# Database Example

Go to Tools, Nuget Packet Manager -> Package Manager Console. At the PM> Prompt, Type:

Install-Package sqlite-net-pcl -Version 1.4.118

Right click the solution, Manage Nuget Packages for Solution, click the installed tab, choose the sqlite-net-pcl package and check both other projects and choose install.

Add interface to App class at the end in the same file as App:

public interface IFileHelper

{

string GetLocalFilePath(string filename);

}

Add FileHelper.cs to each other project:

Android: (replace namespace with whatever the namespace is supposed to be for current solution)

using System.IO;

using Xamarin.Forms;

[assembly: Dependency(typeof(XamarinFuelTracker.Droid.FileHelper))]

namespace XamarinFuelTracker.Droid

{

public class FileHelper : IFileHelper

{

public string GetLocalFilePath(string filename)

{

string path = System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal);

return Path.Combine(path, filename);

}

}

}

iOS:

using System.IO;

using Xamarin.Forms;

[assembly: Dependency(typeof(XamarinFuelTracker.iOS.FileHelper))]

namespace XamarinFuelTracker.iOS

{

public class FileHelper : IFileHelper

{

public string GetLocalFilePath(string filename)

{

string docFolder = Environment.GetFolderPath(Environment.SpecialFolder.Personal);

string libFolder = Path.Combine(docFolder, "..", "Library", "Databases");

if (!Directory.Exists(libFolder))

{

Directory.CreateDirectory(libFolder);

}

return Path.Combine(libFolder, filename);

}

}

}

## Fuel Purchase Class

This will represent a fuel purchase, note the annotations for primary key etc:

using SQLite;

namespace XamBookDatabase

{

public class FuelPurchase

{

[PrimaryKey, AutoIncrement]

public int ID { get; set; }

public DateTime date { get; set; }

public double litres { get; set; }

public double cost { get; set; }

}

}

## Fuel Helper Database Class

This can just go inside the App class, or another file. Doesn’t really matter. You’ll need to import a few libraries, like using SQLite; , but the errors will appear to warn you:

public class Fuel\_Database

{

readonly SQLiteConnection database;

public Fuel\_Database(string dbPath)

{

database = new SQLiteConnection(dbPath);

//database.DropTable<FuelPurchase>(); // can call this to drop if needed

database.CreateTable<FuelPurchase>(); // won’t do anything if already exists

if(database.Table<FuelPurchase>().Count()==0) //if no records make one

{

// this will get the next key

FuelPurchase purchase = new FuelPurchase();

purchase.cost = 15;

purchase.date = new DateTime(2018, 1, 15);

purchase.litres = 15;

SaveItem(purchase);

}

}

public List<FuelPurchase> GetItems()

{

return database.Table<FuelPurchase>().ToList<FuelPurchase>();

}

public List<FuelPurchase> GetItemsOverTen()

{

return database.Query<FuelPurchase>("SELECT \* FROM [FuelPurchase] WHERE [litres] > 10");

}

public FuelPurchase GetItem(int id)

{

return database.Table<FuelPurchase>().Where(i => i.ID == id).FirstOrDefault();

}

public int SaveItem(FuelPurchase item)

{

if (item.ID != 0)

{

return database.Update(item);

}

else

{

return database.Insert(item);

}

}

public int DeleteItemAsync(FuelPurchase item)

{

return database.Delete(item);

}

}

## App Class Properties

static Fuel\_Database database;

public static Fuel\_Database Database

{

get

{

if (database == null)

{

database = new Fuel\_Database (DependencyService.Get<IFileHelper>().GetLocalFilePath("FuelSQLite.db3"));

}

return database;

}

}

This property will be accessible from anywhere in the application as App.Database. We will use the database variable to do queries and so on. We’ll be calling the “Dependency Service” to find out which platform’s file location to use.

After Initialize Component:

database = Database;

EntryCell eLitres = new EntryCell { Label = "Litres:" };

EntryCell eDate = new EntryCell { Label = "Date:" };

EntryCell eCost = new EntryCell { Label = "Cost:" };

EntryCell eID = new EntryCell { Label = "ID:" };

var btnSearch = new Button { Text = "Read" };

btnSearch.Clicked += (sender, e) =>

{

FuelPurchase purchase = database.GetItem(Convert.ToInt32(eID.Text));

eID.Text = purchase.ID.ToString();

eDate.Text = purchase.date.ToString();

eLitres.Text = purchase.litres.ToString();

eCost.Text = purchase.cost.ToString();

};

MainPage = new ContentPage

{

Content = new StackLayout

{

Spacing = 25, // this will alter distance between elements

Padding = 75, // this will alter distance from side of frame

Children =

{ new TableView{Intent = TableIntent.Form, Root =

new TableRoot{

new TableSection("Fuel Purchase"){eID, eDate, eLitres, eCost } }},

new StackLayout{Orientation = StackOrientation.Horizontal,

HorizontalOptions = LayoutOptions.Center,

Children = {btnNew, btnDelete, btnSave, btnSearch } }

},

},

};

You’ll need to remove a few buttons that haven’t been defined yet (next heading), then you can try it out. Should be able to search for ID #1 since it is added by default.

## Insert, Update, Delete

In the App constructor, after the search button is defined, add:

var btnNew = new Button { Text = "New" };

btnNew.Clicked += (sender, e) => {

eID.Text = "0"; eCost.Text = "";

eDate.Text = ""; eLitres.Text = "";

};

var btnDelete = new Button { Text = "Delete" };

btnDelete.Clicked += (sender, e) =>

{ database.DeleteItemAsync(database.GetItem(Convert.ToInt32(eID.Text)));

// add code to clear text boxes too maybe

};

var btnSave = new Button { Text = "Save" };

btnSave.Clicked += (sender, e) =>

{

FuelPurchase purchase = new FuelPurchase()

{

ID = Convert.ToInt32(eID.Text),

cost = Convert.ToDouble(eCost.Text),

litres = Convert.ToDouble(eLitres.Text),

date = Convert.ToDateTime(eDate.Text)

};

database.SaveItem(purchase);

};

Also will need to add to the layout somewhere.

Right now there are some issues with functionality, clearing of text boxes, trying to navigate when nothing there, deleting an item that doesn’t exist, etc. Lots of possible exceptions.