**עבודה עם מסד נתונים - DataBase**

בחלק מהאפליקציות המשתמשות בנתונים יש צורך לשמור נתונים.

נניח שנרצה לשמור את הציונים שלנו ומדי פעם לחשב ממוצעים.

או למשל אפליקציה של רשימת קניות או ToDoList (שתי אפליקציות לפרוייקט).

מצב אחר בו נרצה לשמור נתונים יהיה באפליקציה העובדת OnLine מול האינטרנט אך האינטרנט לא זמין. הנתונים ישמרו במכשיר וכשהאינטרנט יתחבר שוב, יבוצע סנכרון.

למדנו שניתן לשמור מידע באמצעות SharedPreference אך כידוע זה מסוגל לשמור נתונים פרימיטיביים בלבד בשיטת key/value וזה מוגבל לאפליקציה.

שיטה נוספת לשמירת נתונים במחשב היא באמצעות קבצים אך זב גם די מוגבל.

השיטה המומלצת היא עבודה עם מסד נתונים.

**מהו מסד נתונים**

זהו מנוע (תוכנה) המארגן מידע בדיסק (או אמצעי אחסון קבוע אחר) בצורה של טבלאות.

במסד נתונים יכולות להיות מספר טבלאות ואף קשרים בין הטבלאות.

**טבלה:** אוסף של שורות ועמודות.

שורה - מייצגת ישות אחת (כל הנתונים של הישות).

עמודה - מייצגת פריט מידע בודד לגבי הישות.

שורה – Record, עמודה – Field.

דוגמא: תלמיד – שם משפחה, שם פרטי ות.ז.

כל הנתונים יחד יהוו שורה בטבלה. כל אחד מהם בנפרד יהוו עמודה בטבלה.

כל נתוני התלמידים יהוו טבלה.

**SQL – Standart Query Language**

SQL היא השפה שבאמצעותה ניתן לבצע פעולות במסד הנתונים.

כגון הכנסת נתונים חדשים לטבלה, שינוי נתונים קיימים, מחיקת נתונים מהטבלה, שליפת נתונים כולל אופציה של סינון ומיון ואף שליפת נתונים סטטיסטיים.

למד/י [**כאן**](http://webmaster.org.il/articles/sql) עוד על SQL.

**מסדי נתונים**

קיימים מספר רב של מסדי נתונים:

**גדולים** מאוד המאפשרים לאחסן כמויות אדירות של מידע ולבצע פעולות מסובכות על המידע.

**בינוניים** וקטנים המאפשרים לאחסן כמות נתונים קטנה יותר וכח המחשוב שלהם קטן יותר.

**קטנים** המותאמים לעבודה בסביבה דלת זיכרון אחסון וזיכרון עבודה.

מסדי נתונים גדולים: SQL Server של חברת מיקרוסופט, Oracle ו-MySql של חברת Oracle, DB של חברת IBM.

מסדי נתונים בינוניים Access ו-SQL Compact של חברת מיקרוסופט.

מסד נתונים קטן המתאים לעבודה עם מכשירים קטנים SQLite.

**מסד הנתונים SQLite**

זהו מסד נתונים קטן (גודל המנוע שלו כ-150KB) והוא מותאם לעבודה עם מכשירים קטנים ונפוץ מאוד במכשירים ניידים.

אמנם קטן אך בעל יכולות גבוהות בניהול המידע וביצוע פעולות עליו.

לשמחתנו נבנתה ספריה לעבודה מאוד פשוטה עם מסד הנתונים. קיימת ספריה המותאמת ל-Android בעבודה עם Java וקיימת ספריה המותאתמת ל-Xamarin בעבודה ב-C#.

ספריה לעבודה ב-Java – ROOM.

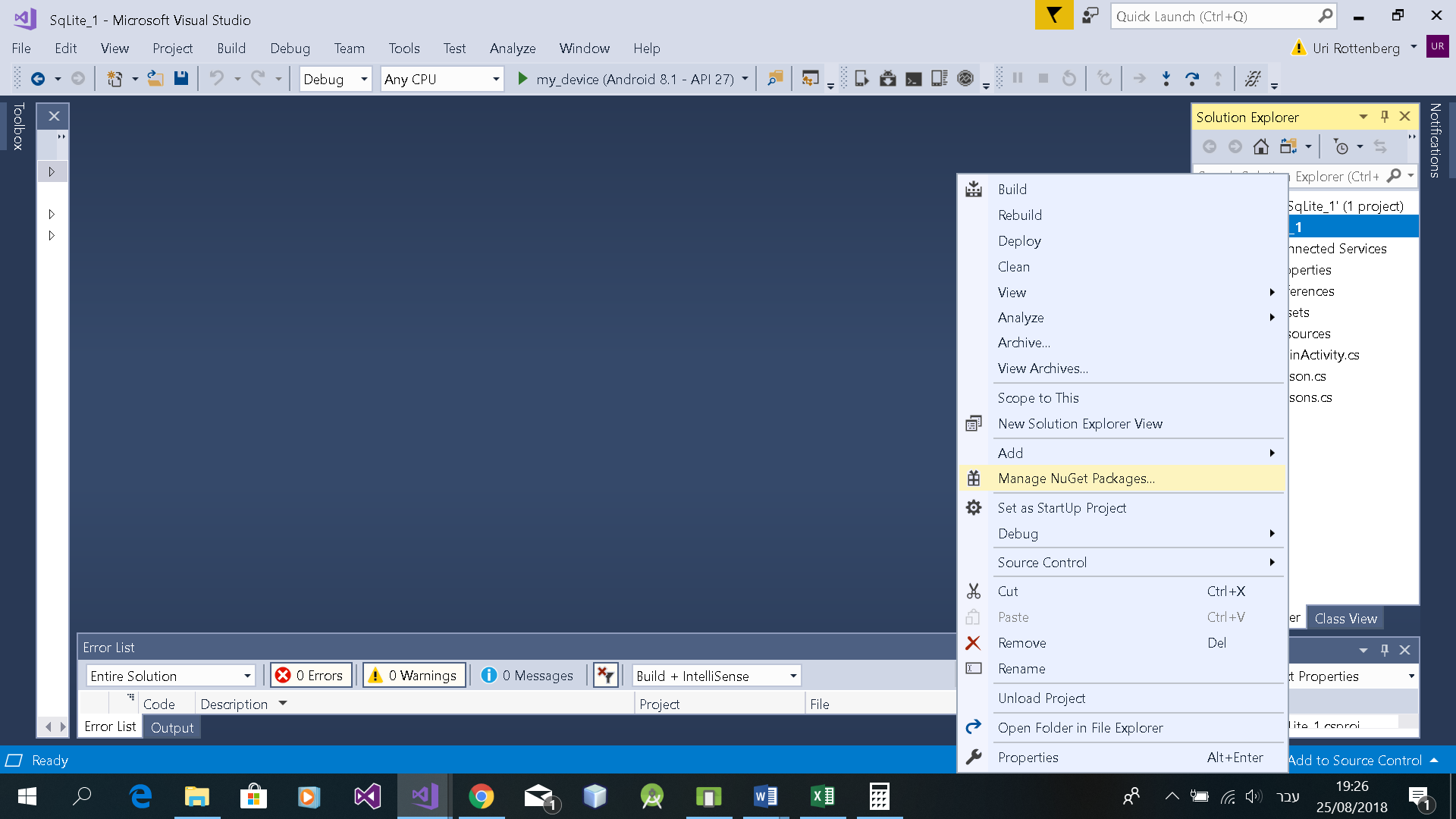
ספריה לעבודה ב-Xamarin + C# - sqlite-net-pcl.

**התחלת עבודה**

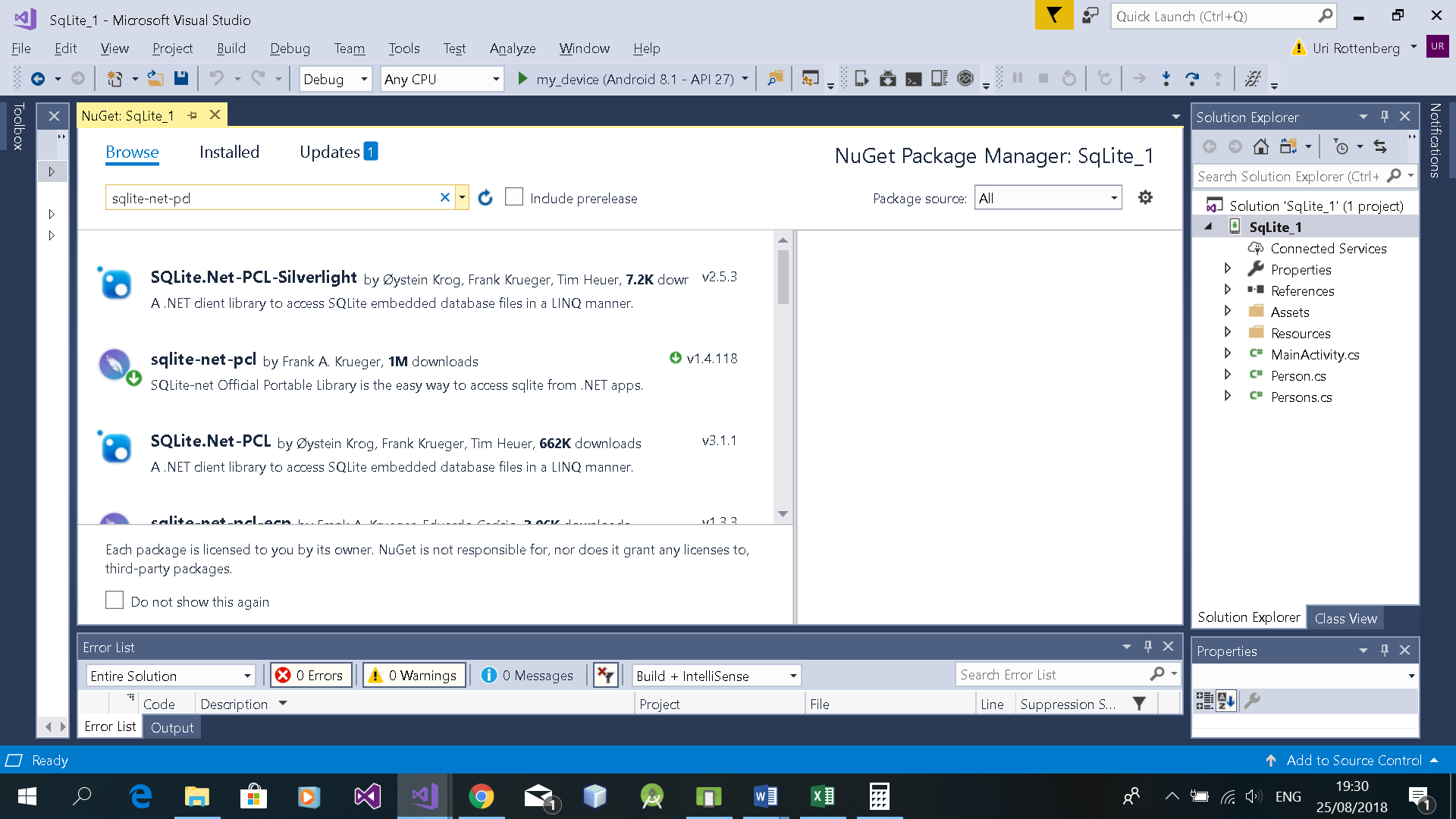
נפתח פרויקט Xamarin-Android חדש.

להוספת הספרייה לעבודה עם SQLite:

1. נלחץ R-Click על שם הפרויקט.
2. בתפריטון שנפתח נבחר: Manage NuGet Packages.



1. נפתח החלון הבא:

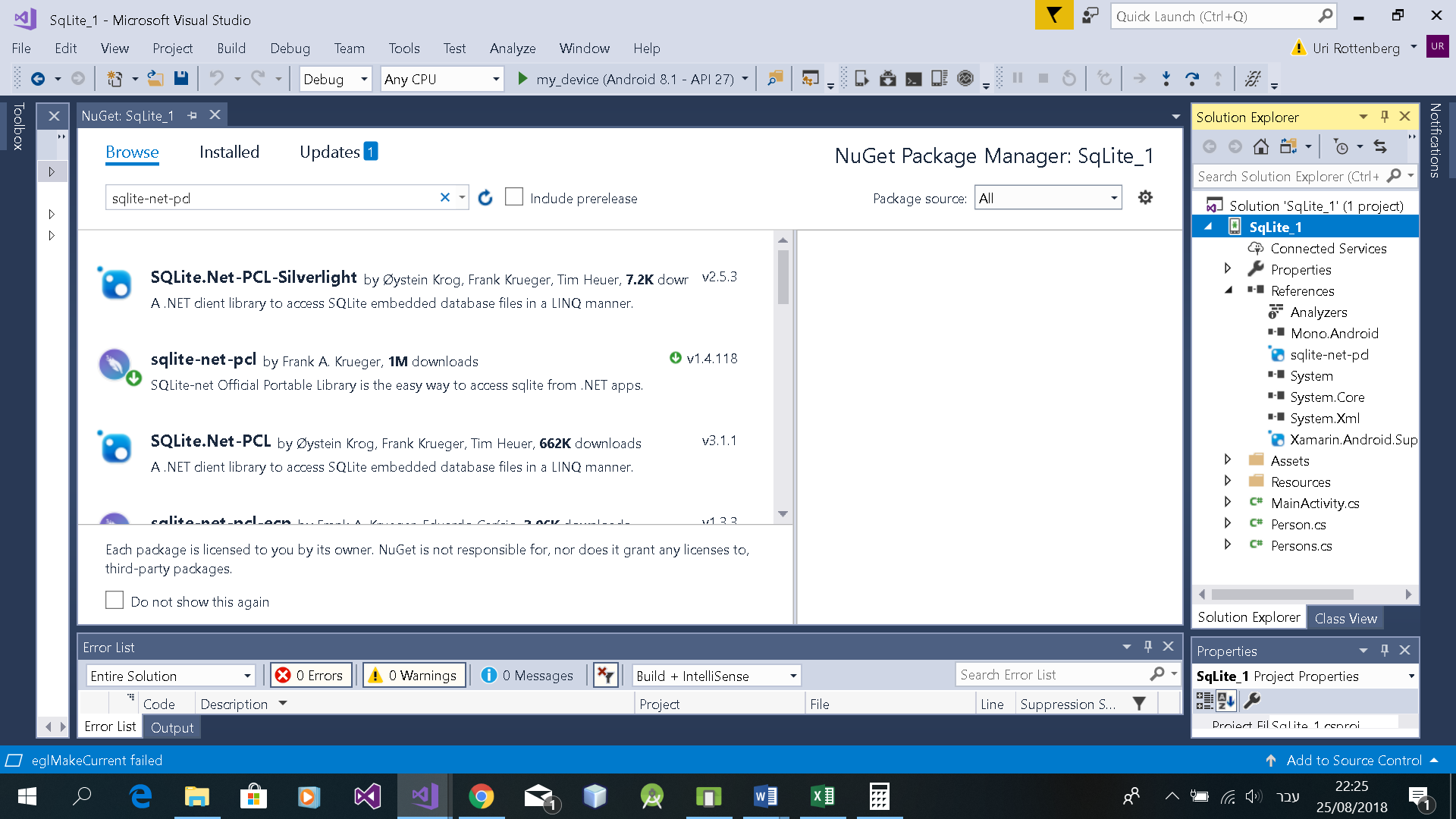


**3**

**2**

**1**

1. בחר/י Browse (1).
2. הקלד/י בשורת החיפוש sqlite-net-pcl (2).
3. לחץ/י על החץ להורדה.
4. נוכל לראות בתיקיה References את הספרייה שנוספה.



1. על מנת להשתמש בספרייה יש לרשום בקובץ C# - using SQLite;

**יצירת Connection**

על מנת שנוכל לגשת ולעבוד עם מסד הנתונים אנו זקוקים ליצירת קשר עם מסד הנתונים.

זאת נעשה ע"י אובייקט מטיפוס SQLiteConnection.

אנו צריכים לספק את המסלול המלא למסד הנתונים.

1. string dbName = "App.db";

1. string documentsPath = System.Environment.GetFolderPath

(System.Environment.SpecialFolder.Personal);

1. string path = System.IO.Path.Combine(documentsPath, dbName);
2. SQLiteConnection connection = new SQLiteConnection(path);
3. קובע את שם מסד הנתונים.
4. מגדיר את תיקיית המסמכים של האפליקציה.

התיקייה SpecialFolder.Personal זו תיקייה המוגדרת במעה"פ Android היא נוצרת לאחר התנת האפליקציה על המכשיר. לכל אפליקציה תיקייה Personal משלה.

זהו נתיב וירטואלי ולא פיזי. הנתיב הפיזי נקבע בהתאם למיקום הפיזי של האפליקציה.

1. מגדיר את המסלול המלא למסד הנתונים.

הפקודה Path.Combine הופכת את המסלול של התיקייה הוירטואלית למסלול פיזי בהתאם לאפליקציה.

1. יצירת אובייקט Connection.

כעת נוכל לבצע פעולות על מסד הנתונים.

* **שים/י לב !!!**

ניתן להריץ פקודה זו אין ספור פעמים. מסד הנתונים יווצר רק פעם אחת.

אם מסד הנתונים כבר קיים אזי האפליקציה תדלג על פקודה זו.

**יצירת טבלה במסד הנתונים**

במסד נתונים רגיל יש ליצור את הטבלה באופן ידני או ע"י פקודת SQL מתאימה.

גם ב-SQLite ניתן לעשות את זה אך...

באמצעות הספרייה sqlite-net-pcl הדבר הפך להיות פשוט ביותר.

ראשית נגדיר מחלקה שאובייקטים ממנה נרצה לשמור במסד הנתונים.

public class Person

{

public int Id { get; set; }

public string Family { get; set; }

public string Name { get; set; }

public int Age { get { return age; }

set

{

age = (value >= 0 && value <= 120) ? value : 1;

}

}

}

השדה Id יהיה מספר ייחודי המזהה את האובייקט.

כמובן שלמחלקה נוסיף את כל הדרוש להפעלתה התקינה כולל Constructor ופעולות נוספות.

**הוספת עיטורים:** עיטור הוא סימון מסוים לטבלה או לשדה הנותן הוראה למסד הנתונים כיצד להגדיר.

על מנת להוסיף עיטורים יש להגדיר את מרחב השמות using המתאים

using SQLite;

[Table ("Persons")]

public class Person

{

[PrimaryKey, AutoIncrement]

public int Id { get; set; }

העיטור Table קובע מה יהיה שמה של הטבלה במסד הנתונים.

אם לא נעטר שם הטבלה במסד הנתונים יהיה כשם המחלקה.

העיטור PrimaryKey הופך את המאפיין המעוטר לשדה המפתח הראשי של הטבלה.

העיטור AutoIncrement קובע כי שדה זה יתקדם ב-1 כל פעם שנוסיף רשומה במסד הנתונים.

קיימים עיטורים נוספים אך לא נדון בעם כעת.

**יצירת הטבלה במסד הנתונים:**

connection.CreateTable<Person>();

פקודה זו תיצור לנו טבלה במסד הנתונים עם שמות השדות והטיפוסים כפי שהוגדרו במחלקה.

* **שים/י לב !!!**

ניתן להריץ פקודה זו אין ספור פעמים. הטבלה תיווצר רק פעם אחת.

אם הטבלה כבר קיימת אזי האפליקציה תדלג על פקודה זו.

**הוספה, שינוי, מחיקה**

**הוספת רשומה:**

Person person = new Person(etFamily.Text,

etName.Text,

Convert.ToInt32(etAge.Text));

connection.**Insert**(person);

**שינוי רשומה:**

person.Family = "Cohen";

connection.**Update**(person);

**מחיקת רשומה:**

connection.**Delete**(person);

**פעולות נוספות**

**שליפת כל הרשומות מהטבלה:**

List<Person> persons = connection.Table<Person>().ToList();

**שליפת נתונים ע"פ משפט SQL:**

persons = connection.Query<Person>

("SELECT Name WHERE Family = 'Cohen').ToList();

**שליפת רשומה עפ"י ה-id שלה:**

person = (Person)connection.Get<Person>(5);

* הפקודה Get מחזירה object.

**חיסרון**

ראינו עד כמה קל לעבוד עם מסד נתונים באמצעות הספרייה אך כפי שניתן להבחין כל הפקודות קשורות באופן הדוק לטיפוס הנתונים עליו היא פועלת.

המשמעות שלכל טיפוס נתונים נצטרך להגדיר מחלקה לטיפול בכל הפעולות עבור אותו טיפוס נתונים.

**הפיתרון:** לכתוב מחלקה Generic - <T> שתבצע את הפעולות בהתאמה.

* קרא/י [**כאן**](https://docs.microsoft.com/en-us/xamarin/android/data-cloud/data-access/using-sqlite-orm) עוד על שימוש ב-SQLite באמצעות Xamarin.