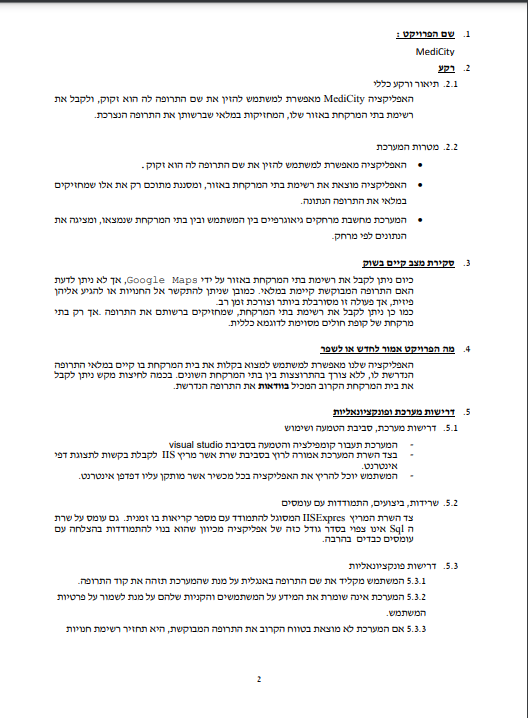
שם המנחה: הגב' מרים שמיעונוביץ

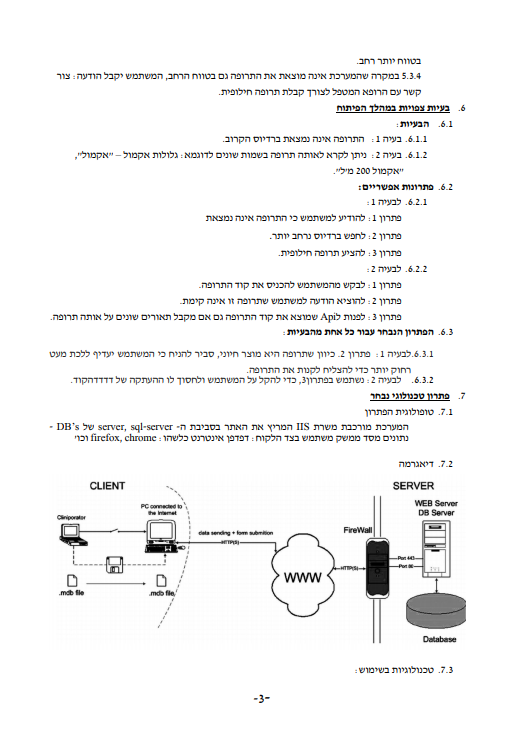
תאריך ההגשה: דצמבר 20

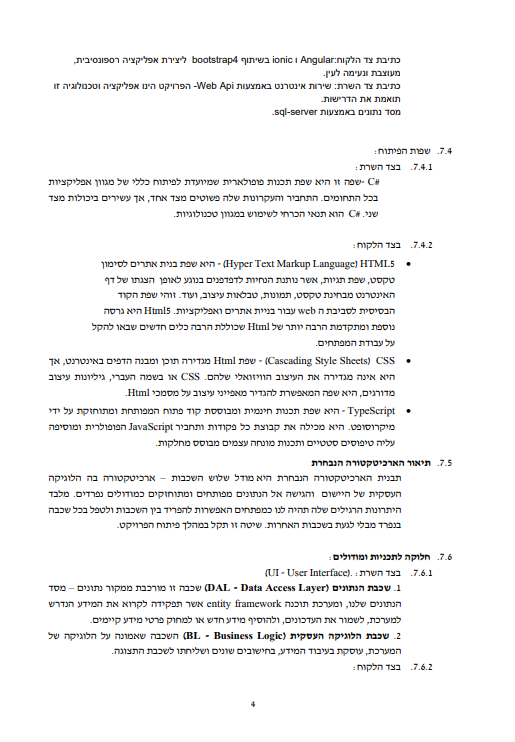
# דף שער

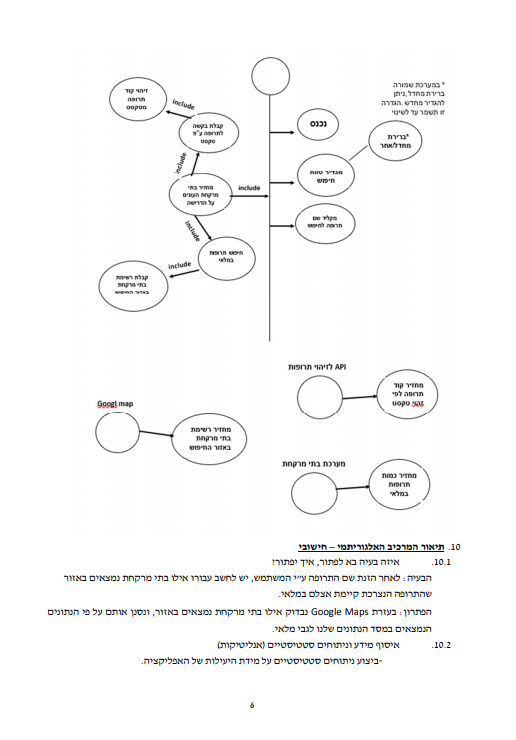
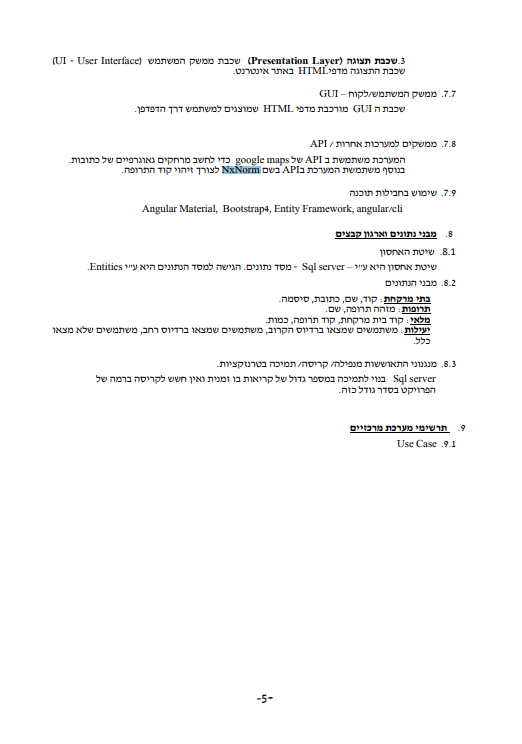
# הצעה לפרויקט גמר

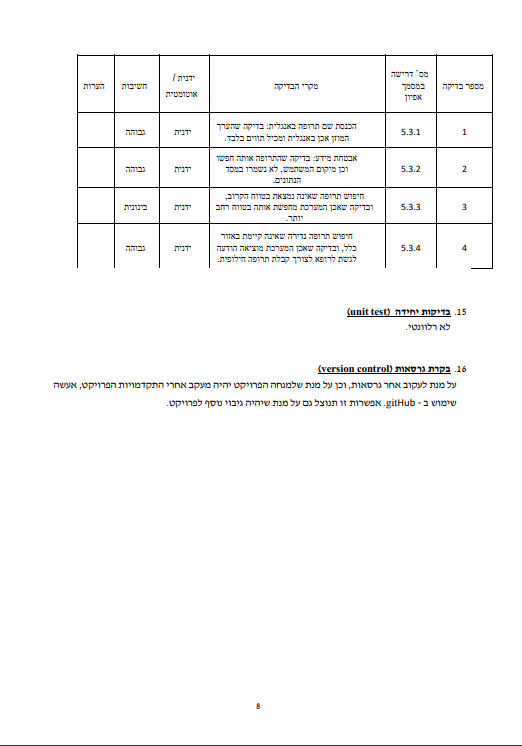
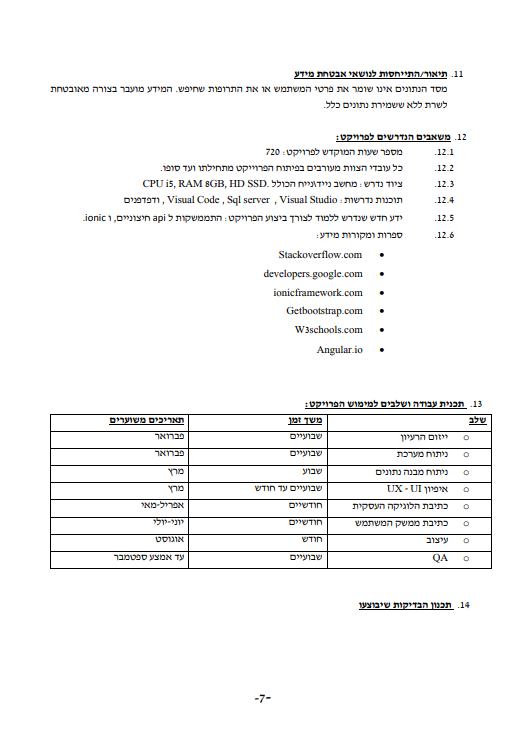




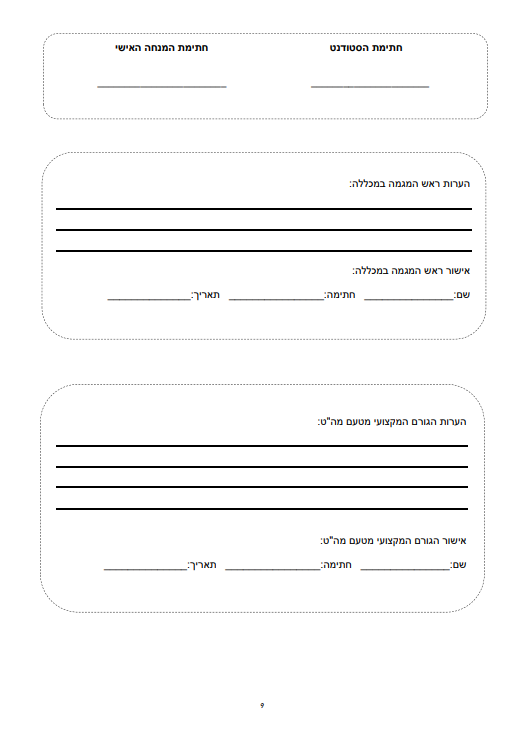








# אישור הצעת הפרויקט ממה"ט



# תודות

ראשית לכל הננו רוצות להודות לריבונו של עולם שהביאנו עד הלום בידו הרחומה והאוהבת.

אנו אסירות תודה לכמות כה רבה של אנשים שליונו בסבלנות, מאור פנים ואוזן קשבת:

לרכזת המגמה גב' חנה ברגמן שמנהלת את המגמה ביד רמה, במסירות ובאחריות מרובה שנים כה רבות .לכל צוות המורות במגמה על הכשרתנו המקצועית ועל סבלנותן הרבה , וכמובן למנחה היקרה חיה וקסשטוק שליוותה אותנו בסבלנות מרובה במשך התקופה ונתנה לנו הרגשה נעימה, עודדה ותמכה והייתה תמיד לצידנו.

וכן תודה ענקית שלוחה להורינו ולמשפחותינו שליוו אותנו במסירות ונתנו לנו עידוד רב. וכן לכל החברות במגמה ששאלו, התעניינו ועודדו...

שה' ישלם שכרכם!

# הצהרה

הצהרת התלמידים/ות

**אנו** קרן מנחם **ת.ז**. 211698055

**ו** חיה צימרמן **ת.ז**. 207645722

החתומים/ות מטה, מצהירים/ות בזאת כי כל הפרויקט המוגש בספר זה, הינו פרי עבודתנו בלבד, על בסיס הנחייתה של המנחה חיה וקסשטוק ותוך הסתמכות על לימודינו במכללה והמידע אשר רכשנו תוך כדי העבודה.

חתימת התלמידים/ות: קרן מנחם וחיה צימרמן

חתימת המנחה: חיה וקסשטוק

# תוכן העניינים

[דף כריכה I](#_Toc526859710)

[דף שער II](#_Toc526859711)

[הצעה לפרויקט גמר III](#_Toc526859712)

[אישור הצעת הפרויקט ממה"ט X](#_Toc526859713)

[תודות 1](#_Toc526859714)

[הצהרה 2](#_Toc526859715)

[תוכן העניינים 3](#_Toc526859716)

[1. מבוא 4](#_Toc526859717)

[2. מדריך למתכנת: 7](#_Toc526859718)

[2.1. אסטרטגיות טכנולוגיות: 7](#_Toc526859719)

[2.2. תיאור מבנה הפרויקט: 7](#_Toc526859720)

[2.3. עקרונות הפיתוח: 8](#_Toc526859721)

[2.4. עקרונות התכנון/ הבניה/ הניתוח 10](#_Toc526859722)

[2.5. תרשימים 11](#_Toc526859723)

[2.6. מבני נתונים מאוחסן 15](#_Toc526859724)

[2.7. תוכן הפרויקט 17](#_Toc526859725)

[3. מדריך למשתמש: 34](#_Toc526859726)

[3.1. הוראות כלליות לשימוש באתר: 34](#_Toc526859727)

[3.2. מסכים: 35](#_Toc526859728)

[4. סיכום ומסקנות: 36](#_Toc526859729)

[5. נספחים: 37](#_Toc526859730)

[7. ביבליוגרפיה: 44](#_Toc526859731)

## מבוא

יצא לך כאשר נצרכת לתרופה חשובה ללכת מבית מרקחת אחד למשנהו כדי למצוא היכן קימת התרופה במלאי?

האפליקציה MediCity בא לפתור בעיה זו.

האפליקציה מכילה ממשק עבור המשתמש:

* המשתמש כאשר נכנס לאפליקציה מקליט או מכניס את שם התרופה הנצרכת.
* האפליקציה מציגה לו את התרופות המתאימות לחיפוש, המשתמש יכול לצפות בפרטי התרופה ולבדוק היכן קימת התרופה במלאי או לפי מרחק או לפי מיקום מסוים שברצונו.
* כמו כן האפליקציה תסיעה לו להגיע לבית מרקחת המבוקש.

**האפליקציה מורכבת מכמה מסכים מרכזיים:**

1. מסך ראשוני – המסך הראשי, במסך זה ניתנת למשתמש להכניס או להקליט שם תרופה ולצפות בתוצאות התואמות לחיפוש

בנוסף יוכל לבחור לפי מה ברצונו לצפות בתוצאות מלאי לפי עיר, מרחק, בית מרקחת מסוים.

1. מסך פרטי תרופה- במסך זה המשתמש צופה בפרטי תרופה שברצונו, יכול לעיין בעלוני התרופה וכן בחומרים הפעילים ומחיר התרופה.
2. מסך הצגת תוצאות מלאי –במסך זה מוצגות למשתמש בתי המרקחת שיש את התרופה במלאי לפי רצונו וכו יכול לבחור אפשרות של יצירת קשר.
3. מסך מפה –במסך זה המשתמש רואה את הדרך שיש לו ללכת או לנסוע עד לבית המרקחת
4. מסך פרטי בית מרקחת- במסך זה רואה את פרטי בית המרקחת שעות פתיחה כתובת טלפונים ליצירת קשר כולל אפשרות ליצירת קשר ועוד.

## מדריך למתכנת:

### אסטרטגיות טכנולוגיות:

בכתיבת פרויקט זה שמנו דגש רב על התאמה טכנולוגית מדויקת ושאיפה לטכנולוגיות חדשניות,

ויחד עם זאת כאלה שלא תפגענה באיכות הפרויקט ותתאמנה לדרישותיו.

צד שרת –server side כתבנו בשפת c# בפלטפורמת Asp.net Web API .

צד לקוח – client side כתבנו בשפות Html5, Typescript, SCSS בפלטפורמת .Ionic

מסד הנתונים SqlServer.

### תיאור מבנה הפרויקט:

#### מבנה הServer side:

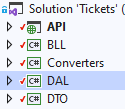
מבנה Server Side השרת נבנה בהתאם למודל השכבות וכולל את הפרויקטים הבאים:

א. DAL- פרויקט זה משמש כשמו Data Access Layer כאחראי על המידע, כלומר על מסד הנתונים. פרויקט זה מתחבר למסד הנתונים הנדרש לפרויקט.

ב. BLL- פרויקט זה מכיל את עיקר הלוגיקה של צד השרת, ומטפל בעיקר בשליפות ושמירת נתונים.

ג. Api- חשיפת API עימו מתממשק האתר, כגון: התחברות משתמש, הוספת משתמש עי ip, שליפת נתונים, שמירת נתונים למשתמש ועוד.

כל שכבה מכירה את השכבה שמעליה בלבד.



מודל שלושת השכבות

#### מבנה הClient side :

מבנה ה-client מורכב משלושה אלמנטים: דפי SCSS דפיHTML ודפי TS.

דפי ה- SCSS אחראיים על העיצוב והמראה הכללי,

דפי ה-HTML מהווים את התצוגה,

ודפי ה-TS כוללים בתוכם פונקציות לוגיות שונות, התחברות לשרתים וכן שירותים שונים

ההתחברות לצד השרת ע"י קריאות Http

### עקרונות הפיתוח:

מאחורי קוד המערכת עומדים עקרונות תכנותיים רבים שחלקם לא היו מוכרים לנו. בסעיף זה ננסה

להבהיר את העקרונות העיקריים שעל פיהם פעלנו לאורך כתיבת הקוד.

**SQl**

היא מערכת לניהול בסיס נתונים במודל היחסי (RDBMS) של חברת מיקרוסופט., שפת הפיתוח שבאמצעותה מועברות הפקודות למערכת (על ידי כתיבת פקודות או שימוש בממשק גרפי ששולח פקודות באופן סמוי) היא Transact-SQL ‏(TSQL) שהיא מימוש תקן ANSI של שפת SQL משמשת לתשאול וטיפול בנתונים (DML), יצירת טבלאות והיחסים ביניהן (DDL) ותחזוקת המערכת תוך שימוש בתוכניות שירות שונות.

SQL הוא מסד נתונים בו נשמרים הנתונים בטבלאות בעלות קשרי גומלין, וניתן לשלוף ממנו מידע ע"י שאלתות SQL.

יש לו יכולות רבות נוספות כמו view, פרוצדורות, פונקציות, טרנזקציות ועוד.

זהו מסד הנתונים הנפוץ ביותר.

**ASP.net** **Web API**

Iמה זה API?

API הוא קיצור של .Interface Program Application במונחים פשוטים, זהו הממשק של פיסת

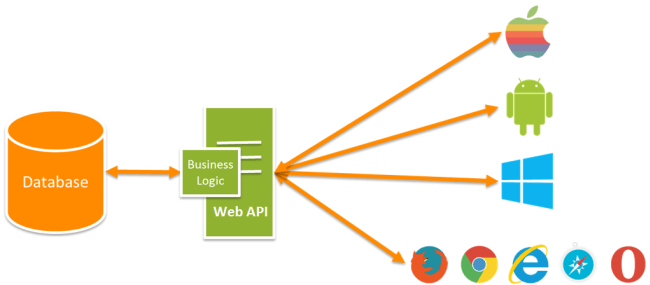
תוכנה עם העולם החיצוני, הקובע איך ישתמשו בה. אלה החוקים שנקבעו עבור האינטראקציה

שלה עם העולם הרחב, אלה יקבעו אילו חלקים של התוכנה יכולות לדבר עם תוכנות אחרות, ואיך

היא תגיב.

מה זה Web API ?

ASP.NET Web API הוא מסגרת המקלה על בניית שירותי HTTP המגיעים למגוון רחב של לקוחות, כולל דפדפנים והתקנים ניידים. ASP.NET Web API הוא פלטפורמה אידיאלית לבניית יישומים RESTful ב- NET Framework.



מה זה REST?

Rest( ראשי תיבות Transfer State Representational )היא סגנון ארכיטקטוני לכתיבת צד שרת.

התפיסה הארכיטקטונית ב- Rest היא תפיסת שרת-לקוח. תפיסה זו מחייבת קיום לקוח ושרת.

לקוחות יוזמים פניות המכילות בקשות לשרתים. השרת מעבד את הפנייה, ומחזיר תגובות

מתאימות.

בכל מצב נתון הלקוח יכול להיות בתהליך של שינוי מצב או במצב מנוחה (rest .(במצב של מנוחה

הלקוח יכול להיות באינטראקציה עם המשתמש, אבל אינו תופס משאבים בשרת. הלקוח שולח פניות

כאשר הוא מוכן לעבור למצב חדש. כאשר קיימת פנייה אחת או יותר שטרם הסתיים הטיפול בהן,

הלקוח נמצא במצב של מעבר ממצב למצב.

REST הוא לא פרוטוקול )כמו SOAP )אלא יותר קונבנציה שבה אנחנו משתמשים. Rest מבוסס ברב

המקרים על פרוטוקול HTTP ,למרות זאת REST היא ארכיטקטורה כללית הניתנת למימוש גם

בסביבות אחרות ולא רק תחת HTTP.

עבודה ב REST-מחייבת התחשבות באילוצים ובמגבלות של ארכיטקטורה זו, נפרט את חלקם:

• שרת-לקוח: עובד רק בתפיסת שרת לקוח, כאשר האחד אינו מושפע ממה שמתרחש באחר,

למעט המסרים העוברים ביניהם.

• Stateless :ההקשר (context (של הלקוח בפניה לשרת, אינו נשמר בשרת. מגבלה זו נועדה

לשפר את מדרגיות (Scalability (השרת.

כשאנו יוצרים שירות שממלא אחר הקונבנציות של REST אנחנו יוצרים שירות RESTful.

:Rest API - לסיכום

ממשק API של REST מגדיר קבוצה של פונקציות אשר מפתחים יכולים לבצע בקשות ולקבל תגובות

באמצעות פרוטוקול HTTP כגון GET ו- POST.

**Typescript**

היא שפת תכנות חינמית ומבוססת קוד פתוח המפותחת ומתוחזקת על ידי מיקרוסופט .היא מכילה

את קבוצת כל פקודות ותחביר JavaScript הפופולרית, כלומר כל קוד

JavaScriptהוא גם קוד Typescript תקין, ומוסיפה עליה טיפוסים סטטיים ותכנות מונחה עצמים

מבוסס מחלקות. בין התכונות שהיא מוסיפה: typing static ,תמיכה במחלקות ותמיכה במודולים

ובדקורטורים.

Typescript היא סופר-סט (superset) של JavaScript כלומר, הקוד מבוסס על JavaScript וחייב

לעבור קומפילציה (תרגום) ל JavaScript-כדי שהדפדפנים יבינו אותו מפני שדפדפנים

מבינים JavaScript בלבד. בסופו של דבר קוד Typescript מעובד לכדי קוד JavaScript ולכן ניתן

לומר "באחריות" שהשפה נתמכת בכל דפדפן שתומך ב- JavaScript.

קבצי Typescript יישמרו בסיומת ts.

### עקרונות התכנון/ הבניה/ הניתוח

הפרדת שכבות:

כל תוכנה/ אתר שנפגוש בעולם מבוססת על ארכיטקטורת שכבות הנקראת בשם Three Tier- -שלש שכבות במבנה שלBL-UI Application DAL-. זוהי תבנית עיצוב בסיסית שמגדירה הפרדת האפליקציה ל: שכבת נתונים, שכבת לוגיקה, ושכבת ממשק משתמש.

לתבנית עיצוב זו יתרונות רבים:

1.תחזוקה:

ניתן להחליף או לתקן מימוש פנימי של שכבה אחת בארכיטקטורה בלי לשנות שכבה אחרת.

2.נוחות פיתוח:

אדם אחד עובד על רכיב בתכנה, אדם אחר עובד על רכיב אחר, כל עוד שהחתימות זהות ניתן לשלב כוחות ולייעל זמני פיתוח.

3. בדיקות:

תקלה כלשהיא מבודדת בכל שכבה בנפרד, לדוגמא אם לא קיבלנו רשימת נתונים לתצוגה נבדוק קודם את שלב הבינים (שכבת ה BL) אם הנתונים שם תקינים נדע בוודאות שגם ברמת שכבת ה DAL הנתונים תקינים וכל שנותר הוא לפתור את התקלה ברמת ה UI .

4. שימוש חוזר:

נניח שהרכיבים BL+ DAL אהובים עלינו ועובדים היטב, ניתן להחליף את פלטפורמת UI לטכנולוגיה אחרת במינימום מאמץ.

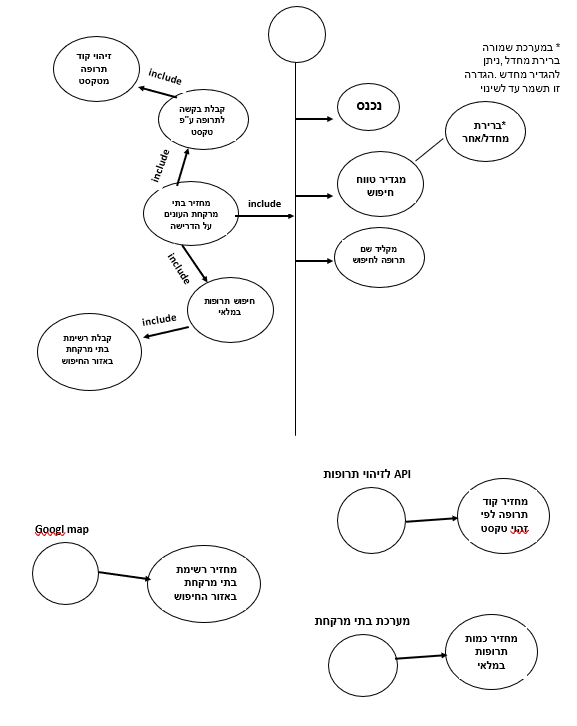
5. אבטחה:

נוח יותר להגדיר Interface API לשכבה מסוימת בלי לחשוף מבני נתונים Logic או Data שלא רלוונטי למשתמש, כמו גם חסימה בפני האקרים (רלוונטי יותר בטכנולוגיות Web ) ברמות שונות.

הפרויקט של צד שרת מכיל את שלושת השכבות הנ"ל.

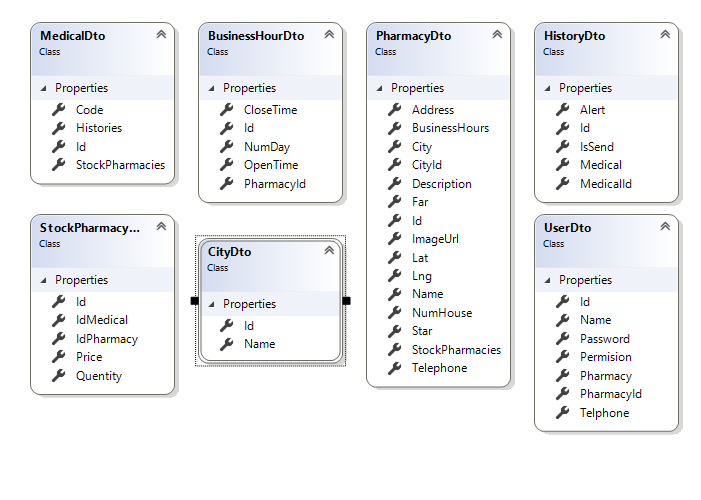
### תרשימים

#### תרשים זרימה

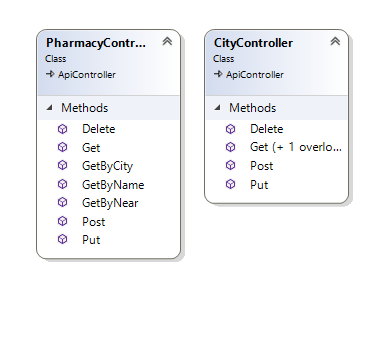


#### תרשים מראה מחלקות

* פרויקט הDal

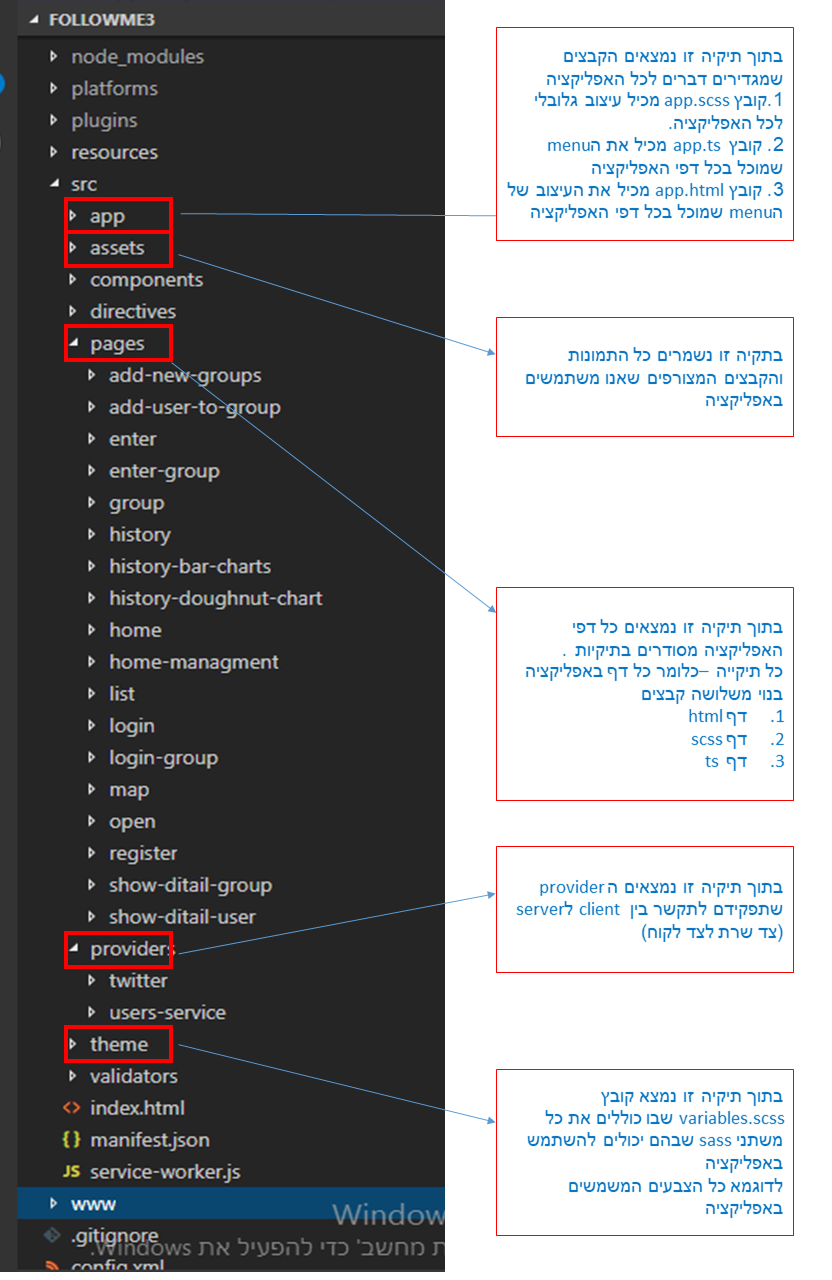
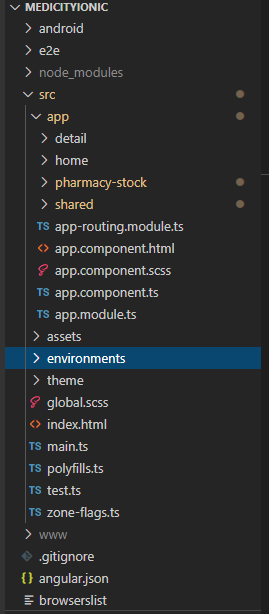


* פרויקט API- Controllers



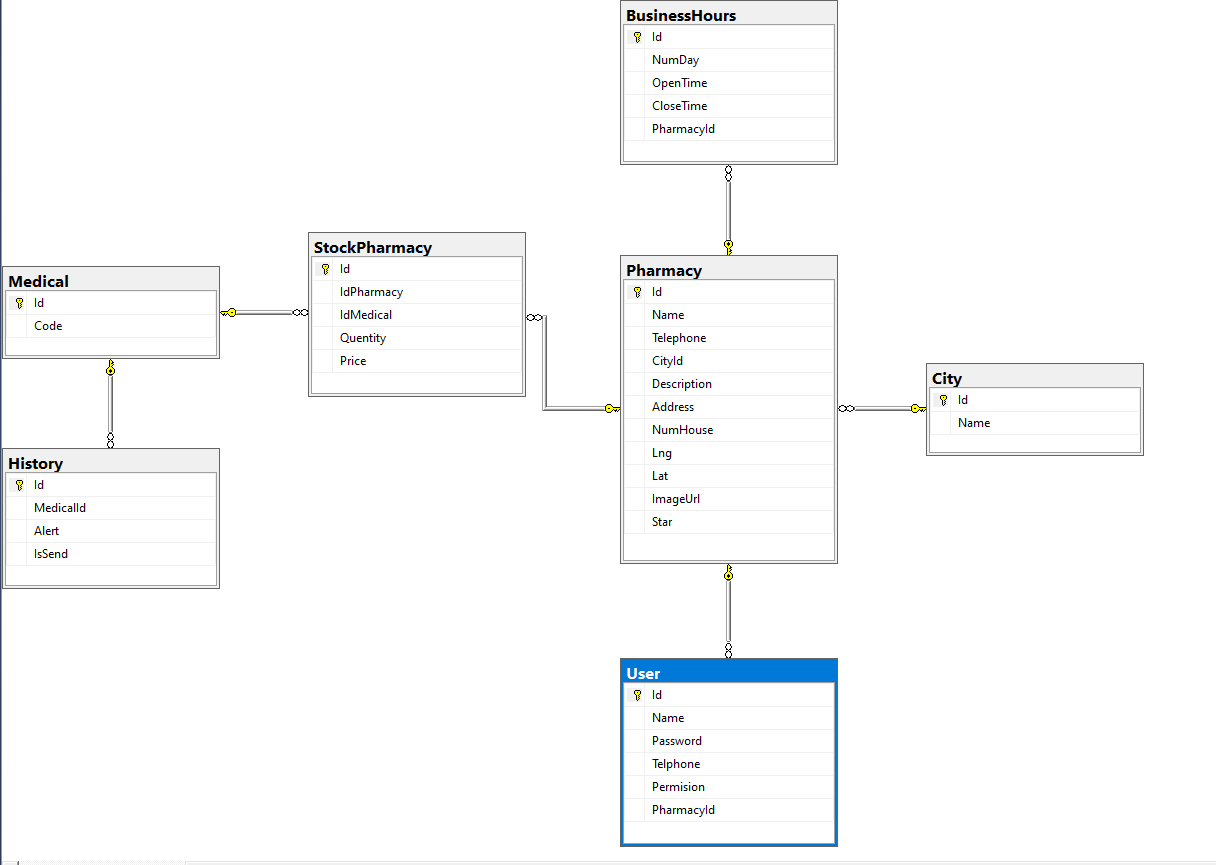
#### מבנה קבצים ותיקיות

צד client :



### מבני נתונים מאוחסן

#### מבנה Sql



### תוכן הפרויקט

#### תיאור מחלקות

מחלקות ב Dal

**User**-(משתמש)

public partial class User

{

public int Id { get; set; }

public string Name { get; set; }

public string Password { get; set; }

public string Telphone { get; set; }

public Nullable<int> Permision { get; set; }

public Nullable<int> PharmacyId { get; set; }

public virtual Pharmacy Pharmacy { get; set; }

}

**City**- (עיר)

public class CityDto

{

public int Id { get; set; }

public string Name { get; set; }

}

**BusinesHour**- (שעות פעילות בתי מרקחת)

public class BusinessHourDto

{

public int Id { get; set; }

public int NumDay { get; set; }

public System.TimeSpan OpenTime { get; set; }

public System.TimeSpan CloseTime { get; set; }

public int PharmacyId { get; set; }

}

**History -** (היסטורית חיפושי תרופה)

public class HistoryDto

{

public int Id { get; set; }

public int MedicalId { get; set; }

public bool Alert { get; set; }

public bool IsSend { get; set; }

public MedicalDto Medical { get; set; }

}

**Medical** -(תרופה )

public class MedicalDto

{

public int Id { get; set; }

public string Code { get; set; }

public List<HistoryDto> Histories { get; set; }

public List<StockPharmacyDto> StockPharmacies { get; set; }

}

**Pharmacy** - (בית מרקחת)

public class PharmacyDto

{

public int Id { get; set; }

public string Name { get; set; }

public string Telephone { get; set; }

public int CityId { get; set; }

public string Description { get; set; }

public string Address { get; set; }

public int NumHouse { get; set; }

public decimal Lng { get; set; }

public decimal Lat { get; set; }

public string ImageUrl { get; set; }

public int Star { get; set; }

public double Far { get; set; }

public List<BusinessHourDto> BusinessHours { get; set; }

public CityDto City { get; set; }

public List<StockPharmacyDto> StockPharmacies { get; set; }

}

**StockPharmacy** - (מלאי תרופה בבית מרקחת)

public class StockPharmacyDto

{

public int Id { get; set; }

public int IdPharmacy { get; set; }

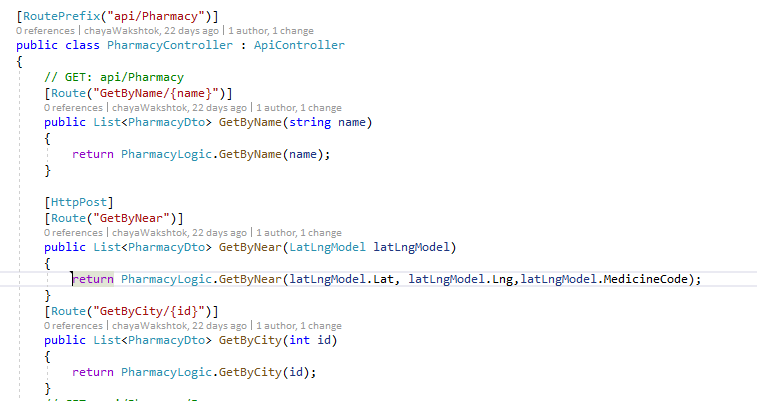
public int IdMedical { get; set; }

public int Quentity { get; set; }

public decimal Price { get; set; }

}

**Controllers :PharmecyController** (בתי מרקחת)

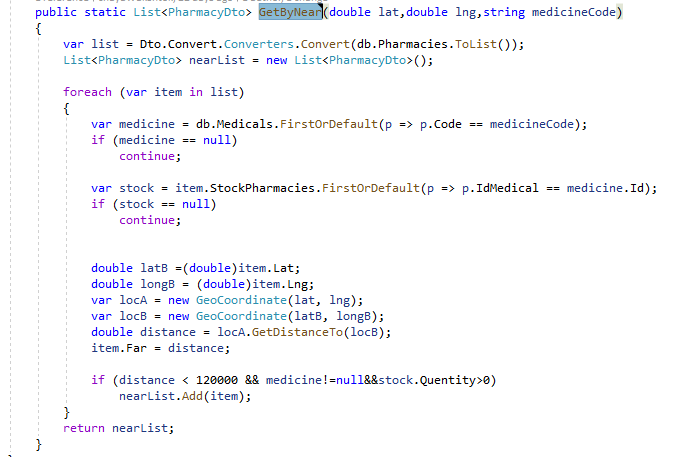


#### תאור הפנקציות

**פונקציות בצד שרת:**

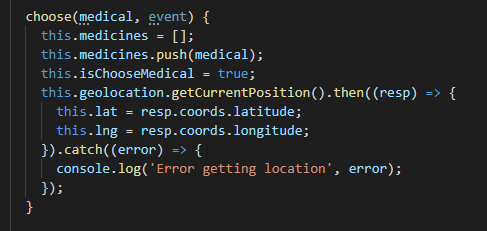
1. פונקצית GetByDistance

פונקציה זו מקבלת קוד תרופה ומיקום בן אדם ומחזירה באילו בתי מרקחת בסביבה יש במלאי את התרופה.



**פונקציות בצד לקוח:**

פונקציה כאשר המשתמש בוחר תרופה מתוצאות החיפוש מזהה את מיקומו.



סרבייס לקבלת נתוני תרופות מapi של משרד הבריאות



## מדריך למשתמש:

### הוראות כלליות לשימוש באפלקציה:

#### מדריך למשתמש:

בעת כניסה לאפליקציה הקלט או הקלד את שם התרופה שברצונך לחפש.

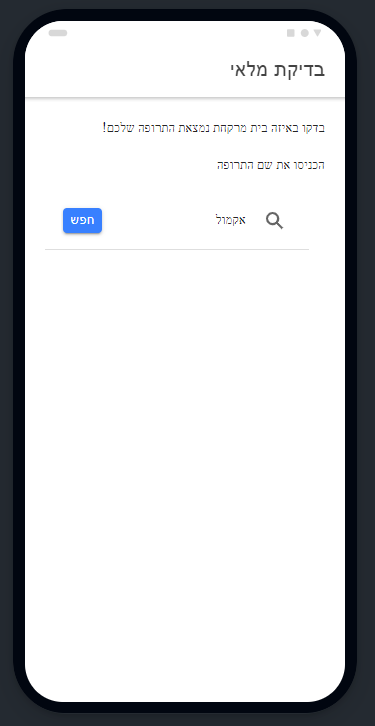
בחר מתוך רשימת התוצאות את התרופה הרצויה והכנס פרמטר לחיפוש בבתי המרקחת

בעת לחיצה על תרופה יוצג פרטי התרופה באופן מפורט.

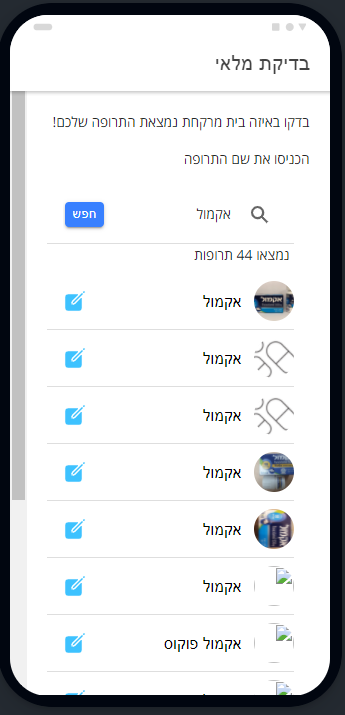
כאשר תאשר את פרטי החיפוש האפלקציה תציג לך את בתי המרקחת שיש ברשותם את התרופה במלאי לפי פרמטר החיפוש (מרחק/עיר/שם)

בדף תוצאות בתי מרקחת יש באפשרותך לבחור לנווט למקום או יצירת קשר, כאשר תלחץ על אחד מבתי המרקחת יוצגו פרטים ושעות פתיחה.

### מסכים:



חיפוש תרופה



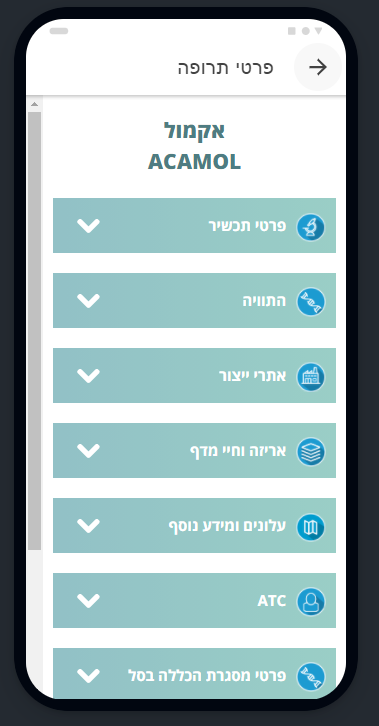
לחץ להצגת פרטים

מציג תוצאות חיפוש

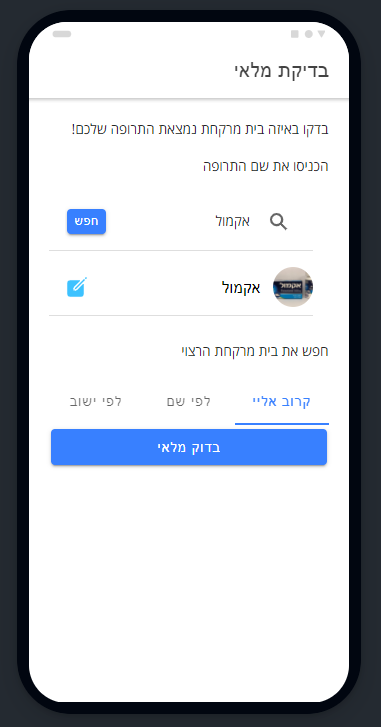
כאשר המשתמש רשום או אחרי הרשמה עובר לדף צילום טיקט בגד

תוצאת ההמרה לשקל מדולר לדוג'

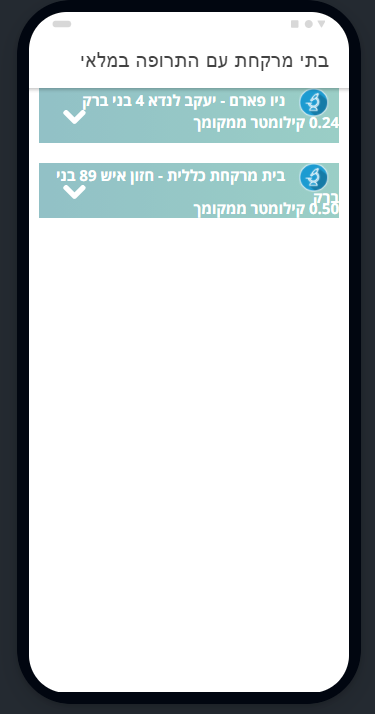
ואפשרות לעדכן רכישה



פרטי תרופה



בחר קריטריון לחיפוש



תוצאות חיפוש

## סיכום ומסקנות:

השקענו רבות בבניה נכונה של המערכת ועל תקשורת נכונה עם השרת, ראשית הפרויקט הקנה לנו מרחב הסתכלות ופרספקטיבה רחבה על פרויקט מושלם, משלב ההצעה ועד לתוצר המוגמר תוך דרך ארוכה הכוללת באגים ופתרונם ויצירתיות רבה על מנת להפוך את האפליקציה ליעילה ,חווייתית ומקצועית ככל האפשר. במהלך הפרויקט למדנו הרבה ,ואנו מרגישות כי הוא תרם לנו רבות . תחומים שהכרנו באופן שטחי ,נעשו ידידותיים בהרבה ממה שחשבנו .השקענו זמן ומאמץ מרובים בלמידת נושאים וטכנולוגיות חדשים .גילינו שמהנה גם אם לא קל ללמוד ולהכיר תחומים חדשים ולנסות דרכי חשיבה מגוונות ומפתיעות שלא חשבנו עליהם ממבט ראשון. למדנו לחשוב בהגיון וביעילות על מנת להשיג תוצאות במהירות. למדנו לעבוד עם api convert ועם ionic ו plugin of Cordova and native

ומעל הכול הפנמנו שסבלנות והתמדה-אין כמותם בכל משימה הנראית קשה עד בלתי אפשרית...

## נספחים:

**Dependences- Native**

 "dependencies": {

    "@angular/common": "~9.1.6",

    "@angular/core": "~9.1.6",

    "@angular/forms": "~9.1.6",

    "@angular/platform-browser": "~9.1.6",

    "@angular/platform-browser-dynamic": "~9.1.6",

    "@angular/router": "~9.1.6",

    "@capacitor/android": "^2.4.0",

    "@capacitor/core": "2.4.0",

    "@ionic-native/core": "^5.0.7",

    "@ionic-native/geolocation": "^5.28.0",

    "@ionic-native/splash-screen": "^5.0.0",

    "@ionic-native/status-bar": "^5.0.0",

    "@ionic/angular": "^5.0.0",

    "@ionic/lab": "^3.2.0",

    "cordova-android-support-gradle-release": "^3.0.1",

    "cordova-plugin-geolocation": "^4.0.2",

    "ionic4-auto-complete": "^2.9.0",

    "rxjs": "~6.5.1",

    "tslib": "^1.10.0",

    "zone.js": "~0.10.2"

  },

  "devDependencies": {

    "@angular-devkit/build-angular": "~0.901.5",

    "@angular/cli": "~9.1.5",

    "@angular/compiler": "~9.1.6",

    "@angular/compiler-cli": "~9.1.6",

    "@angular/language-service": "~9.1.6",

    "@capacitor/cli": "2.4.0",

    "@ionic/angular-toolkit": "^2.1.1",

    "@types/jasmine": "~3.5.0",

    "@types/jasminewd2": "~2.0.3",

    "@types/node": "^12.11.1",

    "codelyzer": "^5.1.2",

    "jasmine-core": "~3.5.0",

    "jasmine-spec-reporter": "~4.2.1",

    "karma": "~5.0.0",

    "karma-chrome-launcher": "~3.1.0",

    "karma-coverage-istanbul-reporter": "~2.1.0",

    "karma-jasmine": "~3.0.1",

    "karma-jasmine-html-reporter": "^1.4.2",

    "protractor": "~5.4.3",

    "ts-node": "~8.3.0",

    "tslint": "~6.1.0",

    "typescript": "~3.8.3"

  },

## ביבליוגרפיה:

אתרים בנושא תכנות:

* stackoverflow.com
* webmaster.org.il
* [.c-sharpcorner.com](https://www.c-sharpcorner.com/article/web-api-in-asp-net)
* [.djamware.com](https://www.djamware.com/post/5a48517280aca7059c142972/ionic-3-angular-5-firebase-and-google-maps-location-tracking)
* .joshmorony.com
* .dotnetcurry.com
* .msdn.microsoft.com