Op welke manier wordt de route- en traingingsinformatie opgeslagen?

Door:

* Maurice Pütz
* Sander Gelderman

Datum:

* 23-03-2020

Module:

* BM07 Cross Platform App Development

Inhoudsopgave

[Waarom moeten er gegevens opgeslagen worden? 2](#_Toc37683166)

[Opslagmethodes 2](#_Toc37683167)

[Cloud vs. Lokale database 2](#_Toc37683168)

[SQLite vs. Realm 3](#_Toc37683169)

[SQLite uitleg 3](#_Toc37683170)

[SQLite installatie 4](#_Toc37683171)

[Attributes 6](#_Toc37683172)

[Methodes 6](#_Toc37683173)

[Functies 7](#_Toc37683174)

[Verwijzingen 8](#_Toc37683175)

# Waarom moeten er gegevens opgeslagen worden?

In de applicatie voor slechtzienden en blinden moeten routes gelopen kunnen worden. De applicatie moet de begeleiders de mogelijkheid bieden om eigen routes aan te maken met bijbehorende zelfgemaakte oefeningen op specifieke locaties. De routes bestaan uit een aantal coördinaten. Op de coördinaten kan een oefening geplaatst worden. Er zullen daardoor routes, coördinaten en oefeningen opgeslagen moeten worden om de applicatie te voorzien van de benodigde functies.

# Opslagmethodes

Gegevens kunnen op meerdere manieren opgeslagen worden. Gegevens kunnen opgeslagen worden in files. App-specific storage houdt in dat files opgeslagen worden die alleen bedoeld zijn voor de gebruiker van de applicatie zelf. Shared storage houdt juist in dat bestanden worden opgeslagen met de intentie om files te delen met andere gebruikers van de app. Bij de eerste wordt een bestand opgeslagen die alleen bedoeld is voor de gebruiker van de app zelf (Developer android, 2020). Een andere manier om gegevens op te slaan is door gebruik te maken van preferences. Preferences gebruiken ook files, maar zijn bedoeld om zogenoemde key-values in op te slaan. Dat houdt in dat een stuk data opgeslagen wordt en gekoppeld wordt aan een key. Deze methode is voor de applicatie niet nuttig, omdat de data die voor de applicatie opgeslagen moeten worden, uit meerdere lagen bestaan. Er is voor de applicatie meer structuur nodig. Een andere optie is om gebruik te maken van een database. In een database wordt een verzameling georganiseerde en gestructureerde data opgeslagen. In de applicatie voor de Robert Coppes Stichting is een database de beste optie, omdat daarin de data gestructureerd wordt opgeslagen en daarmee de verzameling oefeningen, coördinaten en routes aan elkaar gekoppeld zijn.

# Cloud vs. Lokale database

Er zijn twee manieren om data in een database op te slaan, namelijk door middel van de cloud (Microsoft, 2018) of door middel van een lokale database (Microsoft, 2019). Het voordeel van cloud is dat de cloud provider zorgt voor de onderhoud. Hierdoor hoeft de stichting hier zelf minder rekening mee te houden. Een ander voordeel is dat er gemakkelijk meer of minder opslagruimte voor de applicatie vrijkomt. Daarnaast is er altijd toegang tot de data als er internetverbinding is (Labfolder, sd). Een nadeel is echter dat de data niet bereikt kan worden zonder internet. Als de internetverbinding niet sterk is, dan is het bereiken van de data lastiger. Daarnaast zal er nagedacht moeten worden over de verplaatsing van de data, mocht ervoor gekozen worden om de data ook op een andere plek te zetten. Het gebruik van een lokale server heeft echter ook grote voordelen. Gebruikers kunnen data via een lokale server vaak sneller ophalen en aanpassen. Daarnaast is er meer controle over het datasysteem. Aangezien de controle van de data in eigen handen is, is het eenvoudiger om back-ups te maken op diverse plekken. Het is ook veiliger om data op een lokale server te zetten, aangezien alleen de ontwikkelaars en de stichting zelf bij de data kunnen komen. Een nadeel van een lokale server kan zijn dat een lokale server geïnstalleerd moet worden. De installatie van SQLite en Realm als voornaamste lokale servers gebruikt voor Xamarin zijn echter heel eenvoudig. Voor de applicatie voor blinden en slechtzienden is ervoor gekozen om gebruik te maken van een lokale server. De applicatie heeft een relatief weinig ruimte nodig in een database. Voor de applicatie is het prettig om niet afhankelijk te zijn van een goede internetconnectie. Het is wenselijk dat begeleiders ook routes kunnen aanpassen zonder internetverbinding. Veiligheid speelt ook een belangrijke rol bij de keuze voor een lokale database. Op de standaard route na van de Robert Coppes Stichting moeten de ingevoerde routes alleen op het apparaat van de gebruiker zelf staan. De meest gebruikte lokale databases die Xamarin ondersteunt zijn Realm en SQLite. Er zal een afweging gemaakt worden welke database optie het meest geschikt is voor de applicatie.

# SQLite vs. Realm

SQLite en Realm zijn twee databaseoplossingen die bij een Xamarin project kunnen helpen om data lokaal op te slaan. Voor de database is het belangrijk dat gegevens snel opgehaald kunnen worden en dat de database goed onderhouden wordt voor langere periodes. Dhr. Cruijsen heeft onderzocht welke database sneller werkt en kwam tot de conclusie dat er nauwelijks verschil was tussen SQLite en Realm. In sommige testgevallen was Realm sneller en in andere gevallen was SQLite sneller. Hij benadrukte wel dat het belangrijk is dat de ontwikkelaar weet wat hij doet. Als er fouten gemaakt worden, kan dat vertraging veroorzaken bij SQLite (Cruijsen, 2016). Naast de snelheid heeft dhr. Cruijsen ook gekeken naar de verschillen in onderhoud. Hij benadrukt dat de afhankelijkheid van een database beperkt moet blijven. Het liefst zou er alleen in de klassen database communicatie plaatsvinden. Dit is gemakkelijk mogelijk in SQLite, maar is moeilijker te realiseren bij Realm, aangezien nieuwe objecten gecreëerd moeten worden via Realm.CreateObject. Daarnaast ondersteund Realm geen async (asynchronisch) methodes. Met async methodes kunnen taken uitgesteld worden. Dit zijn vaak langlopende bewerkingen, zoals het lezen van grote bestanden of een API-aanroep. Met deze methode kan de interface van een applicatie gewoon blijven werken zonder dat er eerst gewacht moet worden (Microsoft, 2019). Aangezien de applicatie vaak API-aanroepen uitvoert, is de verwachting dat SQLite de betere database is om data in op te slaan voor dit project.

# SQLite uitleg

SQLite is een database die een opzichzelfstaande, serverloze (lokaal op een telefoon) SQL-Database-Engine implementeert. De code van SQLite bevindt zich in het publieke domein en is dus voor iedereen gratis toegankelijk, bij privé en commercieel gebruik. SQLite is een embedded SQL-database engine, vergeleken met de meeste SQL databases heeft SQLite geen aparte server proces. SQLite leest en schrijft rechtstreeks naar gewone schijfbestanden, een volledige SQL-database met meerdere tabellen, triggers, views, etc. Deze informatie wordt op 1 schijfbestand geplaatst. Verder is het bestandsformaat zo gemaakt dat het cross-platform is. Zelfs met alle functies in gebruik kan het database bestand kleiner zijn dan 600KB, afhangen van de compiler optimalisatie instellingen van het desbetreffende platform (64-bit zal groter zijn). SQLite loopt sneller bij een grotere hoeveelheid aan geheugen (dat beschikbaar wordt gesteld aan de database), ook al is de performance ook vrij goed in omgevingen met weinig geheugen.

SQLite wordt voor elke release zorgvuldig getest. Een deel van de broncode is uitsluitend bestemd voor testen en verificatie. Een geautomatiseerde testdienst voert miljoenen testcases uit met elk miljoenen individuele SQL-instructies om er zeker van te zijn dat alles werkt hoe het hoort te werken. Zoals eerder vernoemd is het gebruik van SQLite gratis, verder is er nog professionele hulp waar gebruik van gemaakt kan worden, deze brengt wel kosten met zich mee.

SQLite

<https://docs.microsoft.com/en-us/xamarin/android/data-cloud/data-access/using-sqlite-orm>

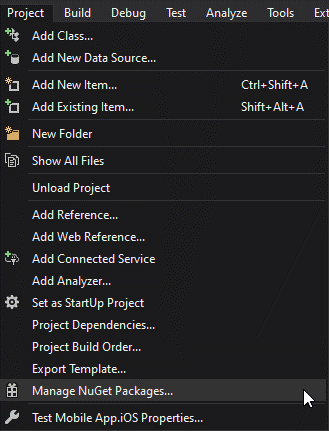
<https://docs.microsoft.com/en-us/xamarin/xamarin-forms/data-cloud/data/databases>

<https://www.sqlite.org>

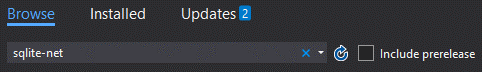
## SQLite installatie

Hier een uitleg over hoe je SQLite toevoegt aan je project.

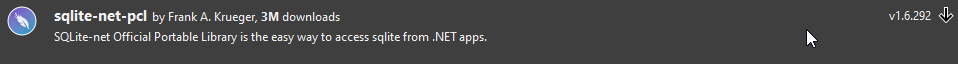
Stap 1: Open je project, klik vervolgens linksboven op project en klik daarna op manage NuGet packages.



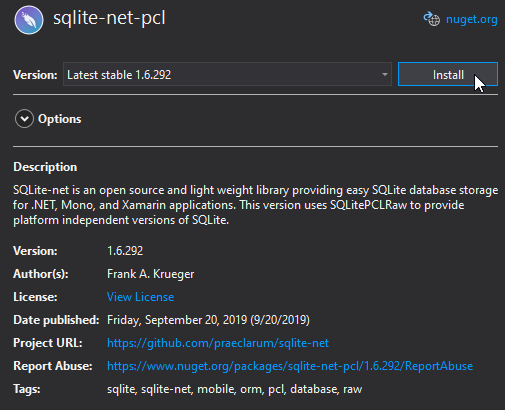
Stap 2: in de packet manager click je op browse en zoek je naar “sqlite-net-pcl”



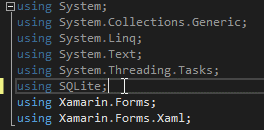
Stap 3: Klik op sqlite-net-pcl door Frank A. Krueger. Let op dat je het correcte package selecteert, deze heeft momenteel 3 miljoen downloads en in de beschrijving staat “acces sqlite from .NET apps”.



Stap 4: Let op dat je de goede versie hebt geselecteerd (1.6.292 is de laatste stabiele versie) en klik vervolgens op install.



Stap 5: om er vervolgens gebruik van te maken hoef je alleen nog maar het package aan te roepen in je code, dat doe je door “using SQLite;” boven aan je code toe te voegen (alleen waar data acces nodig is). Verdere uitleg in de link hieronder.

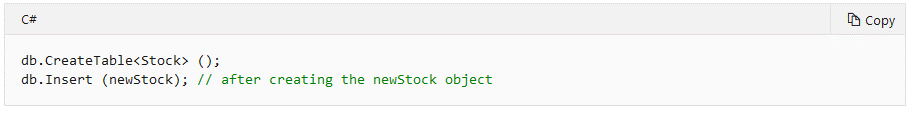


Korte werking:

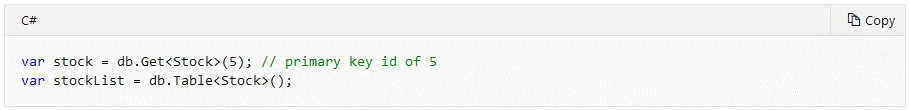
Stap 1: maak een lege database aan.



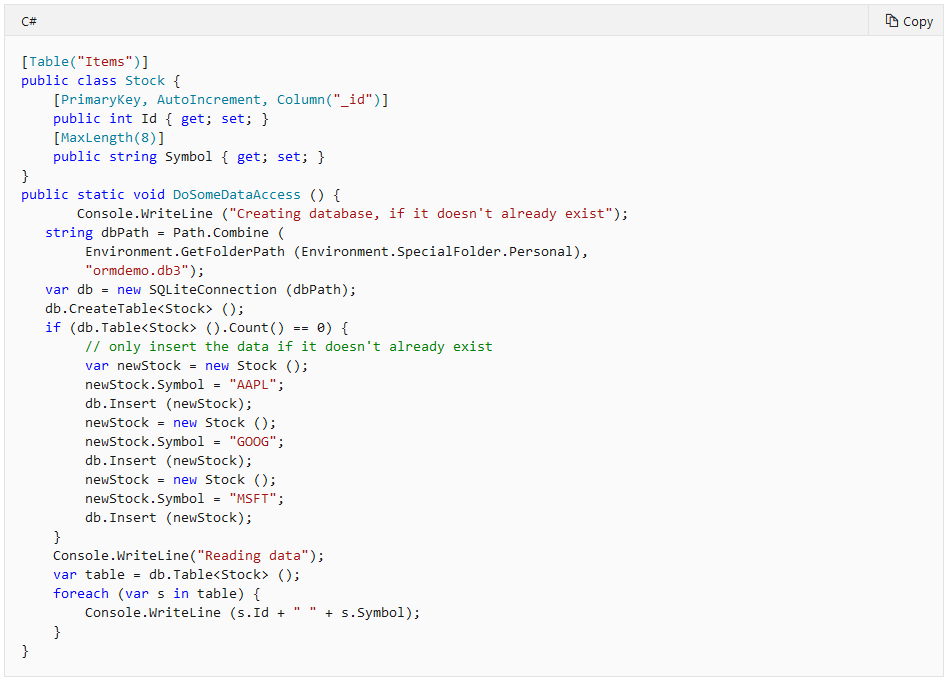
Stap 2: data opslaan, als je een connectie hebt gecreëerd worden databse commands uitgevoerd door de veschillende methoden.



Stap 3: Ontvang data, om data te ontvangen vanuit de databse gebruik je de volgende syntax.



Hier een groter voorbeeld:



## Attributes

Attributes voor SQLite zijn:

* PrimaryKey – deze attribute wordt gebruikt om een integer de primaire sleutel van de onderliggende tabel te geven.
* AutoIncrement – integer gaat omhoog met iedere nieuwe entry in de database.
* Column(name) – instellen van de benaming van een kolom
* Table(name) – klasse kan worden opgeslagen in table met gegeven naam.
* MaxLength(value) – beperkt de lengte van een tekst eigenschap binnen een database.
* Ignore – zo kan SQLite een property negeren. Als iets bijvoorbeeld niet in de database kan of moet worden opgeslagen.
* Unique – zo zijn de waarden in de onderliggende database uniek

## Methodes

De methodes die SQLite biedt zijn:

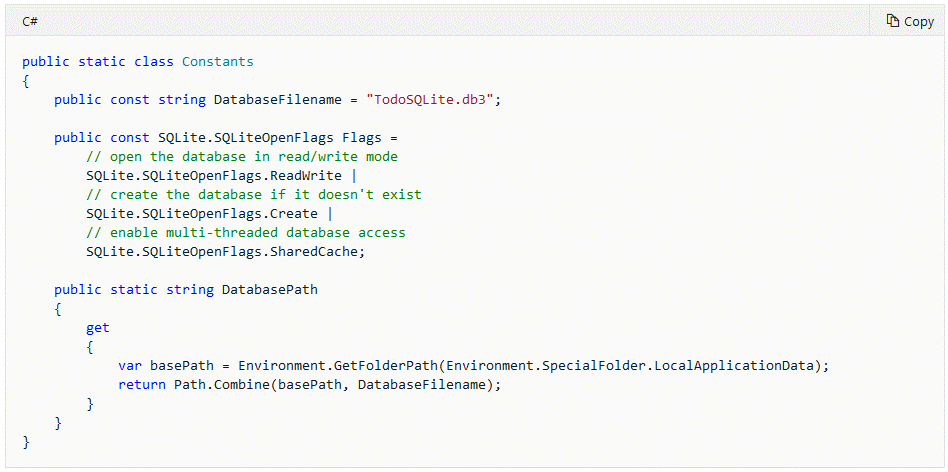
* Insert – Voeg een nieuw object toe aan de database.
* Get<T> - vraag een object aan op basis van de primary key.
* Table<T> - Geeft alle objecten in een table
* Delete – Verwijder een object op basis van de primary key.
* Query<T> - Voert een query uit die een aantal rijen retuned.
* Execute – Gebruik dit (en niet querry) als je geen rijen verwacht van sql (Bijvoorbeeld zoals, Insert, Update en Delete).

## Functies

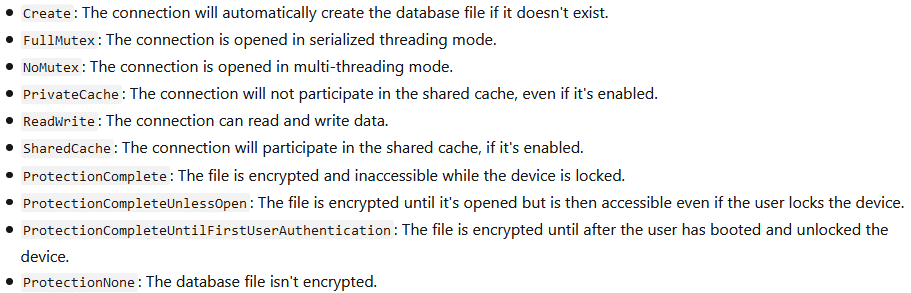
SQLite biedt diverse functies aan.

<https://docs.microsoft.com/en-us/xamarin/xamarin-forms/data-cloud/data/databases>

Gebruikelijke configuratie data:



Als de database op Read/Write staat (SQLiteOpenFlags) dan kunnen de volgende waarden gebruikt worden:



IOS : <https://docs.microsoft.com/en-us/xamarin/ios/data-cloud/data/using-sqlite-orm>

Android : <https://docs.microsoft.com/en-us/xamarin/android/data-cloud/data-access/using-sqlite-orm>

# Verwijzingen

Cruijsen, G. v. (2016, mei 17). *Xamarin apps: Sqlite vs Realm. What’s the best mobile DB solution?* Opgehaald van Mobile first cloud first: https://mobilefirstcloudfirst.net/2016/05/xamarin-apps-sqlite-vs-realm-whats-the-best-mobile-db-solution/

Developer android. (2020, maart 19). *Data and file storage overview*. Opgehaald van developer android: https://developer.android.com/training/data-storage#file-best-practices

Labfolder. (sd). *Cloud vs local Server – Where should you store your data?* Opgehaald van Labfolder: https://www.labfolder.com/cloud-vs-local-server/

Microsoft. (2018, april 25). *Data and Cloud Services*. Opgehaald van Microsoft Xamarin: https://docs.microsoft.com/en-us/xamarin/android/data-cloud/

Microsoft. (2019, Maart 18). *Asynchronous programming with async and await*. Opgehaald van Microsoft: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/

Microsoft. (2019, mei 12). *Xamarin.Forms Local Databases*. Opgehaald van Microsoft Xamarin: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/data-cloud/data/databases