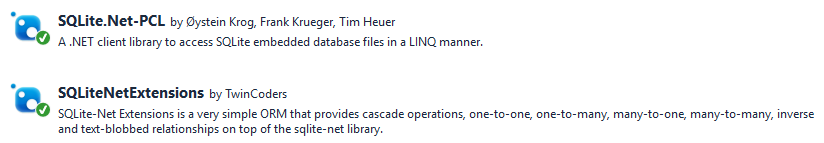
Demo SQLite Relations

1. Add nugets: SQLite.Net-PCL and SQLiteNetExtentions



1. In the folder Classes, add the classes Product and Service:

public class Product

{

[PrimaryKey, AutoIncrement]

public int ProductId { get; set; }

[Unique]

public string Description { get; set; }

public decimal Price { get; set; }

[OneToMany(CascadeOperations = CascadeOperation.All)]

public List<Service> Services { get; set; }

public override int GetHashCode()

{

return ProductId;

}

public override string ToString()

{

return string.Format("{0} {1} {2:C2}", ProductId, Description, Price);

}

}

public class Service

{

[PrimaryKey, AutoIncrement]

public int ServiceId { get; set; }

public DateTime DateService { get; set; }

public DateTime DateRegistered { get; set; }

public int ProductId { get; set; }

[ManyToOne]

public Product Product { get; set; }

public decimal Price { get; set; }

public double Quantity { get; set; }

public decimal Value { get { return Price \* (decimal)Quantity; } }

public override int GetHashCode()

{

return ServiceId;

}

public override string ToString()

{

return string.Format("{0} {1:d} {2:C2}", ServiceId, DateService, Value);

}

}

1. In the folder Interfaces, add the interfaz IConfig

public interface IConfig

{

string DirectoryDB { get; }

ISQLitePlatform Platform { get; }

}

1. Add the implementation for every specific project:

// For Android

using Services.Interfaces;

using SQLite.Net.Interop;

using Xamarin.Forms;

[assembly: Dependency(typeof(Services.Droid.Config))]

namespace Services.Droid

{

public class Config : IConfig

{

private string directoryDB;

private ISQLitePlatform platform;

public string DirectoryDB

{

get

{

if (string.IsNullOrEmpty(directoryDB))

{

directoryDB = System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal);

}

return directoryDB;

}

}

public ISQLitePlatform Platform

{

get

{

if (platform == null)

{

platform = new SQLite.Net.Platform.XamarinAndroid.SQLitePlatformAndroid();

}

return platform;

}

}

}

}

// For iOS

using Services.Interfaces;

using SQLite.Net.Interop;

using System;

using Xamarin.Forms;

[assembly: Dependency(typeof(Services.iOS.Config))]

namespace Services.iOS

{

public class Config : IConfig

{

private string directoryDB;

private ISQLitePlatform platform;

public string DirectoryDB

{

get

{

if (string.IsNullOrEmpty(directoryDB))

{

var directory = System.Environment.GetFolderPath(Environment.SpecialFolder.Personal);

directoryDB = System.IO.Path.Combine(directory, "..", "Library");

}

return directoryDB;

}

}

public ISQLitePlatform Platform

{

get

{

if (platform == null)

{

platform = new SQLite.Net.Platform.XamarinIOS.SQLitePlatformIOS();

}

return platform;

}

}

}

}

1. In the folder Classes, add the DataAccess class:

public class DataAccess : IDisposable

{

private SQLiteConnection connection;

public DataAccess()

{

var config = DependencyService.Get<IConfig>();

connection = new SQLiteConnection(config.Platform,

System.IO.Path.Combine(config.DirectoryDB, "Services.db3"));

connection.CreateTable<Product>();

connection.CreateTable<Service>();

}

public void Insert<T>(T model)

{

connection.Insert(model);

}

public void Update<T>(T model)

{

connection.Update(model);

}

public void Delete<T>(T model)

{

connection.Delete(model);

}

public T First<T>(bool WithChildren) where T : class

{

if (WithChildren)

{

return connection.GetAllWithChildren<T>().FirstOrDefault();

}

else

{

return connection.Table<T>().FirstOrDefault();

}

}

public List<T> GetList<T>(bool WithChildren) where T : class

{

if (WithChildren)

{

return connection.GetAllWithChildren<T>().ToList();

}

else

{

return connection.Table<T>().ToList();

}

}

public T Find<T>(int pk, bool WithChildren) where T : class

{

if (WithChildren)

{

return connection.GetAllWithChildren<T>().FirstOrDefault(m => m.GetHashCode() == pk);

}

else

{

return connection.Table<T>().FirstOrDefault(m => m.GetHashCode() == pk);

}

}

public void Dispose()

{

connection.Dispose();

}

}

1. In the folder Pages Add the HomePage:

<?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="Services.Pages.HomePage"

Title="Services">

<ScrollView>

<StackLayout Spacing="10" Padding="15" VerticalOptions="Center">

<Button x:Name="productsButton"

Text="Products"

BackgroundColor="Yellow"

TextColor="Black"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<Button x:Name="serviesButton"

Text="Services"

BackgroundColor="Aqua"

TextColor="Black"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<Button x:Name="queriesButton"

Text="Queries"

BackgroundColor="Purple"

TextColor="White"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

</StackLayout>

</ScrollView>

</ContentPage>

public partial class HomePage : ContentPage

{

public HomePage()

{

InitializeComponent();

productsButton.Clicked += ProductsButton\_Clicked;

serviesButton.Clicked += ServiesButton\_Clicked;

queriesButton.Clicked += QueriesButton\_Clicked;

}

private async void QueriesButton\_Clicked(object sender, EventArgs e)

{

await Navigation.PushAsync(new QueriesPage());

}

private async void ServiesButton\_Clicked(object sender, EventArgs e)

{

await Navigation.PushAsync(new ServicesPage());

}

private async void ProductsButton\_Clicked(object sender, EventArgs e)

{

await Navigation.PushAsync(new ProductsPage());

}

}

1. Modify the App class:

public App()

{

MainPage = new NavigationPage(new HomePage());

}

1. In the folder Pages, add the ProductPage:

<?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="Services.Pages.ProductsPage"

Title="Products">

<ScrollView>

<StackLayout Spacing="10" Padding="15">

<Entry x:Name="descriptionEntry"

Placeholder="Enter description"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<Entry x:Name="priceEntry"

Placeholder="Enter price"

Keyboard="Numeric"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<Button x:Name="addButton"

Text="Add"

BackgroundColor="Yellow"

TextColor="Black"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<ListView x:Name="productsListView"/>

</StackLayout>

</ScrollView>

</ContentPage>

public partial class ProductsPage : ContentPage

{

public ProductsPage()

{

InitializeComponent();

productsListView.ItemTemplate = new DataTemplate(typeof(ProductCell));

productsListView.ItemSelected += ProductsListView\_ItemSelected;

addButton.Clicked += AddButton\_Clicked;

}

private async void ProductsListView\_ItemSelected(object sender, SelectedItemChangedEventArgs e)

{

await Navigation.PushAsync(new EditProductsPage((Product)e.SelectedItem));

}

protected override void OnAppearing()

{

base.OnAppearing();

using (var da = new DataAccess())

{

productsListView.ItemsSource = da.GetList<Product>(false).OrderBy(p => p.Description);

}

}

private async void AddButton\_Clicked(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(descriptionEntry.Text))

{

await DisplayAlert("Error", "You must enter a description", "Acept");

return;

}

if (string.IsNullOrEmpty(priceEntry.Text))

{

await DisplayAlert("Error", "You must enter a price", "Acept");

return;

}

var price = decimal.Parse(priceEntry.Text);

if (price < 0)

{

await DisplayAlert("Error", "The price must be a value greather or equals to zero", "Acept");

return;

}

var product = new Product

{

Description = descriptionEntry.Text,

Price = price,

};

using (var da = new DataAccess())

{

da.Insert(product);

productsListView.ItemsSource = da.GetList<Product>(false).OrderBy(p => p.Description);

}

descriptionEntry.Text = string.Empty;

priceEntry.Text = string.Empty;

await DisplayAlert("Message", "The product was added successfully", "Acept");

}

}

1. In the folder Cells, add the ProductCell:

public class ProductCell : ViewCell

{

public ProductCell()

{

var descriptionLabel = new Label

{

HorizontalOptions = LayoutOptions.StartAndExpand,

VerticalOptions = LayoutOptions.Center,

FontAttributes = FontAttributes.Bold,

};

descriptionLabel.SetBinding(Label.TextProperty, new Binding("Description"));

var priceLabel = new Label

{

HorizontalTextAlignment = TextAlignment.End,

HorizontalOptions = LayoutOptions.End,

VerticalOptions = LayoutOptions.Center,

};

priceLabel.SetBinding(Label.TextProperty, new Binding("Price", stringFormat: "{0:C2}"));

View = new StackLayout

{

Orientation = StackOrientation.Horizontal,

Children = {

descriptionLabel, priceLabel,

},

};

}

}

1. In the folder Pages, add the EditProductsPage:

<?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="Services.Pages.EditProductsPage"

Title="Edit Product">

<ScrollView>

<StackLayout Spacing="10" Padding="15">

<Entry x:Name="descriptionEntry"

Placeholder="Enter description"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<Entry x:Name="priceEntry"

Placeholder="Enter price"

Keyboard="Numeric"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<StackLayout Orientation="Horizontal">

<Button x:Name="updateButton"

Text="Update"

BackgroundColor="Yellow"

TextColor="Black"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<Button x:Name="deleteButton"

Text="Delete"

BackgroundColor="Red"

TextColor="White"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

</StackLayout>

</StackLayout>

</ScrollView>

</ContentPage>

public partial class EditProductsPage : ContentPage

{

private Product product;

public EditProductsPage(Product product)

{

InitializeComponent();

this.product = product;

descriptionEntry.Text = product.Description;

priceEntry.Text = product.Price.ToString();

updateButton.Clicked += UpdateButton\_Clicked;

deleteButton.Clicked += DeleteButton\_Clicked;

}

private async void DeleteButton\_Clicked(object sender, EventArgs e)

{

var rta = await DisplayAlert("Confirm", "Are you sure to delete the record?", "Yes", "No");

if (!rta)

{

return;

}

using (var da = new DataAccess())

{

var service = da.GetList<Service>(false).Where(s => s.ProductId == product.ProductId).FirstOrDefault();

if (service != null)

{

await DisplayAlert("Message", "The record can't be deleted because it has related records", "Acept");

return;

}

da.Delete(product);

}

await DisplayAlert("Message", "The record was deleted", "Acept");

await Navigation.PopAsync();

}

private async void UpdateButton\_Clicked(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(descriptionEntry.Text))

{

await DisplayAlert("Error", "You must enter a description", "Acept");

return;

}

if (string.IsNullOrEmpty(priceEntry.Text))

{

await DisplayAlert("Error", "You must enter a price", "Acept");

return;

}

var price = decimal.Parse(priceEntry.Text);

if (price < 0)

{

await DisplayAlert("Error", "The price must be a value greather or equals to zero", "Acept");

return;

}

product.Description = descriptionEntry.Text;

product.Price = price;

using (var da = new DataAccess())

{

da.Update(product);

}

await DisplayAlert("Message", "The record was updated", "Acept");

await Navigation.PopAsync();

}

}

1. In the folder Pages, add the ServicePage:

<?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="Services.Pages.ServicesPage"

Title="Services">

<ScrollView>

<StackLayout Spacing="10" Padding="15">

<Picker x:Name="productPicker"

Title="Select a product"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<StackLayout Orientation="Horizontal">

<Label Text="Quantity: "

HorizontalOptions="Start"

VerticalOptions="Center"/>

<Entry x:Name="quantityEntry"

IsEnabled="False"

Text="1"

FontAttributes="Bold"

FontSize="20"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<Stepper x:Name="quantityStepper"

Value="1"

Increment="1"

Minimum="1"

Maximum="999999"

HorizontalOptions="End"

VerticalOptions="Center"/>

</StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="Date: "

HorizontalOptions="Start"

VerticalOptions="Center"/>

<DatePicker x:Name="dateDatePicker"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

</StackLayout>

<Button x:Name="addButton"

Text="Add"

BackgroundColor="Yellow"

TextColor="Black"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<ListView x:Name="servicesListView"/>

</StackLayout>

</ScrollView>

</ContentPage>

public partial class ServicesPage : ContentPage

{

private List<Product> products;

public ServicesPage()

{

InitializeComponent();

LoadProducts();

servicesListView.ItemTemplate = new DataTemplate(typeof(ServiceCell));

servicesListView.RowHeight = 70;

quantityStepper.ValueChanged += QuantityStepper\_ValueChanged;

addButton.Clicked += AddButton\_Clicked;

servicesListView.ItemSelected += ServicesListView\_ItemSelected;

}

private async void ServicesListView\_ItemSelected(object sender, SelectedItemChangedEventArgs e)

{

await Navigation.PushAsync(new EditService((Service)e.SelectedItem));

}

private async void AddButton\_Clicked(object sender, EventArgs e)

{

if (productPicker.SelectedIndex == -1)

{

await DisplayAlert("Error", "You must select a product", "Accept");

return;

}

var service = new Service

{

DateService = dateDatePicker.Date,

DateRegistered = DateTime.Today,

Price = products[productPicker.SelectedIndex].Price,

ProductId = products[productPicker.SelectedIndex].ProductId,

Quantity = quantityStepper.Value,

};

using (var da = new DataAccess())

{

da.Insert(service);

var services = da.GetList<Service>(true)

.Where(s => s.DateRegistered.Year == DateTime.Today.Year &&

s.DateRegistered.Month == DateTime.Today.Month &&

s.DateRegistered.Day == DateTime.Today.Day)

.OrderByDescending(s => s.DateService)

.ToList();

servicesListView.ItemsSource = services;

}

productPicker.SelectedIndex = -1;

dateDatePicker.Date = DateTime.Now;

quantityEntry.Text = "1";

quantityStepper.Value = 1;

await DisplayAlert("Message", "The record was added", "Accept");

}

private void LoadProducts()

{

using (var da = new DataAccess())

{

products = da.GetList<Product>(false).OrderBy(p => p.Description).ToList();

}

foreach (var product in products)

{

productPicker.Items.Add(product.Description);

}

}

protected override void OnAppearing()

{

base.OnAppearing();

dateDatePicker.Date = DateTime.Now;

using (var da = new DataAccess())

{

var services = da.GetList<Service>(true)

.Where(s => s.DateRegistered.Year == DateTime.Today.Year &&

s.DateRegistered.Month == DateTime.Today.Month &&

s.DateRegistered.Day == DateTime.Today.Day)

.OrderByDescending(s => s.DateService)

.ToList();

servicesListView.ItemsSource = services;

}

}

private void QuantityStepper\_ValueChanged(object sender, ValueChangedEventArgs e)

{

quantityEntry.Text = quantityStepper.Value.ToString();

}

}

1. In the folder Cells, add the ServiceCell:

public class ServiceCell : ViewCell

{

public ServiceCell()

{

var dateLabel = new Label

{

HorizontalTextAlignment = TextAlignment.Start,

HorizontalOptions = LayoutOptions.StartAndExpand,

FontAttributes = FontAttributes.Bold,

FontSize = 20,

};

dateLabel.SetBinding(Label.TextProperty, new Binding("DateService", stringFormat: "{0:yyyy/MM/dd}"));

var descriptionLabel = new Label

{

HorizontalTextAlignment = TextAlignment.End,

HorizontalOptions = LayoutOptions.EndAndExpand,

FontAttributes = FontAttributes.Bold,

FontSize = 20,

};

descriptionLabel.SetBinding(Label.TextProperty, new Binding("Product.Description"));

var quantityLabel = new Label

{

HorizontalOptions = LayoutOptions.StartAndExpand,

};

quantityLabel.SetBinding(Label.TextProperty, new Binding("Quantity", stringFormat: "Quantity: {0:N2}"));

var valueLabel = new Label

{

HorizontalOptions = LayoutOptions.EndAndExpand,

};

valueLabel.SetBinding(Label.TextProperty, new Binding("Value", stringFormat: "Value: {0:N2}"));

var line1 = new StackLayout

{

Orientation = StackOrientation.Horizontal,

Children = {

dateLabel, descriptionLabel,

},

};

var line2 = new StackLayout

{

Orientation = StackOrientation.Horizontal,

Children = {

quantityLabel, valueLabel,

},

};

View = new StackLayout

{

Orientation = StackOrientation.Vertical,

Children = {

line1, line2,

},

};

}

}

1. In the folder Pages, add the EditServicePage:

<?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="Services.Pages.EditService"

Title="Edit Service">

<ScrollView>

<StackLayout Spacing="10" Padding="15">

<Picker x:Name="productPicker"

Title="Select a product"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<StackLayout Orientation="Horizontal">

<Label Text="Quantity: "

HorizontalOptions="Start"

VerticalOptions="Center"/>

<Entry x:Name="quantityEntry"

IsEnabled="False"

Text="1"

FontAttributes="Bold"

FontSize="20"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<Stepper x:Name="quantityStepper"

Value="1"

Increment="1"

Minimum="1"

Maximum="999999"

HorizontalOptions="End"

VerticalOptions="Center"/>

</StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="Date: "

HorizontalOptions="Start"

VerticalOptions="Center"/>

<DatePicker x:Name="dateDatePicker"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

</StackLayout>

<StackLayout Orientation="Horizontal">

<Button x:Name="updateButton"

Text="Update"

BackgroundColor="Yellow"

TextColor="Black"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

<Button x:Name="deleteButton"

Text="Delete"

BackgroundColor="Red"

TextColor="White"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

</StackLayout>

</StackLayout>

</ScrollView>

</ContentPage>

public partial class EditService : ContentPage

{

private List<Product> products;

private Service service;

public EditService(Service service)

{

InitializeComponent();

this.service = service;

LoadProducts();

LoadForm();

quantityStepper.ValueChanged += QuantityStepper\_ValueChanged;

updateButton.Clicked += UpdateButton\_Clicked;

deleteButton.Clicked += DeleteButton\_Clicked;

}

private async void DeleteButton\_Clicked(object sender, EventArgs e)

{

var rta = await DisplayAlert("Confirm", "Are you sure to delete the record?", "Yes", "No");

if (!rta)

{

return;

}

using (var da = new DataAccess())

{

da.Delete(service);

}

await DisplayAlert("Message", "The record was deleted", "Acept");

await Navigation.PopAsync();

}

private async void UpdateButton\_Clicked(object sender, EventArgs e)

{

if (productPicker.SelectedIndex == -1)

{

await DisplayAlert("Error", "You must select a product", "Accept");

return;

}

service.DateService = dateDatePicker.Date;

service.Price = products[productPicker.SelectedIndex].Price;

service.ProductId = products[productPicker.SelectedIndex].ProductId;

service.Quantity = quantityStepper.Value;

using (var da = new DataAccess())

{

da.Update(service);

}

await DisplayAlert("Message", "The record was updated", "Acept");

await Navigation.PopAsync();

}

private void LoadForm()

{

int i = 0;

for (; i < products.Count; i++)

{

if (products[i].ProductId == service.ProductId)

{

break;

}

}

if (i == products.Count)

{

i = -1;

}

productPicker.SelectedIndex = i;

quantityEntry.Text = service.Quantity.ToString();

quantityStepper.Value = service.Quantity;

dateDatePicker.Date = service.DateService;

}

private void QuantityStepper\_ValueChanged(object sender, ValueChangedEventArgs e)

{

quantityEntry.Text = quantityStepper.Value.ToString();

}

private void LoadProducts()

{

using (var da = new DataAccess())

{

products = da.GetList<Product>(false).OrderBy(p => p.Description).ToList();

}

foreach (var product in products)

{

productPicker.Items.Add(product.Description);

}

}

}

1. In the folder Pages, add the QueriesPage:

<?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="Services.Pages.QueriesPage"

Title="Queries">

<ScrollView>

<StackLayout Padding="15" Spacing="10">

<StackLayout Orientation="Horizontal">

<Label Text="Date: "

HorizontalOptions="Start"

VerticalOptions="Center"/>

<DatePicker x:Name="dateDatePicker"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

</StackLayout>

<ListView x:Name="servicesListView"/>

<StackLayout Orientation="Horizontal">

<Label Text="Total: "

HorizontalOptions="Start"

VerticalOptions="Center"/>

<Entry x:Name="totalEntry"

IsEnabled="False"

FontSize="20"

FontAttributes="Bold"

HorizontalTextAlignment="End"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"/>

</StackLayout>

</StackLayout>

</ScrollView>

</ContentPage>

public partial class QueriesPage : ContentPage

{

public QueriesPage()

{

InitializeComponent();

servicesListView.ItemTemplate = new DataTemplate(typeof(ServiceCell));

servicesListView.RowHeight = 70;

dateDatePicker.Date = DateTime.Today;

LoadServices();

dateDatePicker.DateSelected += DateDatePicker\_DateSelected;

}

private void DateDatePicker\_DateSelected(object sender, DateChangedEventArgs e)

{

LoadServices();

}

private void LoadServices()

{

using (var da = new DataAccess())

{

var list = da.GetList<Service>(true)

.Where(s => s.DateService.Year == dateDatePicker.Date.Year &&

s.DateService.Month == dateDatePicker.Date.Month &&

s.DateService.Day == dateDatePicker.Date.Day)

.ToList();

var total = list.Sum(l => l.Value);

servicesListView.ItemsSource = list;

totalEntry.Text = string.Format("{0:C2}", total);

}

}

}