Xamarin

TODO

Contents

[Introduction 2](#_Toc113486717)

[Create a new GitHub branch 3](#_Toc113486718)

[Create TUFCv3 4](#_Toc113486719)

[Update Software 5](#_Toc113486720)

[Run the app 7](#_Toc113486721)

[Install MySqlConnector 8](#_Toc113486722)

[Temporary layout for MainPage() 11](#_Toc113486723)

[Test commands for SQL 11](#_Toc113486724)

[Layout 11](#_Toc113486725)

[Temporary XAML layout to test SQL commands … 12](#_Toc113486726)

[Create and execute a query 13](#_Toc113486727)

[Designing for layouts for small devices 15](#_Toc113486728)

[Return values using MySqlConnector 15](#_Toc113486729)

[Troubleshooting ‘Unable to connect to any of the specified MySQL Hosts’ 20](#_Toc113486730)

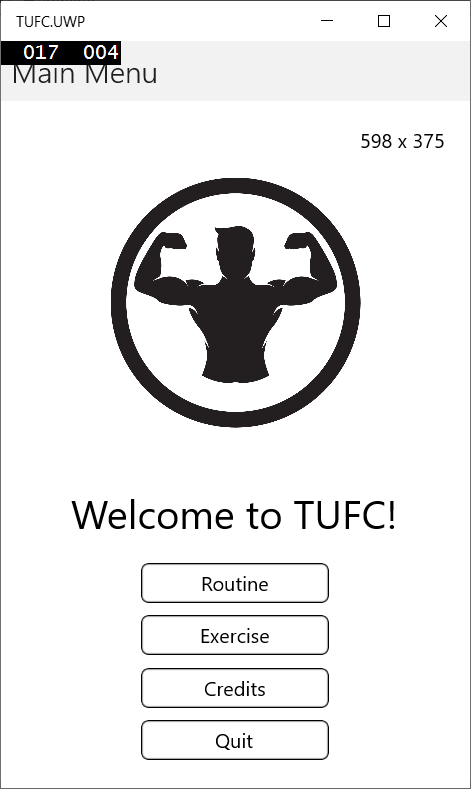
[MVVM: *‘Model, View, ViewModel’* 21](#_Toc113486731)

# Introduction

The Xamarin application **TUFCv3** *(The Ultimate Fitness Companion)*

is part of the project XWM *(Xamarin, Web server, MySQL)*

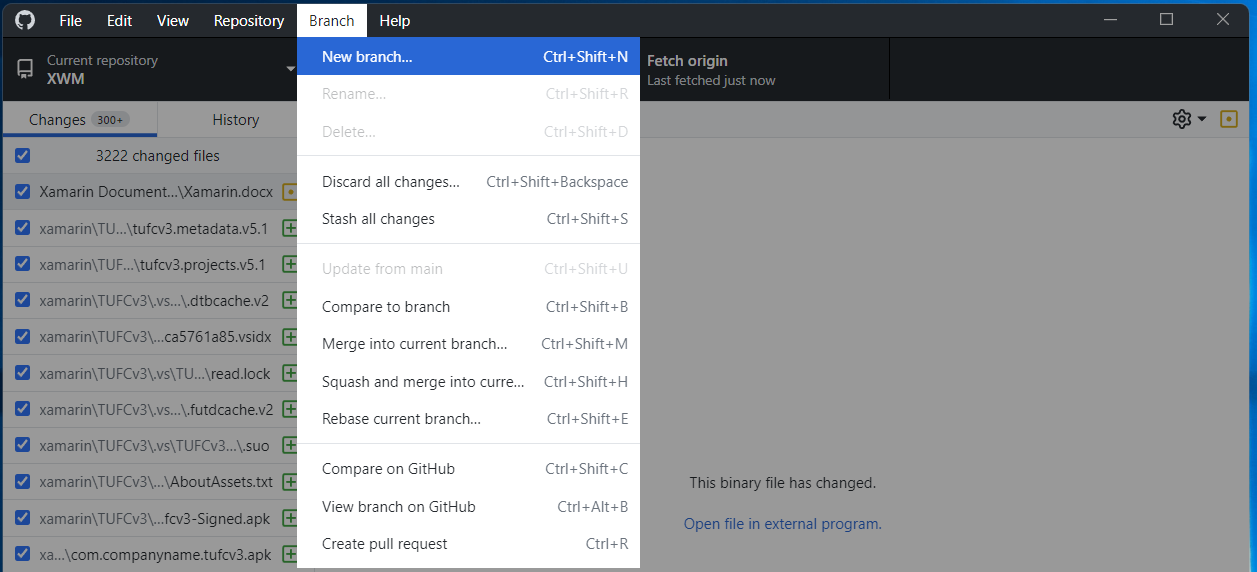
XWM is demonstration, including ‘how-to’ documentation,

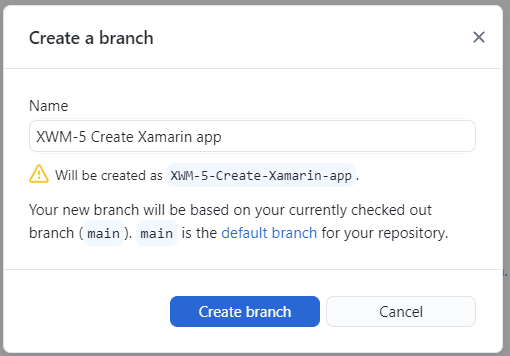
bringing these technologies together.

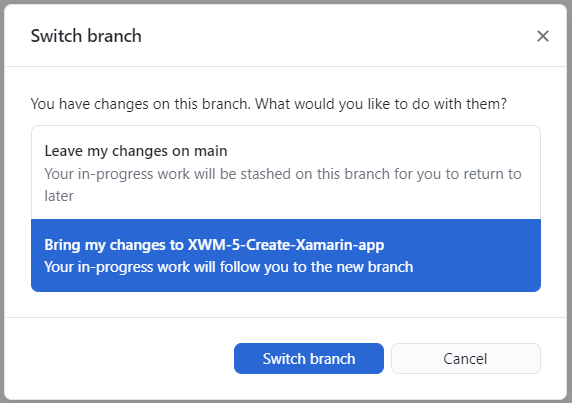
# Create a new GitHub branch

Before saving Xamarin software to GitHub

create a new branch.



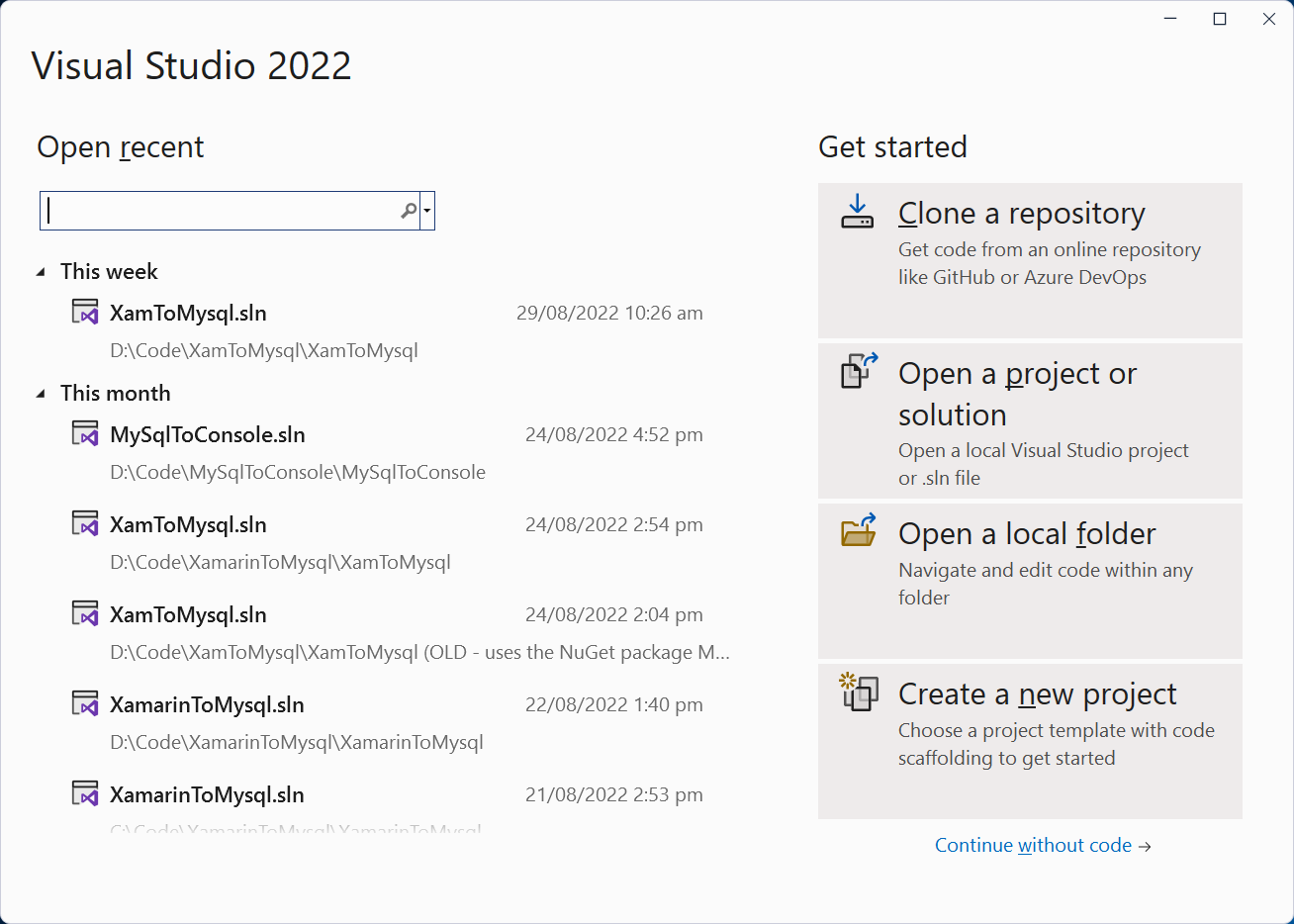


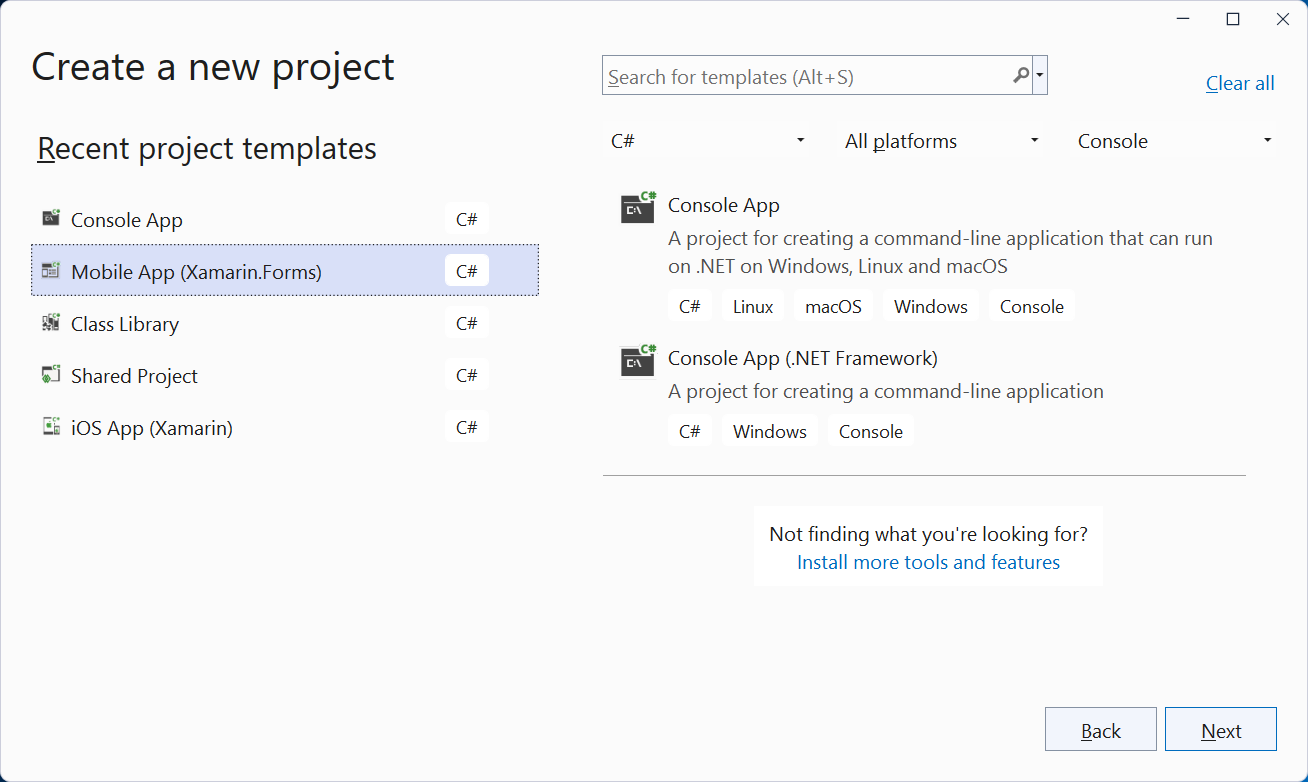


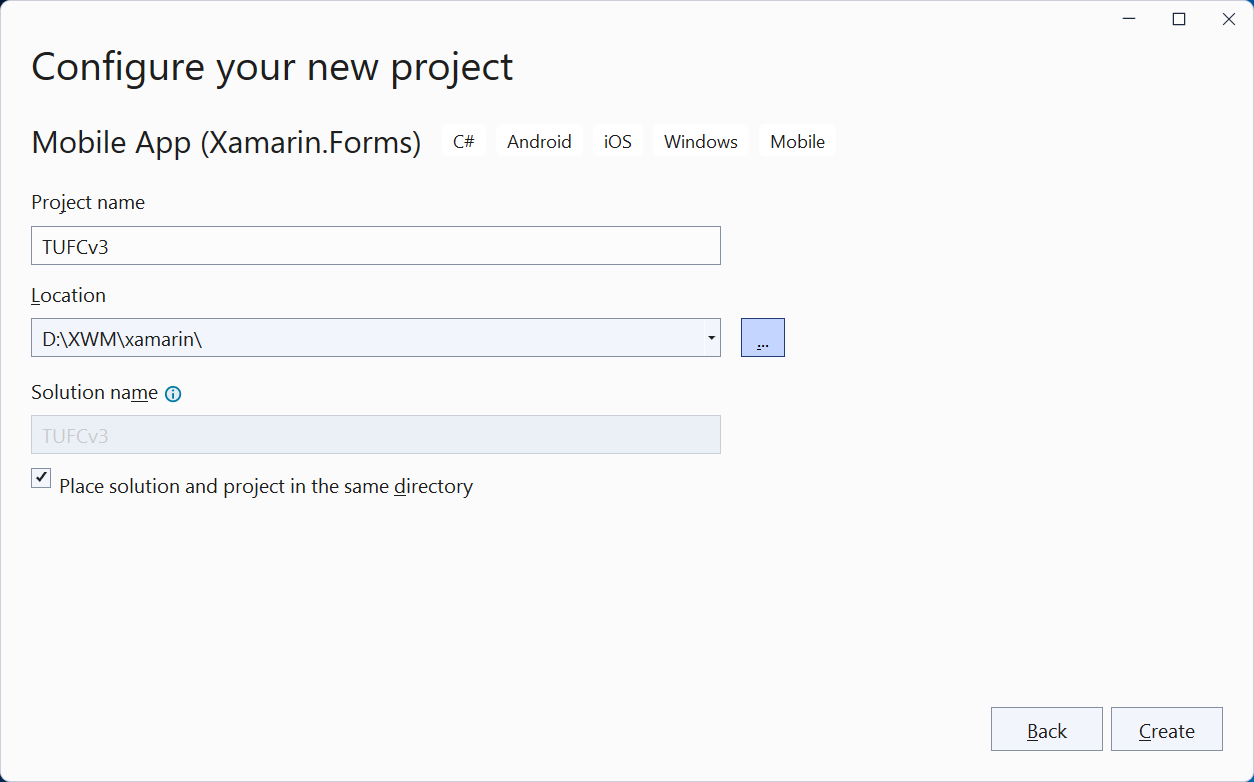
# Create TUFCv3

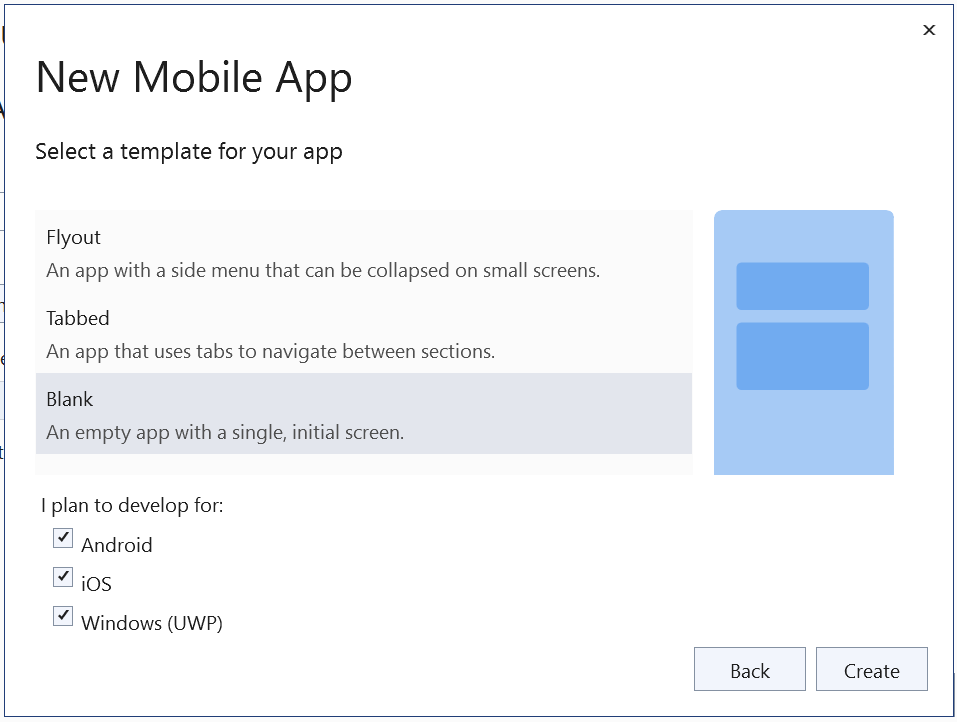
Create the Xamarin project TUFCv3

in the folder D:\XWM\xamarin









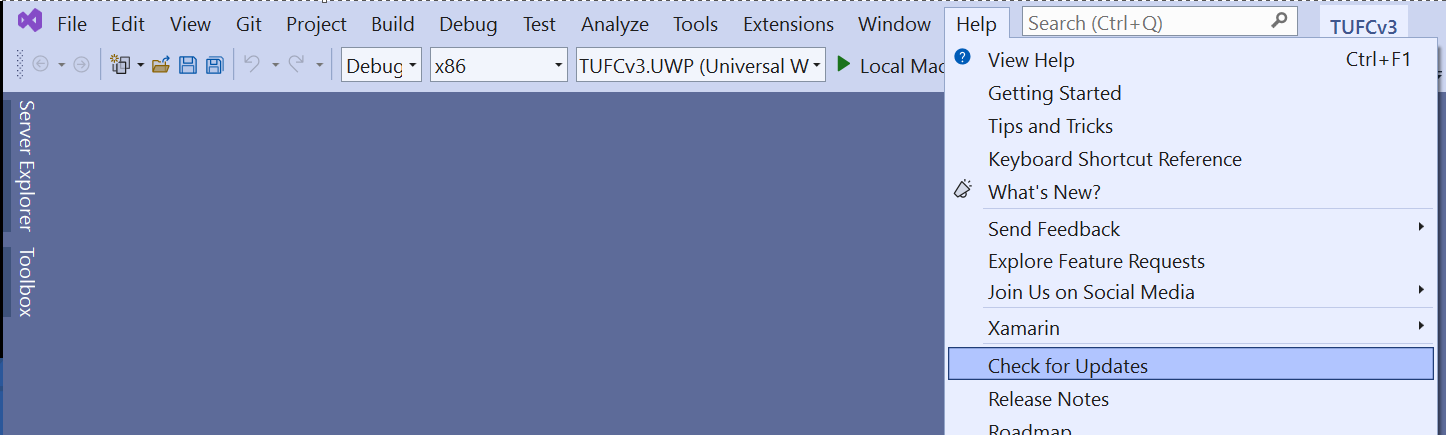
*Next, Update NuGet and run the application …*

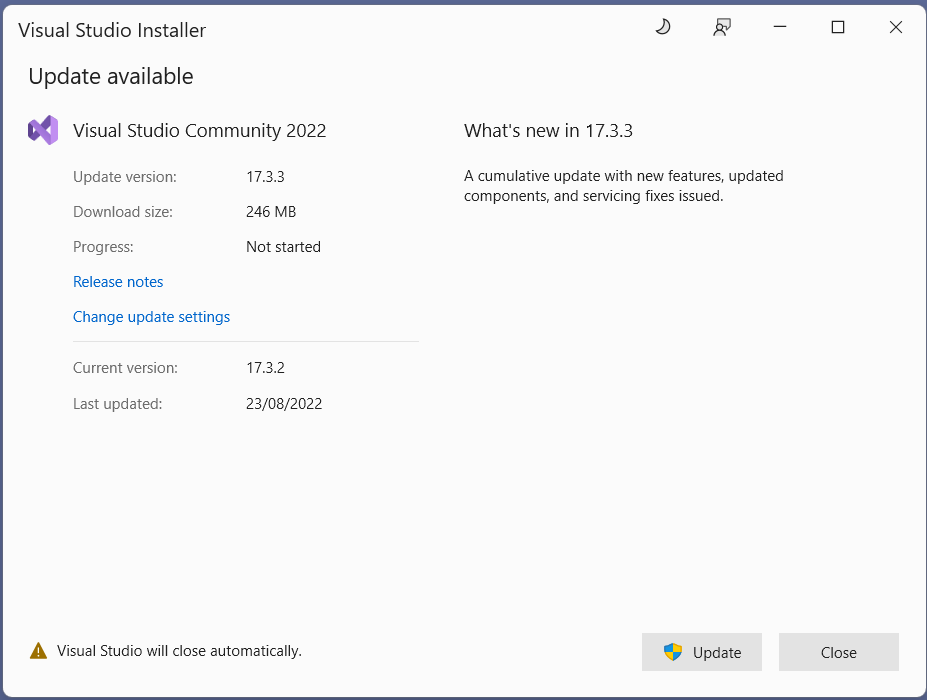
## Update Software

Before running the application check for updates on:

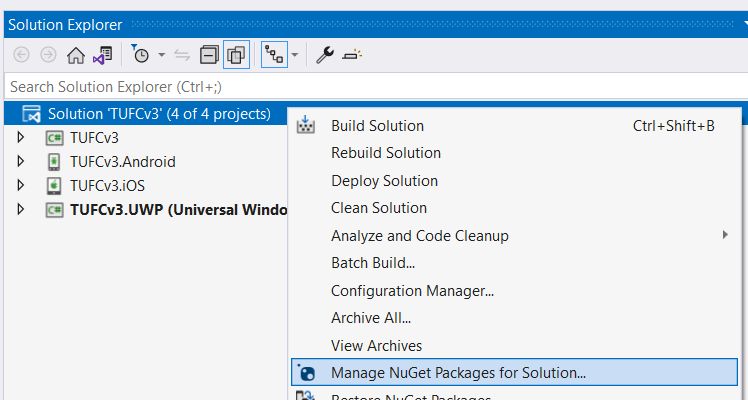
* Visual Studio
* NuGet
* Android
* Mac

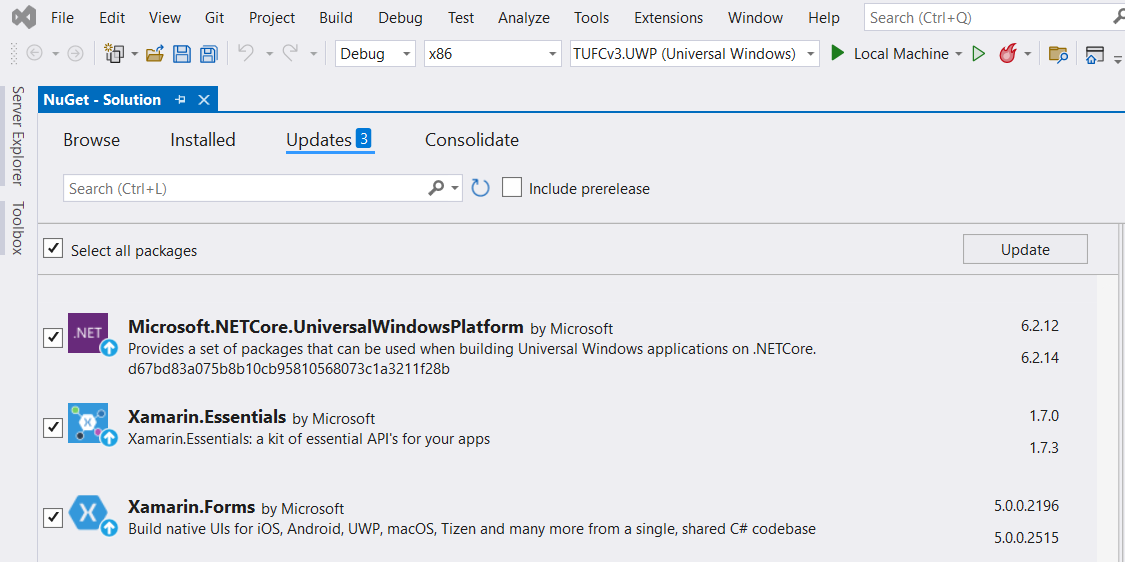
#### Update Visual Studio





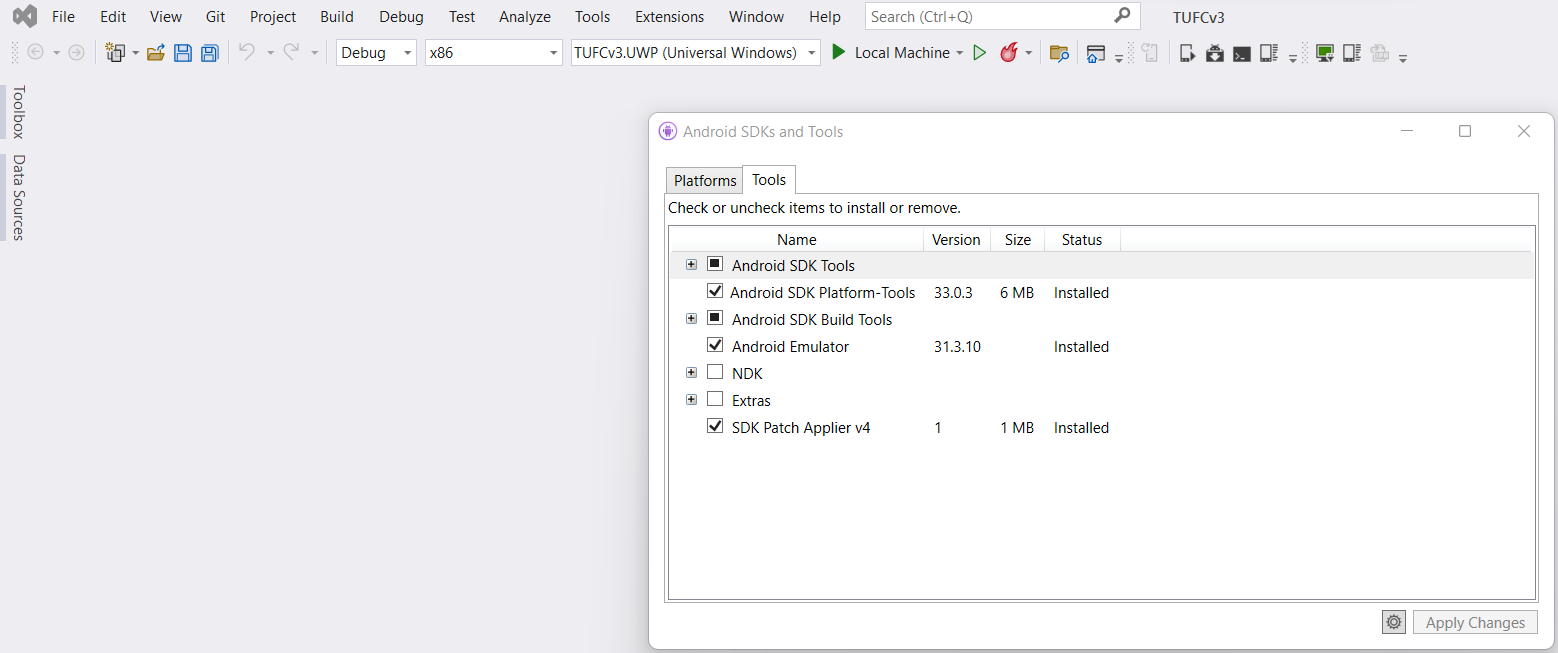
#### NuGet





*Next, Android SDK updates …*

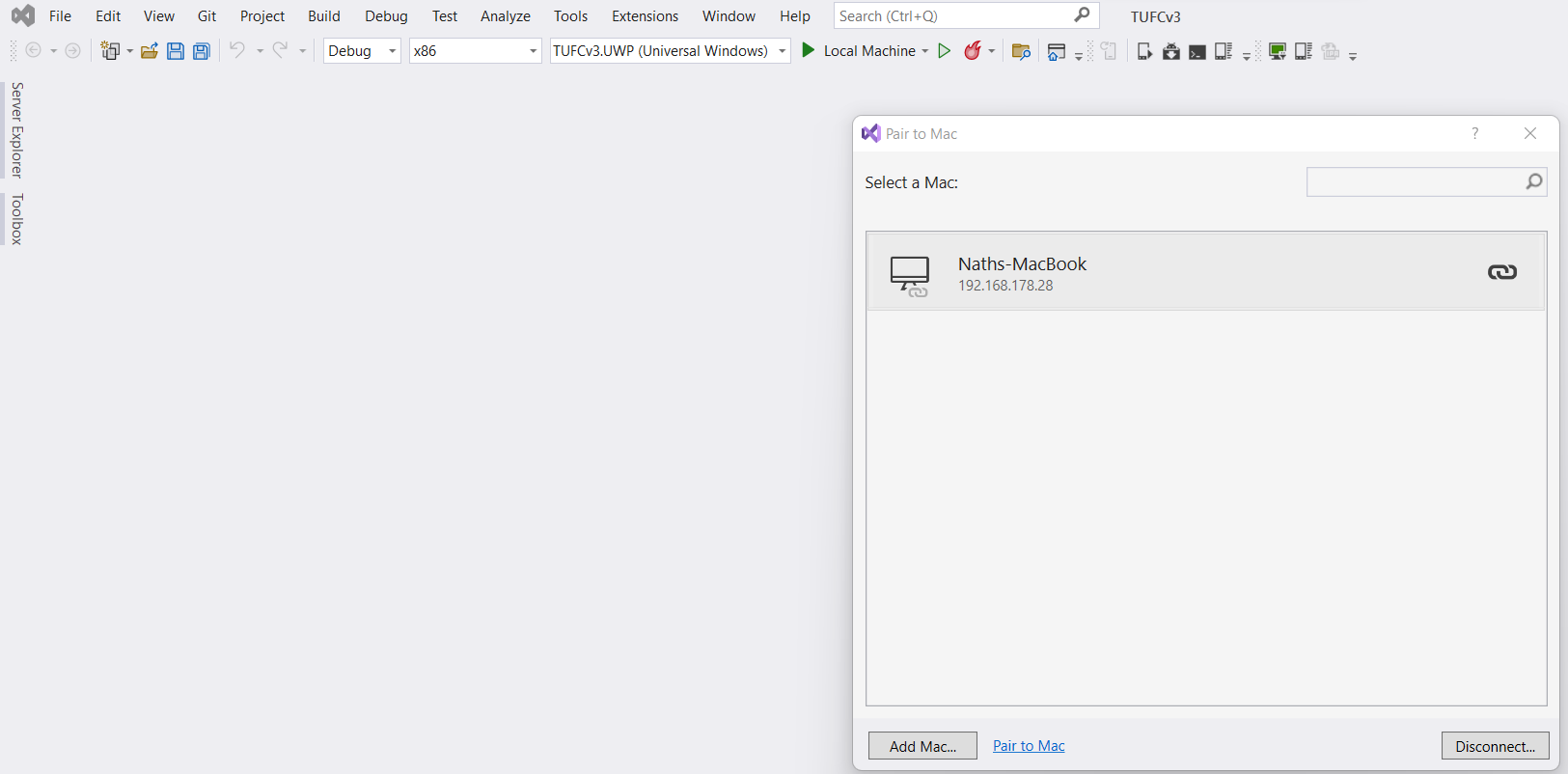
#### Android – Check the SDK manager for updates



#### Mac

When running iOS emulators on a PC, Visual Studio checks for software compatibility

while pairing with the Mac and updates Xcode versions automatically.



If the PC and Mac pair but the emulator does not start properly

it is worth trying to update the Mac’s operating system.

Although Microsoft, Android and Apple do their best to minimise incompatibility

there may also be times when one company updates software

before compatible software is released by the others.

In this case it is usually a matter of waiting 24 hours

before compatible updates become available,

after which the Android/iOS app will build

and run correctly on the emulator.

## Run the app

Now that the solution TUFCv3 has been created

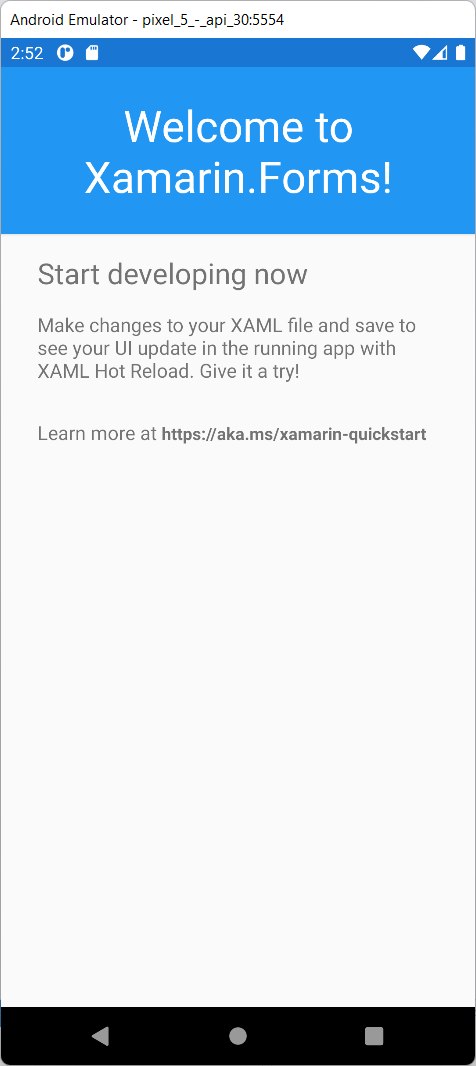
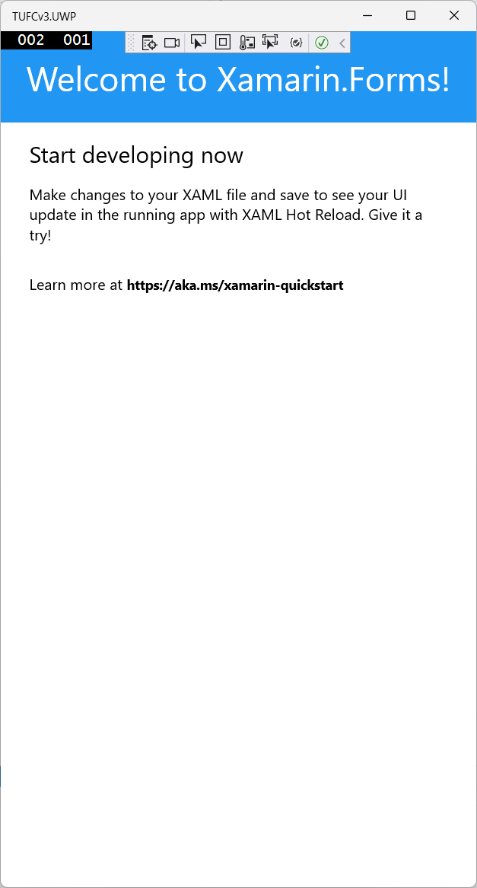
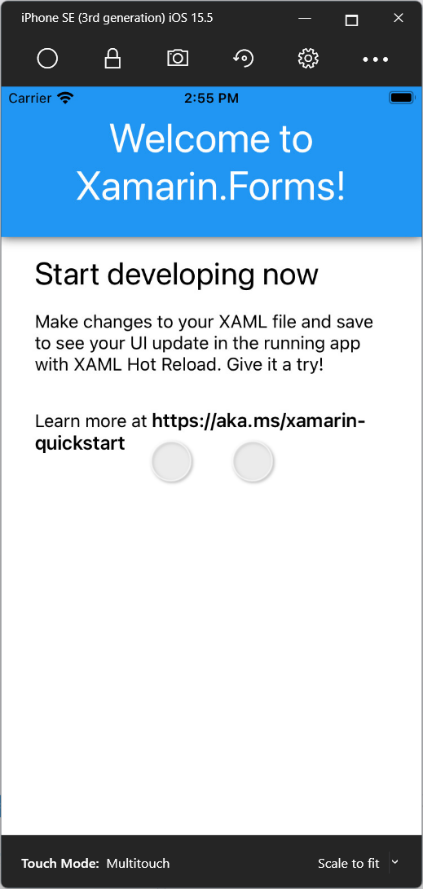
and software is up to date, run the application.

#### Test – Run the application ‘TUFCv3’

Before making any changes, run the application TUFCv3

to make sure the environment is working correctly - okay

*Android iOS UWP*



# Install MySqlConnector

https://mysqlconnector.net/

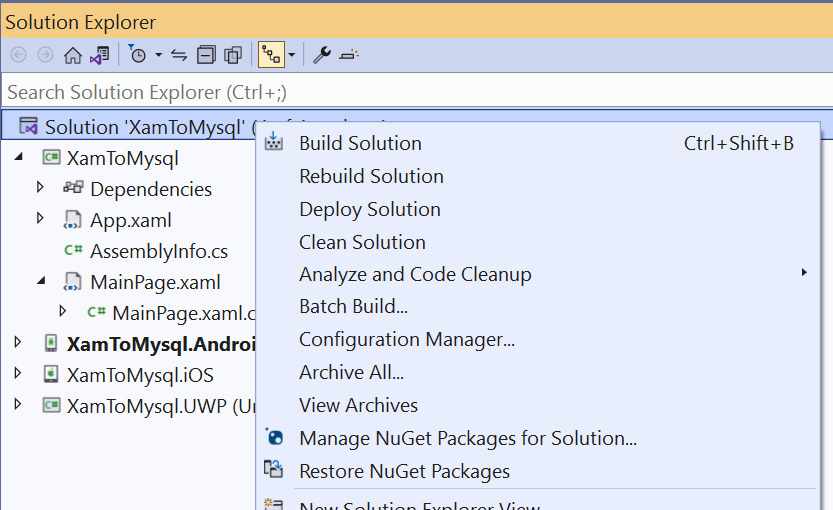
To allow the Xamarin application to connect to the MySQL database

install the NuGet package MySqlConnector

#### Example - Install MySqlConnector using NuGet

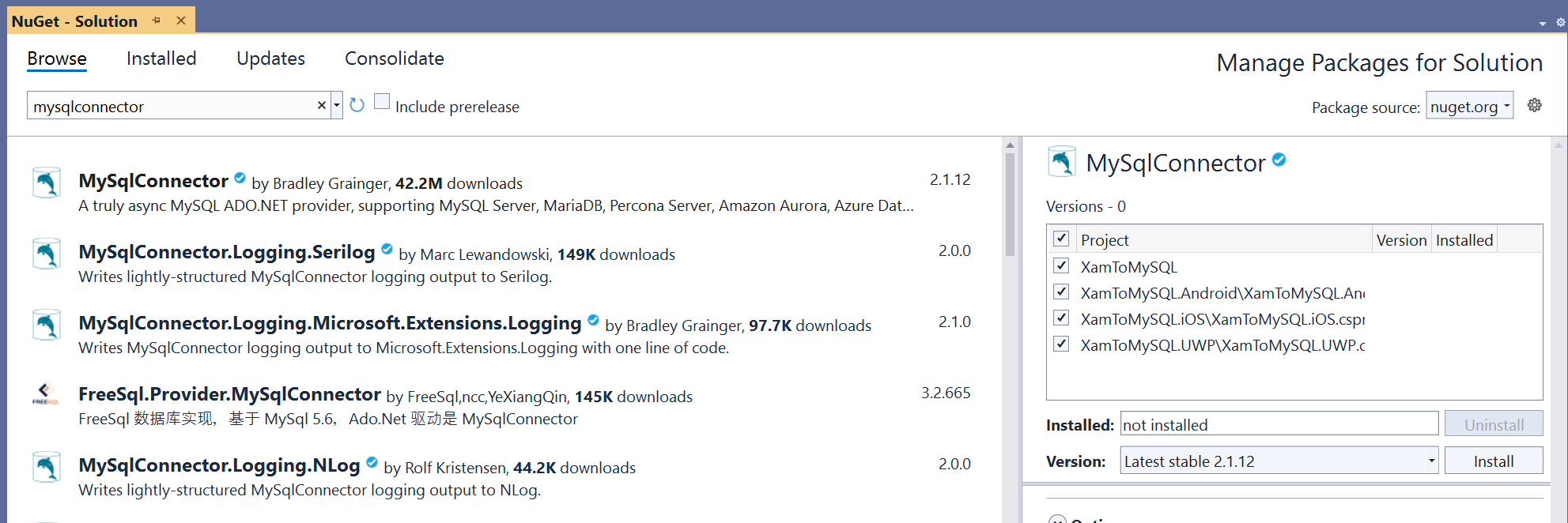
Right click the solution

and select: Manage NuGet Packages for Solution



Search for MySqlConnector

then install.



*Next, Connect to a database example using MySqlConnetor …*

#### Example - Connect to a database using MySqlConnector

In MainPage.xaml.cs

call the method ConnectToMySql()

from the constructor

using System;

using Xamarin.Forms;

namespace XamToMySQL

{

public partial class MainPage : ContentPage

{

public MainPage()

{

InitializeComponent();

ConnectToMySql();

}

public void ConnectToMySql()

{

}

}

}

In the method ConnectToMySql()

create a connection to the database

by calling the method MySqlConnection()

using the server’s connection details as arguments.

Then connect to the server.

using MySqlConnector;

using Xamarin.Forms;

namespace XamToMySQL

{

public MySqlConnection connection = new MySqlConnection();

public partial class MainPage : ContentPage

{

public MainPage()

{

InitializeComponent();

ConnectToMySql();

}

public void ConnectToMysql()

{

// Create a MySqlConnection, using the server xwm-mysql's connection details

connection = new MySqlConnection(

"Server=xwm-mysql;" +

"User ID=admin;" +

"Password=***pw***;" +

"Database=tufc");

}

}

}

In a try/catch statement open the connection

using a DisplayAlert() to indicate any connection errors.

namespace XamToMySQL

{

public partial class MainPage : ContentPage

{

public MySqlConnection connection = new MySqlConnection();

public MainPage()

{

InitializeComponent();

ConnectToMySql();

}

public void ConnectToMysql()

{

// Create a MySqlConnection, using the server xwm-mysql's connection details

connection = new MySqlConnection(

"Server=xwm-mysql;" +

"User ID=admin;" +

"Password=***pw***;" +

"Database=tufc");

try

{

connection.Open();

DisplayAlert("Connection", "Connected to the database xwm-mysql", "Okay");

connection.Close();

}

catch(Exception ex)

{

DisplayAlert("Connection", ex.Message, "Okay");

}

}

}

}

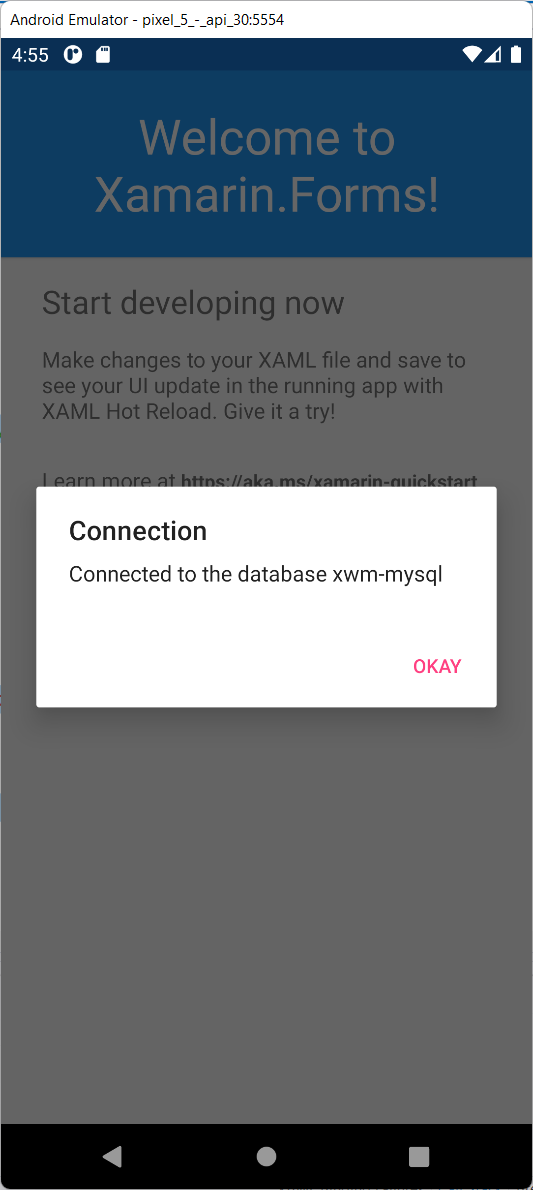
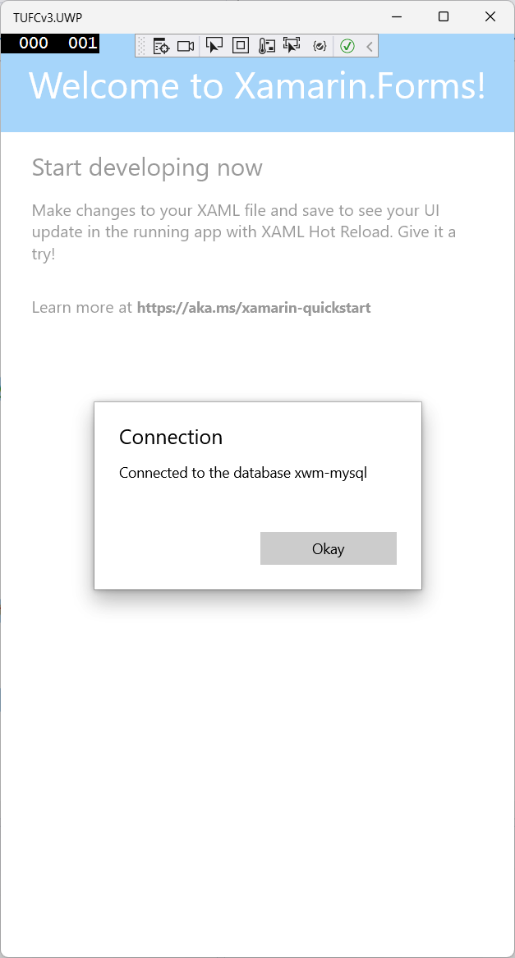
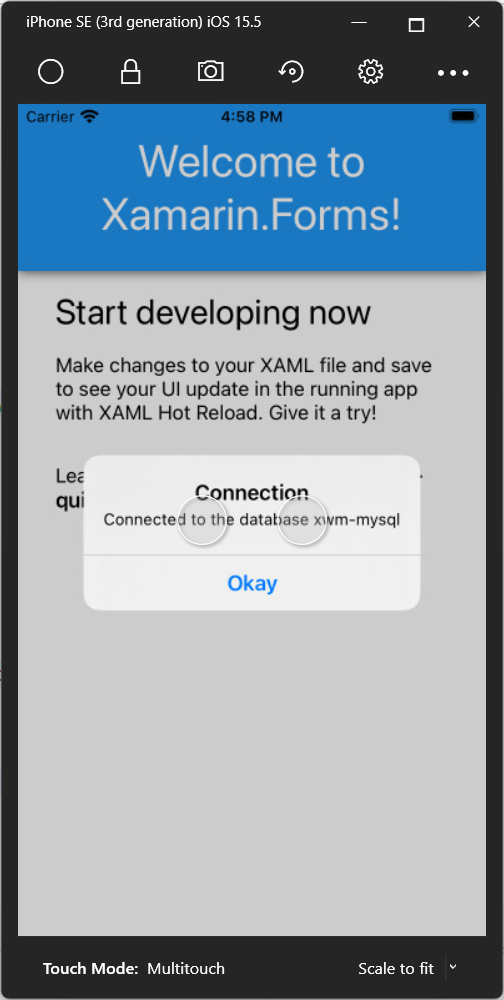
*Next, Test the connection …*

#### Test – Confirm connectivity from mobile devices to the server

Confirm connectivity from Android, iOS and UWP emulators

to the MySQL server xwm-mysql - okay

*Android iOS UWP*



*Next, Create a query …*

## Temporary layout for MainPage()

My current goal is to test interaction with the MySQL database.

This test code will be refactored into other classes

so, I don’t want to get carried away with an object model

or complicated layout, which will be modified in the near future.

Instead, I’ll create a minimal layout

that contains temporary layout controls

used while testing connectivity to the MySQL database.

### Test commands for SQL

I will test the following SQL commands *(CRUD)*

* Create INSERT
* Read SELECT
* Update UPDATE
* Delete DELETE

### Layout

Mock up the layout, before creating the XAML page.

*Title: ‘The Ultimate Fitness Companion’*

*Subtitle ’SQL command tests’*

*Label: ‘Email’ Entry: ‘Email’*

*Label: ‘New Email’ Entry: ‘newEmail’*

*Button: ‘INSERT’*

*Button: ‘UPDATE’*

*Button: ‘DELETE’*

*Button: ‘SELECT’*

*Label: ‘Users’*

*Label: (containing the list of users)*

*Next, Create the test layout for SQL commands …*

### Temporary XAML layout to test SQL commands …

#### Example – XAML, temporary layout to test SQL commands

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="TUFCv3.MainPage">

<StackLayout Padding="20">

<Label Text="The Ultimate Fitness Companion" FontSize="Large" HorizontalOptions="Center"/>

<Label Text="MySqlConnector Tests" FontSize="12" FontAttributes="Italic" HorizontalOptions="Center"/>

<Grid Padding="10">

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="0.2\*"/>

<ColumnDefinition Width="0.8\*"/>

</Grid.ColumnDefinitions>

<Label Text="Email:" Grid.Row="0" Grid.Column="0" HorizontalOptions="End" VerticalTextAlignment="Center"/>

<Entry x:Name="email" Grid.Row="0" Grid.Column="1" />

<Label Text="New email:" Grid.Row="1" Grid.Column="0" HorizontalOptions="End" VerticalTextAlignment="Center"/>

<Entry x:Name="newEmail" Grid.Row="1" Grid.Column="1"/>

</Grid>

<Button Clicked="OnInsertClick" Text="INSERT" HorizontalOptions="Center" HeightRequest="40" WidthRequest="250" Margin="5"/>

<Button Clicked="OnUpdateClick" Text="UPDATE" HorizontalOptions="Center" HeightRequest="40" WidthRequest="250" Margin="5"/>

<Button Clicked="OnDelectClick" Text="DELETE" HorizontalOptions="Center" HeightRequest="40" WidthRequest="250" Margin="5"/>

<Button Clicked="OnSelectClick" Text="SELECT" HorizontalOptions="Center" HeightRequest="40" WidthRequest="250" Margin="5" />

<Label Text="Users" FontSize="Medium" Padding="40,10,40,0"/>

<Label x:Name="lblUsers" Text="Email: test@mail.com &#x0a;Create Date: 1/2/2022" Padding="40,0" VerticalOptions="Start"/>

</StackLayout>

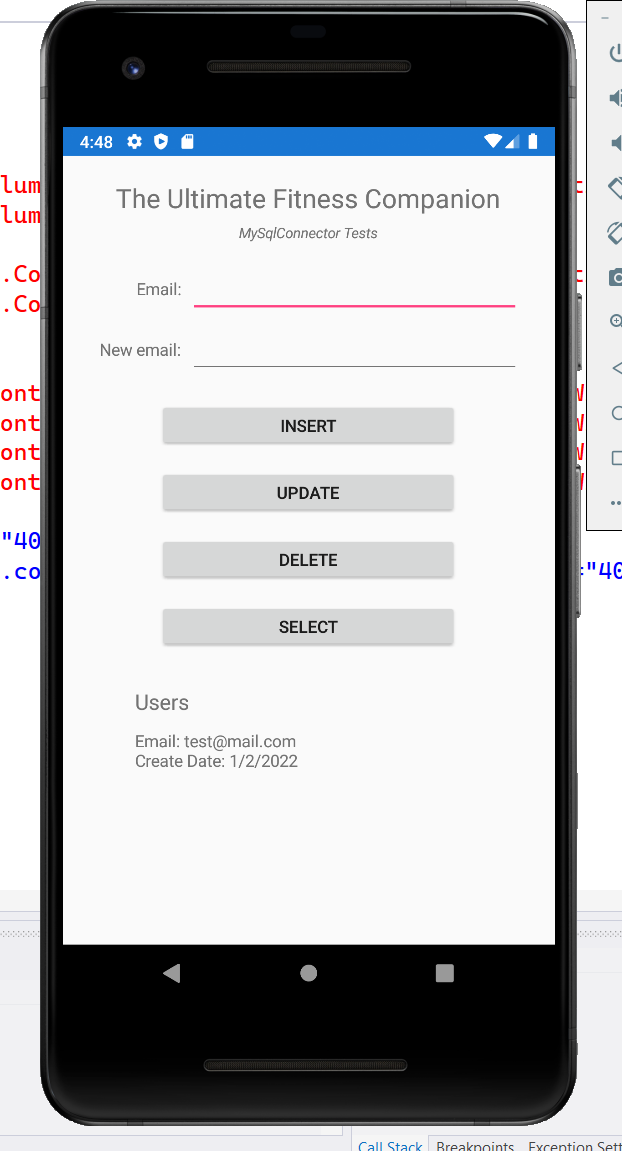
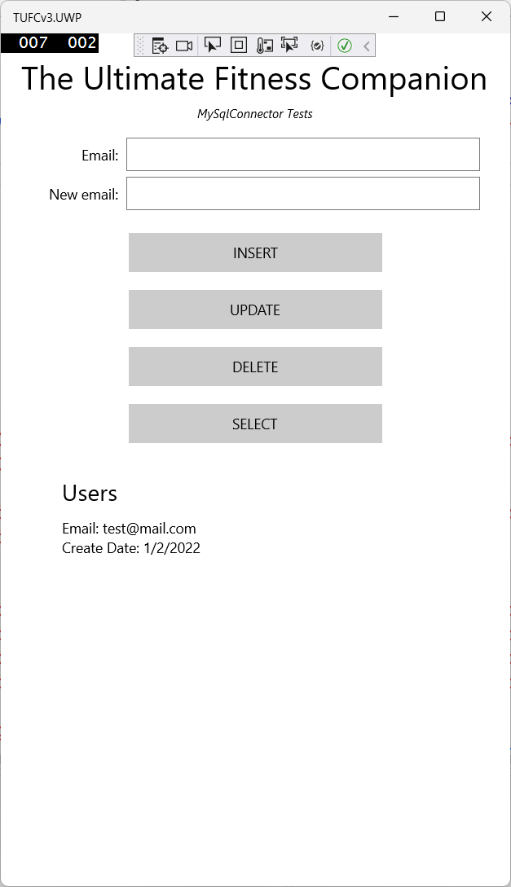
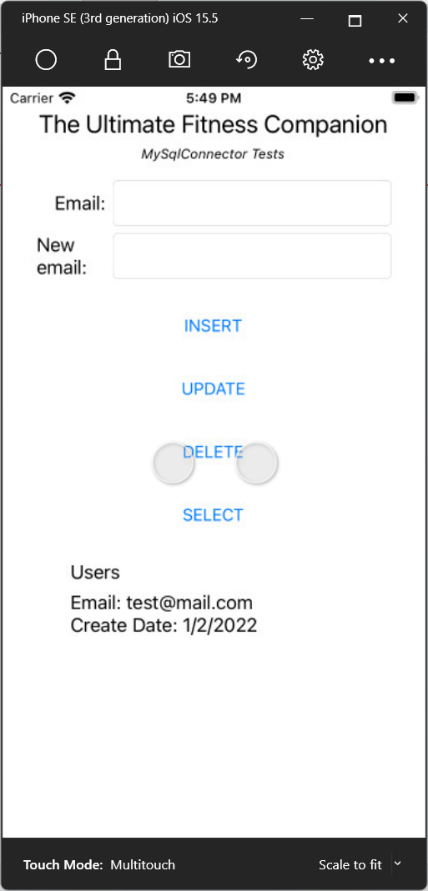
</ContentPage>

#### Test – Check temporary XAML layout

Check temporary XAML layout

to test SQL commands is working - okay

*Android iOS UWP*



## Create and execute MySqlConnector commands

I have decided to use asynchronous connections for mobile devices

while connecting to the database.

This will speed up the performance of the device, reducing hang times experienced by the user

but more importantly, it reduces the time the device is connected to the database

freeing the database for other devices queries.

*Next, MySqlConnector SELECT …*

#### Example – MySqlConnector SELECT

using System;

using Xamarin.Forms;

using MySqlConnector;

namespace TUFCv3

{

public partial class MainPage : ContentPage

{

public MySqlConnection connection = new MySqlConnection();

public MainPage()

{

InitializeComponent();

ConnectToMysql();

}

public void ConnectToMysql()

{

}

// OnSelectClick()

private async void OnSelectClick(object sender, EventArgs e)

{

// Connect to the server

try

{

connection.Open();

}

catch (Exception ex)

{

await DisplayAlert("Connection", ex.Message, "Okay");

return;

}

var query = new MySqlCommand( // Create the query

"SELECT \*" +

"FROM User",

connection);

var reader = await query.ExecuteReaderAsync();

string displayedString = "";

while (await reader.ReadAsync())

{

string email = reader.GetString(1); // User's email is already a string

string dateTime = reader.GetValue(3).ToString(); // Creation date (DateTime)

// Create the string to be displayed

displayedString +=

"\n Email: " + email +

"\n Created: " + dateTime + "\n";

}

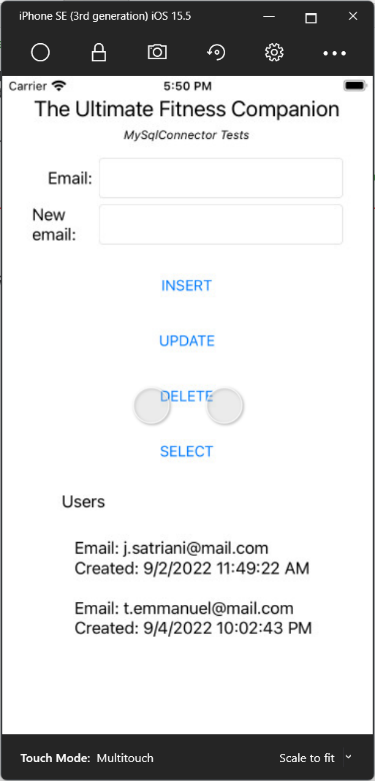
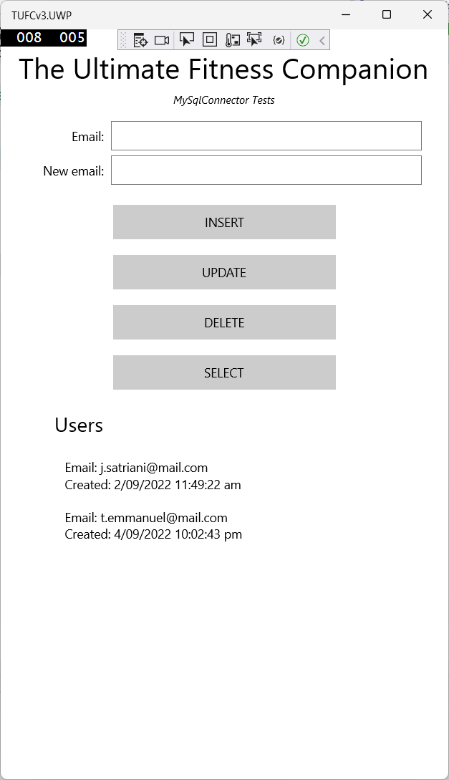
lblUsers.Text = displayedString;

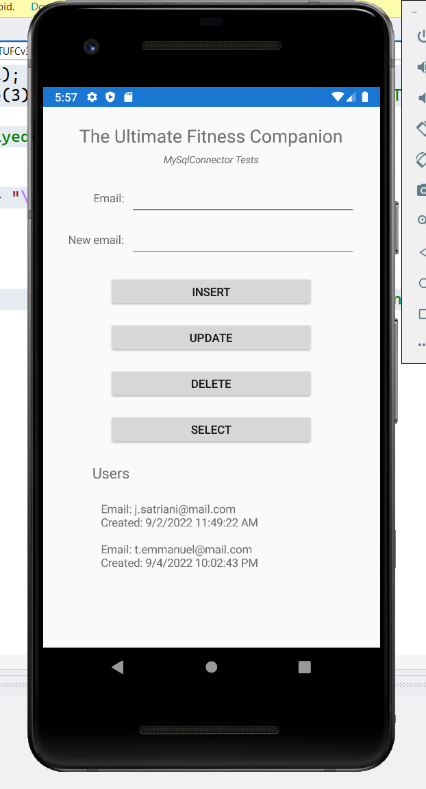
connection.Close(); // Close the connection

}

}

}

*Android iOS UWP*



Two things to note about the previous test.

* designing for layouts for small devices
* return values using MySqlConnector

### Designing for layouts for small devices

The emulators I use for testing have small screen dimensions.

They are the:

* Android ‘Pixel 2’
* Apple ‘SE3’

Designing for a small screen is easy, when you start a new project.

However, if you design layouts using emulators with larger screens

then, later discover they don’t render well on smaller devices

it can be messy to tidy up.

### Return values using MySqlConnector

The User values returned

when doing queries with MySqlConnector

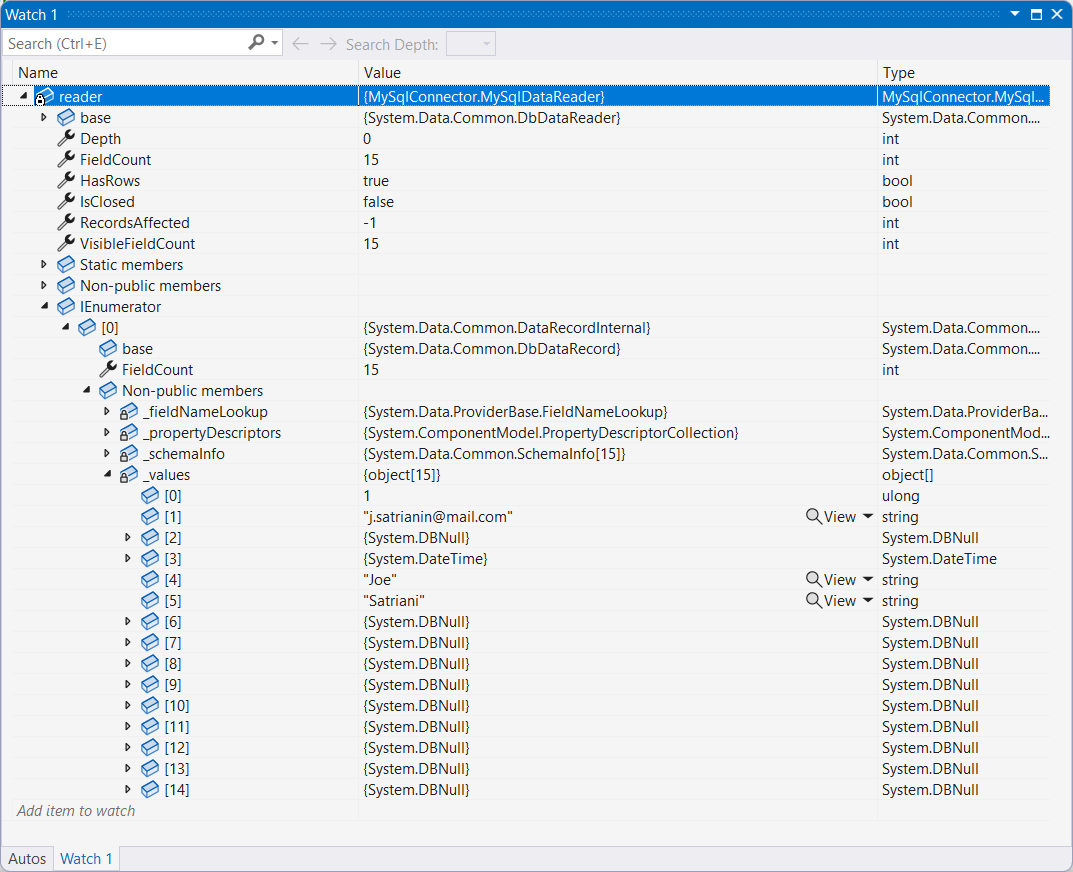
are nested arrays.

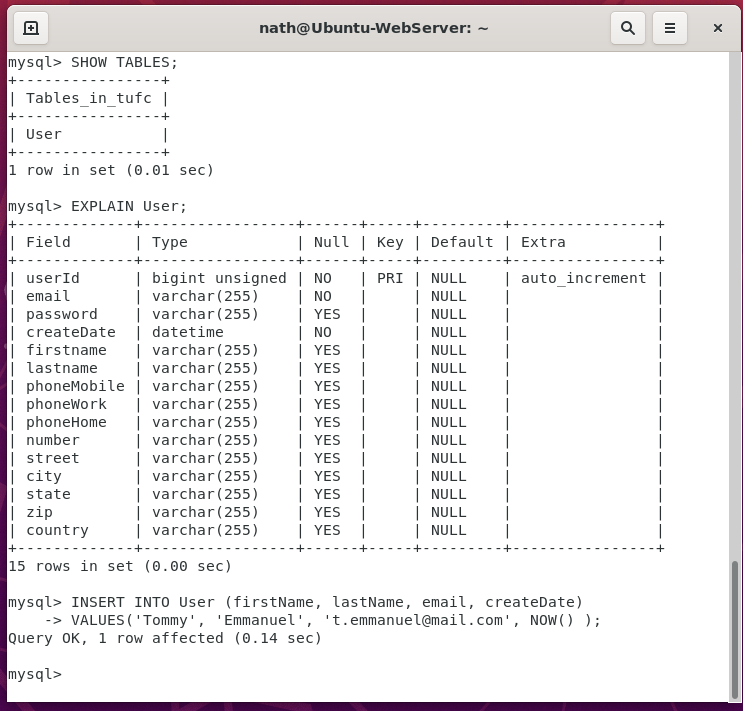
The outer array is a list of all the returned users

and the inner array is a user’s property values.

In the previous code, the MySQL User fields *(*email *and* createDate*)*

are referenced using array element numbers.

MySQL table: User Visual Studio variable: reader



#### Example – MySqlConnector INSERT

using System;

using Xamarin.Forms;

using MySqlConnector;

namespace TUFCv3

{

public partial class MainPage : ContentPage

{

public MySqlConnection connection = new MySqlConnection();

public MainPage()

{

InitializeComponent();

ConnectToMysql();

}

public void ConnectToMysql()

{

}

private async void OnInsertClick(object sender, EventArgs e)

{

// Connect to the server

try

{

connection.Open();

}

catch (Exception ex)

{

await DisplayAlert("Connection", ex.Message, "Okay");

return;

}

// Create an execute an INSERT command

using (var cmd = new MySqlCommand())

{

cmd.Connection = connection;

cmd.CommandText = "INSERT INTO User (email, createDate) VALUES (@e, @d)"; // Create a query to save data to MySQL

cmd.Parameters.AddWithValue("e", email.Text); // String from the xaml entry 'email'

cmd.Parameters.AddWithValue("d", DateTime.Now); // Current time, obtained using the function DateTime()

// Save the data

try

{

cmd.ExecuteNonQuery();

await DisplayAlert("Insert data", email.Text + " inserted \ninto the table 'User'", "Okay");

}

catch (Exception ex)

{

await DisplayAlert("Insert data", ex.Message, "Okay");

}

}

connection.Close();

}

private async void OnUpdateClick(object sender, EventArgs e)

{

}

private async void OnDeleteClick(object sender, EventArgs e)

{

}

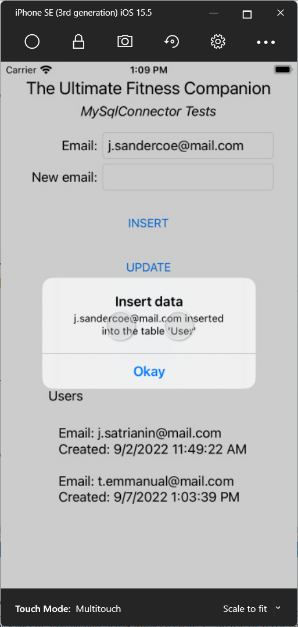
private async void OnSelectClick(object sender, EventArgs e)

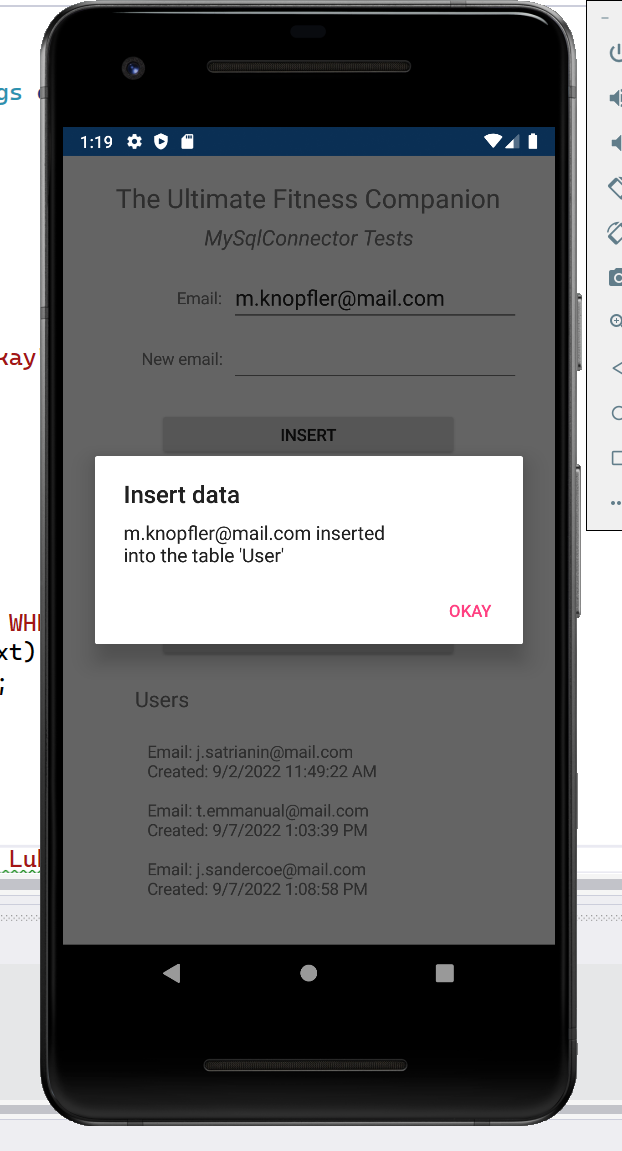
{

}

}

}

*Android iOS*



*Next, MySqlConnector command syntax …*

Note the syntax used when inserting values with MySqlConnector

// Create an execute an INSERT command

using (var cmd = new MySqlCommand())

{

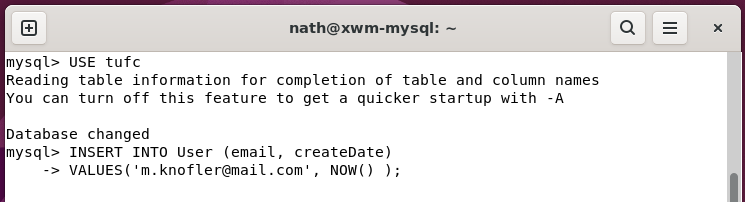
cmd.Connection = connection;

cmd.CommandText = "INSERT INTO User (email, createDate) VALUES (@e, @d)"; // Create a query to save data to MySQL

cmd.Parameters.AddWithValue("e", email.Text); // String from the xaml entry 'email'

cmd.Parameters.AddWithValue("d", DateTime.Now); // Current time, obtained using the function DateTime()

The way the values are entered not the same as a SQL command



Instead, each VALUE is assigned a parameter *(for example* @e*)*

Which is combined into the SQL command

using the method cmd.Parameters.AddWithValue(*parameter, value*)

This must be followed when using values in MySqlConnector

and ensures the correct types are inserted into the MySQL database.

#### Example – MySqlConnector UPDATE

using System;

using Xamarin.Forms;

using MySqlConnector;

namespace TUFCv3

{

public partial class MainPage : ContentPage

{

public MySqlConnection connection = new MySqlConnection();

public MainPage()

{

InitializeComponent();

ConnectToMysql();

}

public void ConnectToMysql()

{

}

private async void OnInsertClick(object sender, EventArgs e)

{

}

private async void OnUpdateClick(object sender, EventArgs e)

{

// Connect to the server

try

{

connection.Open();

}

catch (Exception ex)

{

await DisplayAlert("Connection", ex.Message, "Okay");

return;

}

using (var cmd = new MySqlCommand())

{

cmd.Connection = connection;

// Create the command UPDATE

cmd.CommandText = "UPDATE User " +

"SET email = @newEmail " +

"WHERE email = @oldEmail ";

cmd.Parameters.AddWithValue("@newEmail", newEmail.Text);

cmd.Parameters.AddWithValue("@oldEmail", email.Text);

// Execute the query

try

{

cmd.ExecuteReader();

await DisplayAlert("Connection", email.Text + " updated to " + newEmail.Text + "\n on the database 'xwm-mysql' ", "Okay");

}

catch (Exception ex)

{

await DisplayAlert("Connection", ex.Message, "Okay");

}

}

connection.Close();

}

private async void OnDeleteClick(object sender, EventArgs e)

{

}

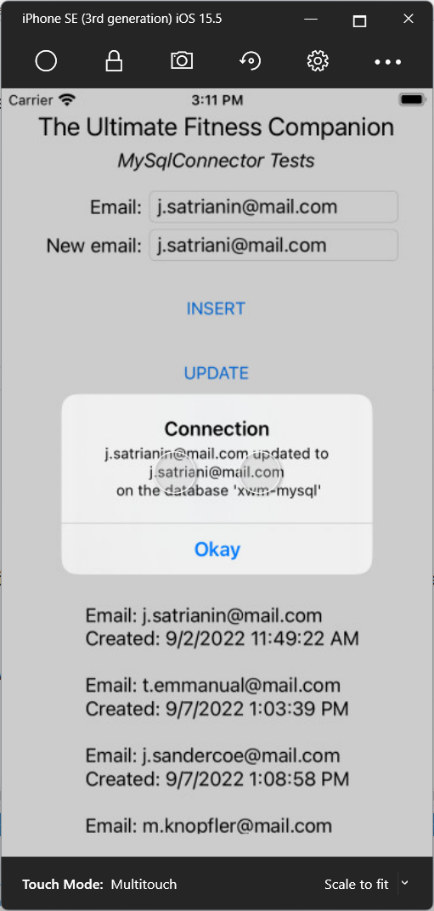
private async void OnSelectClick(object sender, EventArgs e)

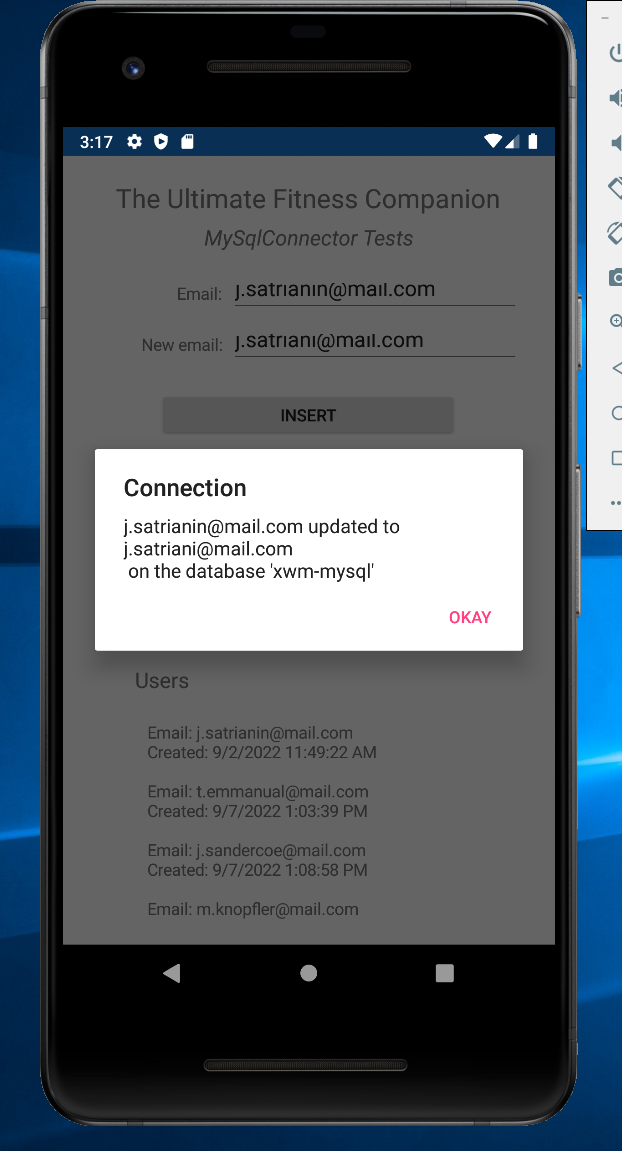
{

}

}

}

*Android iOS*



##### Example – MySqlConnector DELETE

using System;

using Xamarin.Forms;

using MySqlConnector;

namespace TUFCv3

{

public partial class MainPage : ContentPage

{

public MySqlConnection connection = new MySqlConnection();

public MainPage()

{

InitializeComponent();

ConnectToMysql();

}

public void ConnectToMysql()

{

}

private async void OnInsertClick(object sender, EventArgs e)

{

}

private async void OnUpdateClick(object sender, EventArgs e)

{

}

private async void OnDeleteClick(object sender, EventArgs e)

{

// Connect to the server

try

{

connection.Open();

}

catch (Exception ex)

{

await DisplayAlert("Connection", ex.Message, "Okay");

return;

}

using (var cmd = new MySqlCommand())

{

// Create the command to DELETE

cmd.Connection = connection;

cmd.CommandText = "DELETE FROM User " +

"WHERE email = @e";

cmd.Parameters.AddWithValue("@e", email.Text);

// Execute the query

try

{

cmd.ExecuteReader();

await DisplayAlert("Connection", "Deleted " + email.Text + "\nfrom 'Ubuntu-MySQL' ", "Okay");

}

catch (Exception ex)

{

await DisplayAlert("Connection", ex.Message, "Okay");

}

connection.Close();

}

}

private async void OnSelectClick(object sender, EventArgs e)

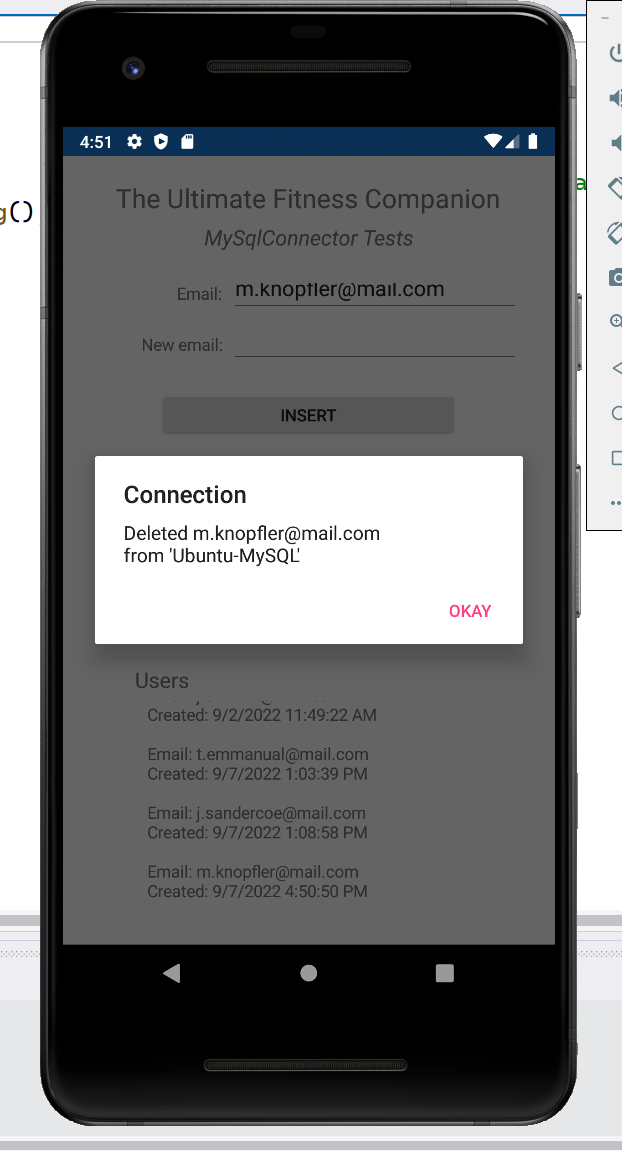
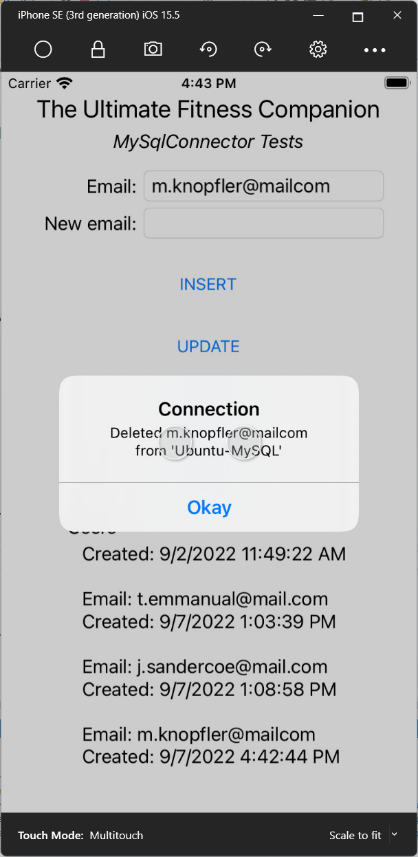
{

}

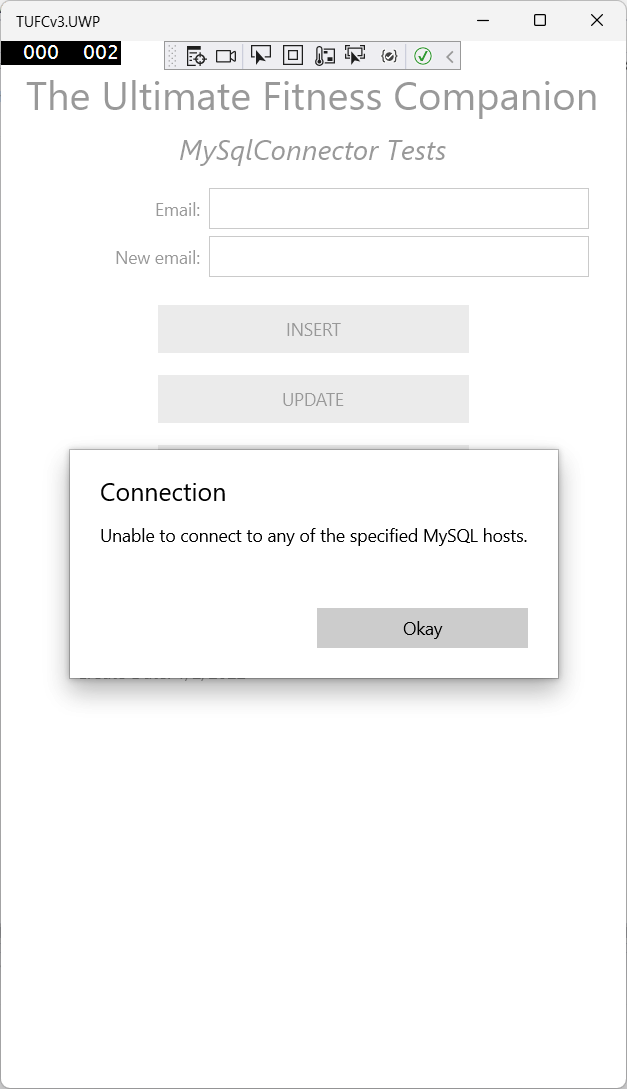
}

}

*Android iOS*



### Troubleshooting ‘Unable to connect to any of the specified MySQL Hosts’



# MVVM: *‘Model, View, ViewModel’*

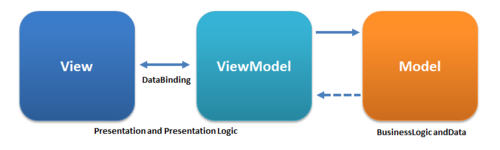
As the next step in I need in the Xamarin application TUFCv3

is to create the model User

it’s a good time to take a look at the design pattern MVVM

MVVM separates the development of the user interface (the ***View***)

from the business logic or back-end logic (the ***Model***)



*https://commons.wikimedia.org/wiki/File:MVVMPattern.png#/media/File:MVVMPattern.png*

The ***ViewModel*** exposes object models

in a way that it’s data can be easily managed and presented.

allowing the graphical user interface, and it’s back-end,

to be developed independently.