# Oefeningen databases

[Oefeningen databases 1](#_Toc437456831)

[ER- schema opgave 2](#_Toc437456832)

[ER- schema beschrijving 2](#_Toc437456833)

[Database structuur 2](#_Toc437456834)

[SQLite 2](#_Toc437456835)

[Project introductie: GPXTool 2](#_Toc437456836)

[Opzet van het project (u krijgt dit van ons) 2](#_Toc437456837)

[SQLite in een UWP project 2](#_Toc437456838)

[SQLiteService 3](#_Toc437456839)

[Initialisatie van de SQLite omgeving 3](#_Toc437456840)

[SQLite tabellen via klasses 4](#_Toc437456841)

[Project uitwerking 4](#_Toc437456842)

[Models 4](#_Toc437456843)

[Repositories (gedeeltelijk) 4](#_Toc437456844)

[TrkRepository 4](#_Toc437456845)

[Bewaren van Trks 4](#_Toc437456846)

[Verzamelen van Trks 5](#_Toc437456847)

[Testen van de gedeeltelijke oplossing 5](#_Toc437456848)

[New Track toevoegen aan de database 5](#_Toc437456849)

[Verificatie van de database 6](#_Toc437456850)

[Verdere Repositories uitwerking 6](#_Toc437456851)

[TrkPtRepository 6](#_Toc437456852)

[Tracker implementatie 7](#_Toc437456853)

[Tracker klasse 7](#_Toc437456854)

[GeoLocator class 7](#_Toc437456855)

[Location Authorisatie 7](#_Toc437456856)

[Locatie wijzigingen opvangen 7](#_Toc437456857)

[Export database 8](#_Toc437456858)

[StorageFile 9](#_Toc437456859)

[Visualisatie van de Trk 9](#_Toc437456860)

## ER- schema opgave

We ontwikkelen een toepassing om een GPS tracker app te maken. Met deze app zijn we in staat om een afgelegd parcours te registreren.

### ER- schema beschrijving

In de database zullen we Tracks registreren. Elke Track is een afgelegd parcours met een titel en een tijdstip waarop de Track werd geregistreerd.

Elke Track heeft 0, 1 of meer TrackPoints. Een TrackPoint hoort bij exact 1 Track. Voor elk TrackPoint registreren we het moment waarop dit TrackPoint werd geregistreerd, tezamen met de Latitude en Longitude.

Ontwikkel (op papier) het ER- schema voor bovenstaande probleemstelling.

## Database structuur

Pas de conversie regels toe op uw ER schema en ontwikkel (op papier) een database structuur.

## SQLite

[SQLite](http://sqlite.org/) is een database systeem dat onder andere veel wordt gebruikt in mobiele omgevingen: elke Android installatie komt by default met een SQLite installatie. Ook voor Windows oplossingen (ook voor UWP solutions) en IOs kan SQLite geïnstalleerd worden.

De webpagina [Appropriate uses for SQLite](http://sqlite.org/whentouse.html) beschrijft meer in detail de omstandigheden waarin het gebruik van SQLite aan- of af te raden is.

### Project introductie: GPXTool

We zullen een app ontwikkelen waarmee we een afgelegde route kunnen registreren (in een database) en visualiseren in een map- control. Extra features die in deze les niet aan bod komen zijn het exporteren van de data als GPX- file en het aanpassen van de routekleur aan de snelheid van elk afgelegd route stukje (sommige stukken was u sneller dan andere stukken).

#### Opzet van het project (u krijgt dit van ons)

* Start een nieuw UWP project. Voor de eenvoud van het project zullen we geen gebruik maken van deelprojecten maar alles organiseren via folders
  + Voorzie de folder Models, Data, ViewModels, Pages, …
* Installeer het package MVVMLight
* Voorzie een NewTrackPage met bijhorend ViewModel NewTrackViewModel
* Voorzie een ApplicationViewModel met een CurrentPage die de NewTrackPage toont
* Ontwikkel IocContainer (in de Services folder) zoals gezien in de voorbije lessen. Voorzie een static resource in App.xaml. Via deze static resource zullen de Page DataContext waardes ingesteld worden.

### SQLite in een UWP project

Om uw project klaar te maken om met SQLite te werken moeten we een aantal stappen doorlopen. Deze stappen worden in detail beschreven in de tekst [Using SQLite in Windows 10 Universal apps](http://igrali.com/2015/05/01/using-sqlite-in-windows-10-universal-apps/). Het heeft weinig (geen) zin om die tekst over te tikken ☺. Waar u moet op letten:

* U moet een vsix package installeren. Dit is een systeem installatie die slechts één keer moet gebeuren. Deze installatie maakt een dll beschikbaar voor dit en andere projecten.
* Installeer het in de tekst vermeldde package (anders dan in de tekst vermeld wordt er slechts één extra dll in de referenties opgenomen).
* Voeg de zopas geïnstalleerde dll toe aan het project (zie bovenstaande tekst). Het betreft de dll **SQLite for Universal app** platform (zie bovenstaande tekst)

Indien u bovenstaande stappen correct uitvoerde bent u klaar om SQLite in uw project te gebruiken.

#### SQLiteService

Om alle SQLite code van het project op één plaats te coderen voorzien we de klasse SQLiteService in de Data folder (er is ook een Service folder, maar deze behoort bij de UI van de solution. Indien we de solution zouden onderverdelen in projecten zou de SQLiteService klasse in het Data project geplaatst worden omdat deze klasse het ophalen en wegschrijven van de gegevens implementeert).

#### Initialisatie van de SQLite omgeving

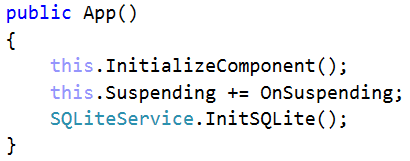
De initialisatie van de SQLite omgeving zal de nodige tabellen definiëren (indien ze nog niet bestaan).

De [SQLite create table statement beschrijving](http://www.sqlite.org/lang_createtable.html) is nogal formeel. Een meer praktische benadering waaruit u onmiddellijk kan copiëren vindt u in [SQLite create table and INSERT syntax examples](http://alvinalexander.com/android/sqlite-create-table-insert-syntax-examples).

* Een int primary key veld is bij default autoincrement (het systeem zorgt voor goede waardes).
* SQLite kent geen datetime type. U moet elk tijdstip als string wegschrijven. U kiest uiteraard zelf uw formaat, ik zou aanraden dat u trk.Tijdstip.ToString("yyyy MM dd HH:mm:ss") gebruikt (in de veronderstelling dat Tijdstip een DateTime property is op een Trk object.

Voor deze tabel geven we nog de voorbeeld code waarmee u de tabel kan aanmaken:



* Elke database operatie die we uitvoeren zal de locatie van de database file moeten kennen. Eerder dan deze berekening telkens te copiëren stop ik deze in een property. Bij wijziging moet deze dan maar op één plaats aangepast worden.
* Bemerk de ***if not exists*** clause die vermijdt dat de table gedropped zou worden indien ze reeds bestaat (dit zou de eerder geregistreerde gegevens verwijderen).
* Voorzie op basis van uw eigen tabel structuur ook de code voor CreateTrkPtTB die een TrackPoint tabel creeërt.
* We zullen deze code (die de database file en de tabellen aanmaakt indien ze nog niet bestaan) uitvoeren bij het opstarten van de applicatie. We kunnen de oproep plaatsen in de constructor van de App- klasse (in de App.xaml.cs file):

#### SQLite tabellen via klasses

SQLite laat toe om tabellen te definiëren door een type mee te geven. Hiervoor zijn dataannotaties vereist. Ikzelf verwacht deze code first manier van werken te gebruiken wanneer Entity Framework 7 gereleased wordt (begin 2016?).

* De moeilijkheid is onder andere het uitzoeken van de foreign key attributen

## Project uitwerking

### Models

* Implementeer voor elke tabel een model klasse. Deze moet natuurlijk erven van een ModelBaseClass die u voorziet. Ik verwacht een klasse Trk (Track) en TrkPt (TrackPoint).

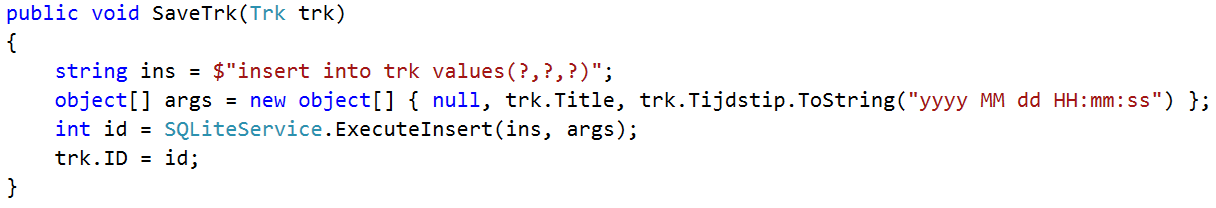
### Repositories (gedeeltelijk)

Voorzie voor elk model een Repository klasse. Deze erft in dit geval van geen andere klasse omdat we geen gemeenschappelijke code zullen terugvinden.

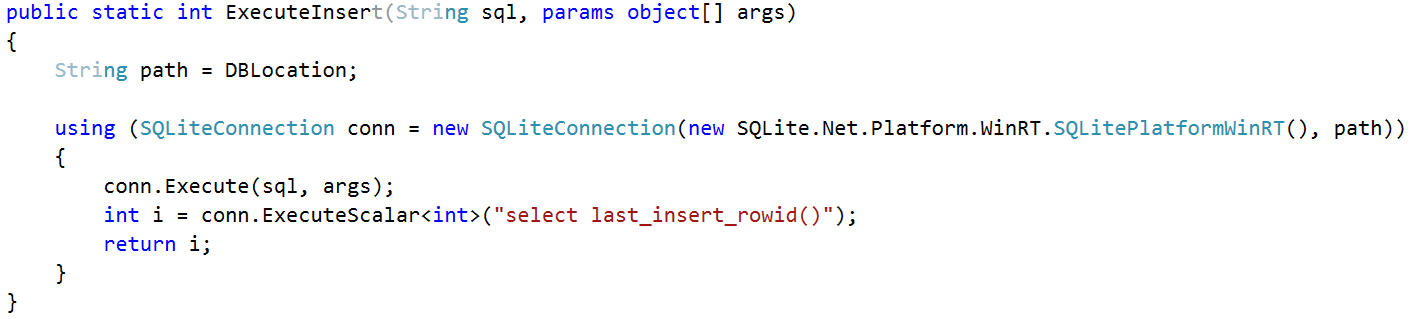
#### TrkRepository

##### Bewaren van Trks

Deze repository klasse behandelt de informatie uitwisseling met de database voor de Trk (Track) klasse. Een voorbeeld zegt meer dan 1000 woorden, zie hieronder voor de bewaarlogica:

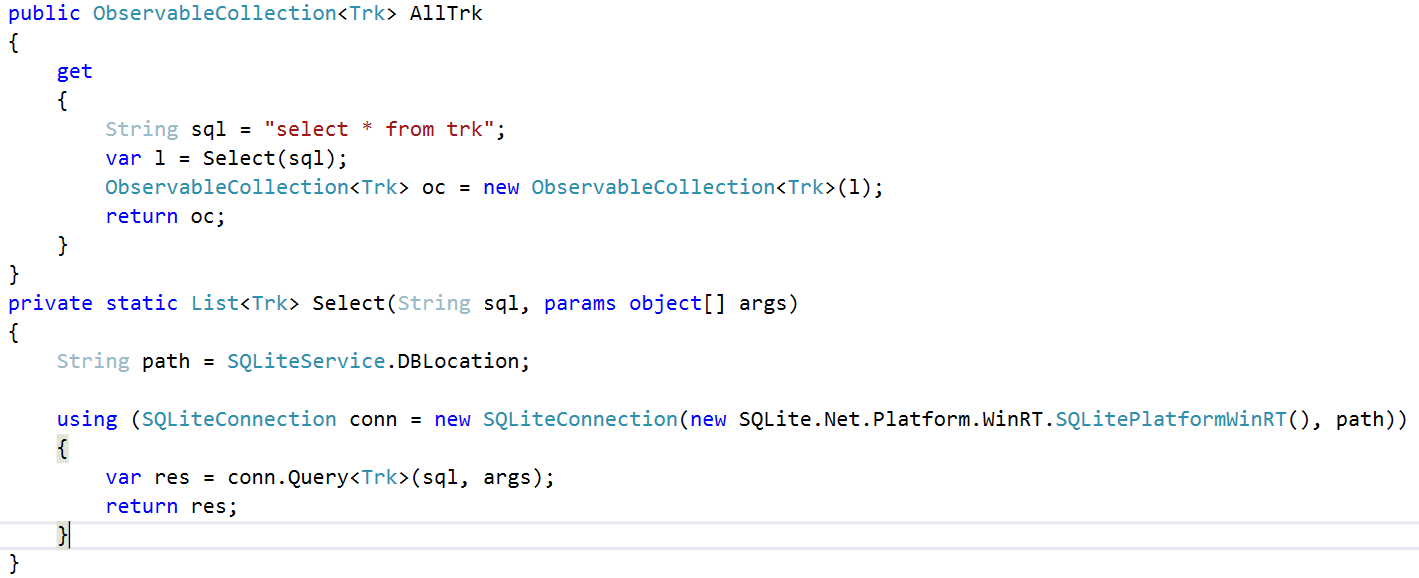


* We gebruiken parameters om geen SQL injection toe te laten (omdat we geen kolomnamen opgeven moeten de parameters in dezelfde volgorde als de kolommen in de tabel creatie worden meegegeven)
* De parameters zijn gewoon objecten in een array
* De methode ExcecuteInsert van de SQLiteService klasse is een beetje bijzonder:

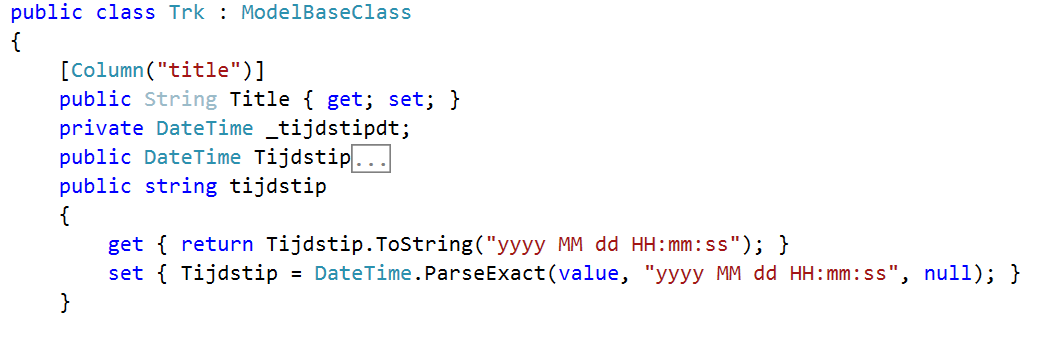


* Nadat een rij aan een tabel wordt toegevoegd wensen we zijn PK in de tabel te kennen (het ID). De select na de Execute selecteert de laatst toegevoegde rowid binnen deze connectie (indien u deze select in een nieuwe connectie uitvoert krijgt u het getal 0). Deze waarde geven we terug zodat ze als ID kan ingesteld worden op het object dat we in de tabel bewaarden.

##### Verzamelen van Trks



* Wanneer we een connectie ondervragen krijgen we een lijst van een welbepaald type terug (we hebben niet de ADO feature om een DataReader of DataTable terug te krijgen). Om deze reden kan de select code niet gemeenschappelijk zijn, omdat elke repository andere types in de lijst verwacht.
* Omdat ons viewmodel een ObservableCollection nodig heeft maken we hier onmiddellijk de conversie. Dit zou eventueel ook in het ViewModel zelf kunnen gebeuren.
* Opdat de mapping van de tabel kolomnamen en de klasse property namen zou gebeuren moeten deze identiek zijn (dit is niet het geval in mijn code omdat de C# code een Camel casing volgt en mijn tabel kolomnamen lower case werden gedefinieerd). Door gebruik te maken van DataAnnotaties kunnen we de mapping tussen tabel kolommen en klasse properties duiden:



* De property Title wordt gemapped op de kolom title;
* Bemerk dat de kolom tijdstip overeenstemt met de property tijdstip (type string). Er is ook een property Tijdstip (type DateTime) waarvan de waarde wordt bepaalt bij het toekennen van de string property (de string property wordt trouwens niet bewaard).

### Testen van de gedeeltelijke oplossing

#### New Track toevoegen aan de database

Wanneer we in de NewTrack page op ‘start new track’ klikken moet een nieuwe Trk aan de trk tabel worden toegevoegd:

* Voorzie op het NewTrackViewModel een RelayCommand NewTrackCMD dat we koppelen aan de NewTrack knop (via binding op de Command property)
* Als dit commando wordt uitgevoerd maken we een nieuw Trk object:
  + Title = ‘Trk test’
  + Tijdstip = DateTime.Now
* We vragen de repository om dit object te bewaren in de database.

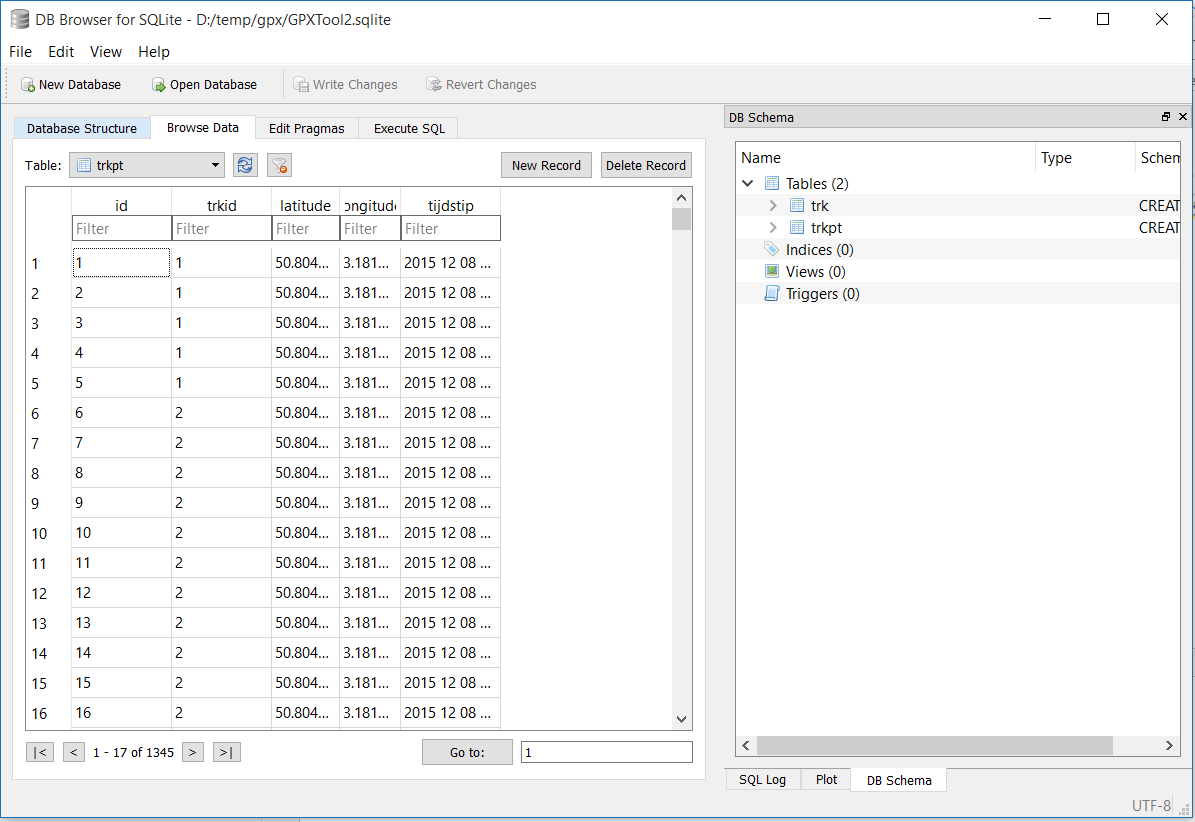
#### Verificatie van de database

We kunnen zelf code schrijven om de database na te kijken, maar er bestaat een [SQLiteBrowser](http://sqlitebrowser.org/) toepassing die toelaat om een SQLite file na te kijken

* Installeer deze app en kijk uw SQLite database na.
* Waar vindt u deze file? Plaats een breakpoint in uw code



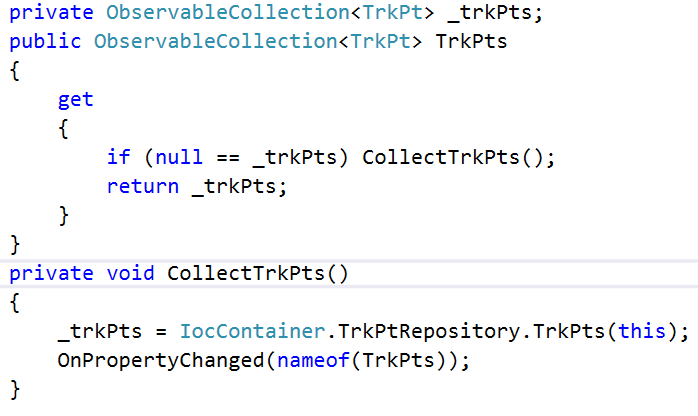
* Bovenstaande locatie is op de emulator, indien u de app op local machine opstart zal u de file vinden *ergens* in uw user data
* Kopieer de sqlite file naar een gemakkelijk toegankelijke plaats en open deze met de app. De app laat toe om de informatie in de SQLite file gemakkelijk na te pluizen:



Indien u het zopas toegevoegde Trk record niet terugvindt zal u uw code moeten corrigeren. Indien u uw record wel terugvindt bent u goed bezig ☺.

### Verdere Repositories uitwerking

#### TrkPtRepository

* Het bewaren van een TrkPt is gelijkaardig aan het bewaren van een Trk
* Het ophalen van TrkPts gebeurt altijd voor een Trk (de TrkId wordt in een where clause vermeld!)
* Gebruik het ophalen van TrkPts in de Trk klasse om de TrkPts property van deze klasse te implementeren. Dit is een *lazy* manier van werken: we zullen de TrkPts van een Trk maar ophalen wanneer ze nodig zijn.

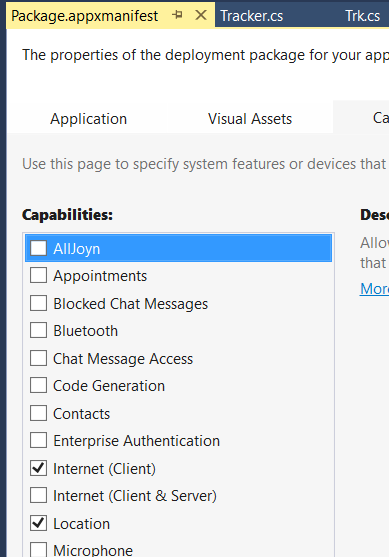
## Tracker implementatie

We hebben de infrastructuur opgezet om Trk en TrkPt objecten in een database te stoppen. We moeten nu nog de code voorzien om deze objecten aan te maken en te bewaren. Even herhalen dat de bedoeling is om een GPS tracker app te ontwikkelen: we willen onze positie op een gegeven moment registreren en deze op een kaart tonen als afgelegde route.

### Tracker klasse

We zullen alle nodige functionaliteit in een Tracker klasse stoppen. Deze klasse maken we in de Models folder en erft ook van ModelBaseClass.

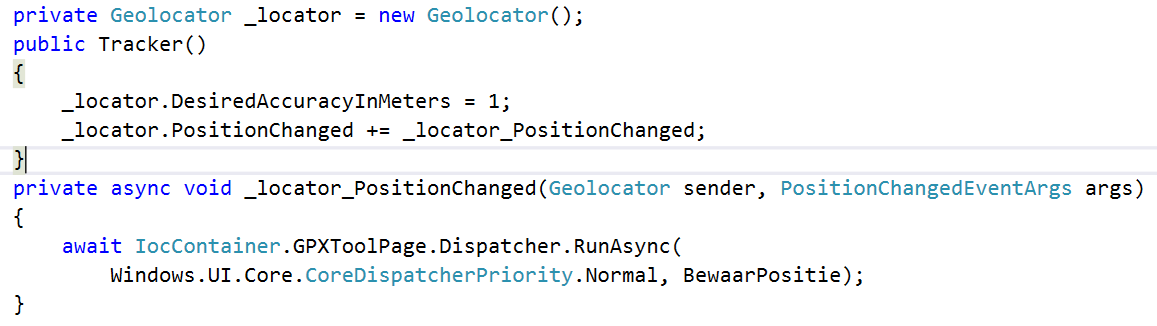
#### [GeoLocator class](https://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.geolocation.geolocator.aspx)

Deze klasse verschaft informatie omtrent de fysieke locatie van het toestel (uiteraard moet locatie opvraagbaar zijn).

#### Location Authorisatie

By default heeft een app zeer weinig rechten (sandboxed, zoals een website). U kan uw app bijkomende rechten laten aanvragen (bij installatie) door deze in de manifest (Package.appxmanifest) file aan te duiden. In dit geval moet onze app de locatie van het toestel kunnen opvragen:

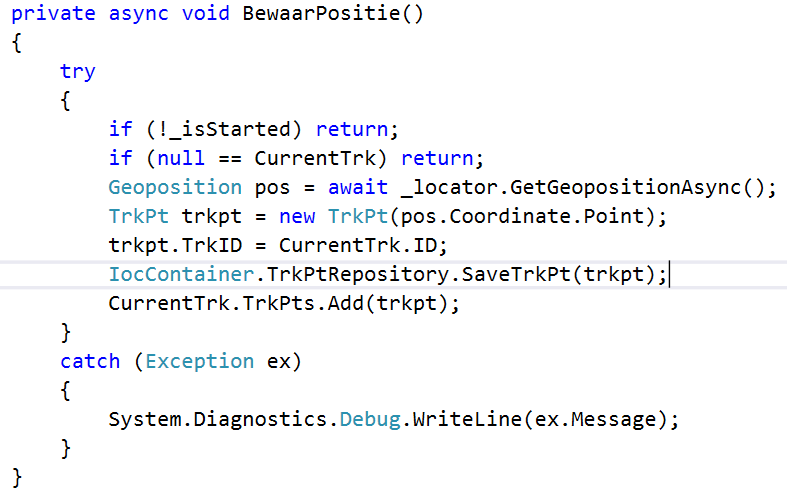
#### Locatie wijzigingen opvangen



Bovenstaande code toont een Tracker constructor die kan gebruikt worden om locatie wijzigingen op te vragen:

* Deze implementatie triggert het event wanneer de positie 1 meter wijzigt
* Het event zelf wordt afgevuurd op een andere thread. Om de UI te kunnen aanpassen moeten we de BewaarPositie methode uitvoeren op de UI- thread. Indien de methode BewaarPositie enkel in de database schrijft kan ze gewoon opgeroepen worden op de andere thread. Omdat de nieuwe positie ook gevisualiseerd zal worden (de UI wordt aangepast via een Binding) moeten we op de UI thread werken)
* De GeoLocator wordt als field gedefinieerd zodat hij niet door een garbage collector run wordt opgekuist. Indien dit wel zo zou zijn zouden zijn events niet meer gegenereerd worden (het GeoLocator object zou niet meer bestaan, dus zouden er ook geen events meer zijn).
* Vergeet niet de Tracker klasse te registreren in de IocContainer

Een voorbeeld BewaarPositie zou kunnen zijn:



* \_isStarted is een flag waarmee wordt bijgehouden of de tracker gestart is: indien de tracker niet gestart is willen we natuurlijk de locatie wijzigingen niet in de database bewaren.
* CurrentTrk is een Trk property waarin de huidige Trk wordt bijgehouden (u moet deze property zelf nog aanmaken). Indien deze niet is ingesteld voor de tracker moeten we geen positie wijzigingen bijhouden. Immers, elke positiewijziging in de database moet aan een Trk gekoppeld kunnen worden.
* Bemerk dat de klasse TrkPt een constructor heeft waaraan een Point wordt meegegeven. Daaruit wordt dan de Latitude en Longitude opgehaald
* De TrkId van een TrkPt wordt opgevraagd aan de CurrentTrk
* Nadat we het TrkPt bewaren in de database zullen we het toevoegen aan de lijst van TrkPts van de CurrentTrk.

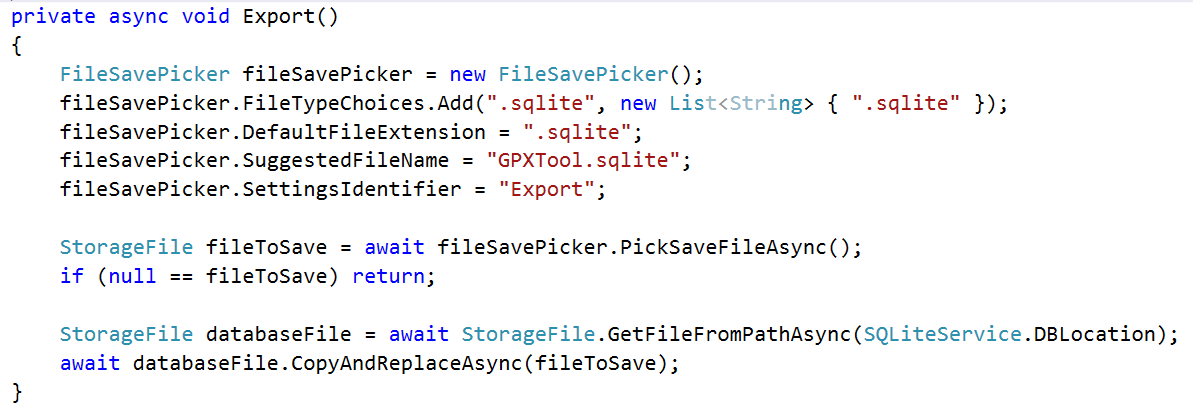
## Export database

Indien we op het toestel verschillende Tracks hebben verzameld is het interessant om deze te kunnen exporteren, zodat ze ook op een ander (groter scherm?) toestel gelezen kunnen worden.

### StorageFile



In een WPF toepassing gebruiken we bij FileIO de System.IO namespace. Deze heeft toegang tot het volledige filesysteem van het toestel. In een UWP app hebben we default enkel toegang tot de isolated storage van de app (een voor de app gereserveerde zone) en moet de gebruiker toegang geven om elders files te lezen om te schrijven. Deze toegang kan verleend worden via FilePicker dialogen. Om de database file te exporteren naar elders gebruiken we volgende code:



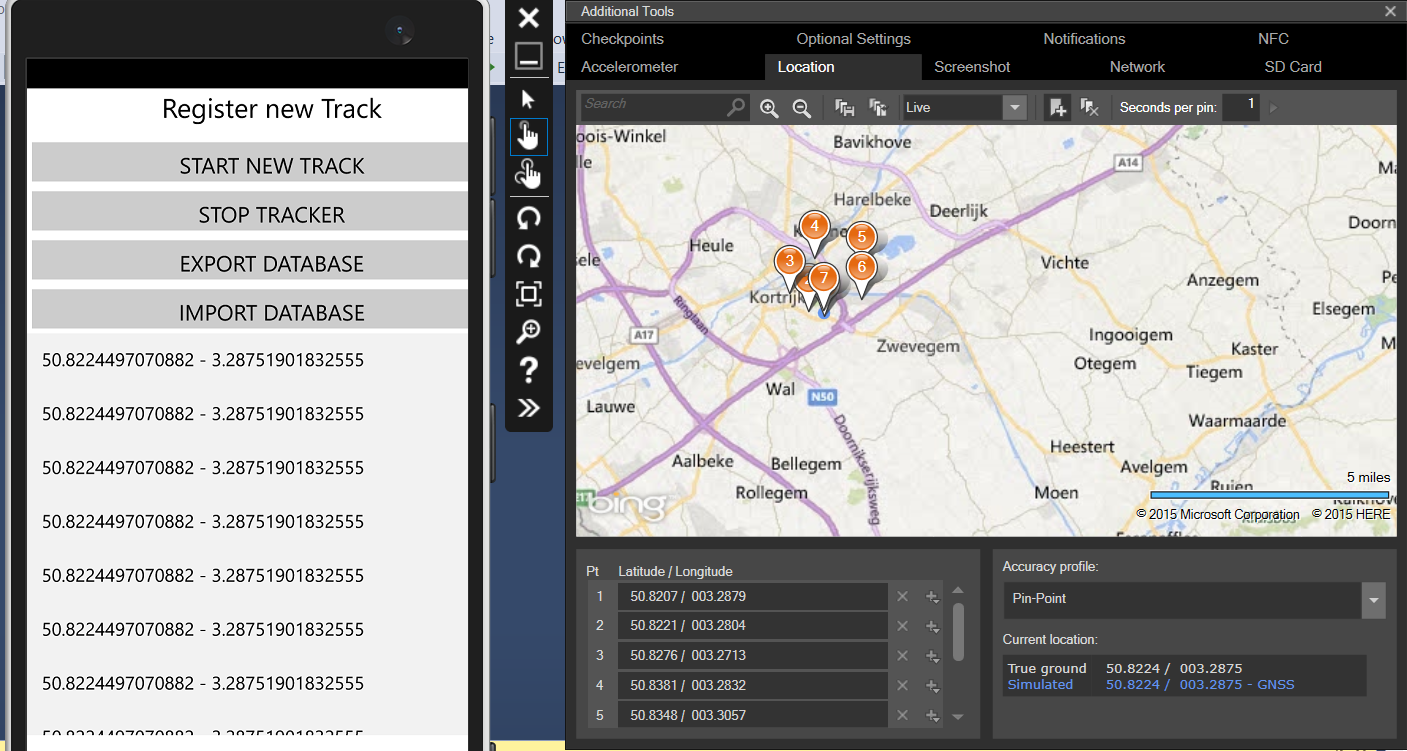
* De FileSavePicker dialoog laat de gebruiker een save file selecteren (eventueel op OneDrive). De geselecteerde file is een StorageFile object
* Bemerk dat quasi alle bewerking met StorageFile objecten asynchroon gebeuren!
* De gecopieerde file kan uiteraard gelezen worden met de SQLiteBrowser app die eerder werd geïnstalleerd.

## Visualisatie van de Trk

Het uiteindelijke doel van deze toepassing is het visualiseren van een Trk: welk parcours is er afgelegd? Het zal u wellicht niet verbazen dat we hiervoor terug de MapControl zullen gebruiken.

De code is op zich niet zo moeilijk, maar omdat deze labo les wellicht reeds overvol zit krijgt u de code bij het project. Eventuele kleine aanpassingen in klasse en property namen zijn voor rekening van de student.

## GeoLocator testen op de simulator



## Mogelijke uitbreidingen

* Ontwikkel een Settings pagina waar onder andere de import en export mogelijkheden beschikbaar worden gesteld. Ook instellingen zoals de gevoeligheid van de locator kan men hier instellen.
* Ontwikkel de mogelijheid om een track als GPX file te exporteren
* Ontwikkel de mogelijkheid om een GPX file als track te importeren
* Zoek mooie images voor in de commandbar