Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по учебной практике

УП 01.01

Выполнил: Максимов Кирилл Эдуардовчи

Группа: ПР-21

Преподаватель: Мирошниченко Г.В.

2023

Содержание

[1.](#_gjdgxs) Индивидуальное задание №6.2 (вариант - 6) 3

[1.1](#_30j0zll) Описание задачи 3

[1.2](#_1fob9te) Структура проекта 3

[1.3](#_3znysh7) Описание разработанных функций 19

[1.4](#_2et92p0) Алгоритм решения 22

[1.5](#_tyjcwt) Используемые библиотеки 22

[1.6](#_3dy6vkm) Тестовые случаи 22

[1.7](#_1t3h5sf) Используемые инструменты 22

[1.8](#_4d34og8) Описание пользовательского интерфейса 23

[1.9](#_2s8eyo1) Приложение (print screen) 26

# Индивидуальное задание №6.2 (вариант - 6)

## Описание задачи

Необходимо создать приложение для базы данных "Колледж" с использованием асинхронных методов работы с базой данных в среде C# Xamarin (Visual Studio).

Требования к проекту:

На главной странице приложения должна быть реализована авторизация и выбор режима работы (для приемной комиссии, для преподавателя, для студента). Приложение должно содержать не менее 4-5 окон и использовать оптимальные методы навигации для приложения (например, TabbedPage, CarouselPage, Flyout Page, Navigation) - необходимо использовать не менее 2 видов навигации.

## Структура проекта

1. MainPage.xaml: Файл разметки XAML для главной страницы:

<?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="CollegeApp.MainPage"

NavigationPage.HasNavigationBar="False"

StyleClass="contentpage">

<StackLayout>

<StackLayout.Resources>

<StyleSheet Source="./styles.css" />

</StackLayout.Resources>

<Label Text="Войдите в аккаунт"

FontSize="Title"

StyleId="header"

HorizontalOptions="Center"

VerticalOptions="Center"

Padding="30,10,30,0"/>

<Picker x:Name="myPicker"

StyleId="picker"

Title="режим работы: (студент/преподователь)"

HorizontalOptions="Center"

VerticalOptions="Center">

<Picker.ItemsSource>

<x:Array Type="{x:Type x:String}">

<x:String>Студент</x:String>

<x:String>Преподователь</x:String>

</x:Array>

</Picker.ItemsSource>

</Picker>

<Entry x:Name="userNameEntry"

FontSize="17"

Margin="20,30,20,0"

Grid.ColumnSpan="3"

Placeholder="Логин"

PlaceholderColor="Gray"

HorizontalTextAlignment="Center"

MaxLength="20"/>

<Entry x:Name="passwordEntry"

FontSize="17"

Margin="20,-5,20,0"

IsPassword="True"

Grid.ColumnSpan="3"

Placeholder="Пароль"

PlaceholderColor="Gray"

HorizontalTextAlignment="Center"

MaxLength="20"/>

<Button x:Name="loginButton"

Text="Войти"

Margin="20,30,20,0"

StyleId="button"

Clicked="loginButton\_Clicked"/>

<Button x:Name="registerButton"

Text="Регистрация"

Margin="20,30,20,0"

StyleId="button"

Clicked="registerButton\_Clicked"/>

</StackLayout>

</ContentPage>

1. MainPage.xaml.cs: Файл, содержащий логику и обработчики событий для главной страницы:

using Microsoft.Data.Sqlite;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using SQLite;

using System.IO;

using CollegeApp.Models;

namespace CollegeApp

{

public partial class MainPage : ContentPage

{

private CollegeDatabase \_database;

public MainPage()

{

InitializeComponent();

\_database = new CollegeDatabase(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "collegeDB"));

}

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

{

if (myPicker.SelectedIndex == -1)

{

\_ = DisplayAlert("Внимание пользователь!", "Выберите режим работы.", "OK");

return;

}

else if (string.IsNullOrEmpty(userNameEntry.Text) || string.IsNullOrEmpty(passwordEntry.Text))

{

\_ = DisplayAlert("Внимание пользователь!", "Заполните логин и пароль.", "OK");

return;

}

string login = userNameEntry.Text;

string password = passwordEntry.Text;

string selectedMode = myPicker.SelectedItem.ToString();

if (selectedMode == "Студент")

{

Student student = await \_database.FindStudentByLoginAndPasswordAsync(login, password);

if (student != null)

{

await Navigation.PushModalAsync(new StudentPage(student));

}

else

{

await DisplayAlert("Ошибка", "Неверный логин или пароль.", "OK");

}

}

else if (selectedMode == "Преподователь" || selectedMode == "Приемная комиссия")

{

Teacher teacher = await \_database.FindTeacherByLoginAndPasswordAsync(login, password);

if (teacher != null)

{

if (selectedMode == "Приемная комиссия" && !teacher.IsAdmission)

{

await DisplayAlert("Ошибка", "У вас нет доступа к приемной комиссии.", "OK");

return;

}

if (selectedMode == "Преподователь")

{

await Navigation.PushModalAsync(new TeacherPage(teacher));

}

}

else

{

await DisplayAlert("Ошибка", "Неверный логин или пароль.", "OK");

}

}

}

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

{

await Navigation.PushModalAsync(new RegisterPage());

}

}

}

1. StudentPage.xaml: Файл разметки XAML для страницы студента:

<?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="CollegeApp.StudentPage"

xmlns:converters="clr-namespace:CollegeApp"

StyleClass="contentpage">

<ContentPage.Resources>

<StyleSheet Source="/styles.css" />

<converters:IsBudgetConverter x:Key="IsBudgetConverter" />

</ContentPage.Resources>

<ContentPage.Content>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<BoxView Grid.Row="0" Grid.Column="0" Grid.RowSpan="2" Grid.ColumnSpan="2" BackgroundColor="Gray" />

<Label Grid.Row="0" Grid.Column="0" Text="{Binding FullName}" StyleClass="label-black" Grid.ColumnSpan="2" HorizontalTextAlignment="Center"/>

<Label Grid.Row="2" Grid.Column="0" Text="Дата рождения:" StyleClass="label" />

<Label Grid.Row="2" Grid.Column="1" Text="{Binding DateOfBirth, Converter={StaticResource DateFormatConverter}}" StyleClass="label-value" />

<Label Grid.Row="3" Grid.Column="0" Text="Форма обучения:" StyleClass="label" />

<Label Grid.Row="3" Grid.Column="1" Text="{Binding IsBudget, Converter={StaticResource IsBudgetConverter}}" StyleClass="label-value" />

<Label Grid.Row="4" Grid.Column="0" Text="Обучение на курсе:" StyleClass="label" />

<Label Grid.Row="4" Grid.Column="1" Text="{Binding Course}" StyleClass="label-value" />

<Label Grid.Row="5" Grid.Column="0" Text="Группа:" StyleClass="label" />

<Label Grid.Row="5" Grid.Column="1" x:Name="GroupName" StyleClass="label-value" />

<Label Grid.Row="6" Grid.Column="0" Text="Специальность(-и):" StyleClass="label" />

<Label Grid.Row="6" Grid.Column="1" x:Name="StudentSpecialties" StyleClass="label-value" />

</Grid>

</ContentPage.Content>

</ContentPage>

1. StudentPage.xaml.cs: Файл, содержащий логику и обработчики событий для страницы студента:

using CollegeApp.Models;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace CollegeApp

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class StudentPage : ContentPage

{

private Student student;

private CollegeDatabase \_database;

public StudentPage(Student student)

{

InitializeComponent();

\_database = new CollegeDatabase(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "collegeDB"));

this.student = student;

Title = "Добро пожаловать, студент!";

BindingContext = this.student;

\_ = LoadStudentSpecialtiesAsync();

\_ = LoadGroupNameAsync();

}

private async Task LoadStudentSpecialtiesAsync()

{

List<Specialization> studentSpecialties = await \_database.GetStudentSpecialtiesAsync(student.StudentId);

string specialtiesText = string.Join(", ", studentSpecialties.Select(s => s.Name));

StudentSpecialties.Text = specialtiesText;

}

private async Task LoadGroupNameAsync()

{

string groupName = await \_database.GetGroupNameAsync(student.GroupId);

GroupName.Text = groupName;

}

}

}

1. TeacherPage.xaml: Файл разметки XAML для страницы учителя:

<?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="CollegeApp.TeacherPage"

StyleClass="contentpage">

<ContentPage.Resources>

<StyleSheet Source="./styles.css" />

</ContentPage.Resources>

<ContentPage.Content>

<StackLayout>

<Label Text="Группы." HorizontalOptions="Center" FontSize="Title" Margin="0,20,0,10" StyleClass="label" TextDecorations="Underline" />

<Label Text="Нажмите на нужную вам группу, чтобы увидеть список студентов." HorizontalOptions="Center" Margin="0,0,0,10" StyleClass="label" HorizontalTextAlignment="Center" />

<ListView x:Name="GroupListView"

ItemsSource="{Binding GroupNames}"

ItemSelected="Handle\_GroupSelected"

CachingStrategy="RecycleElement">

<ListView.ItemTemplate>

<DataTemplate>

<TextCell Text="{Binding}" TextColor="White"/>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</StackLayout>

</ContentPage.Content>

</ContentPage>

1. TeacherPage.xaml.cs: Файл, содержащий логику и обработчики событий для страницы учителя:

using CollegeApp.Models;

using System;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace CollegeApp

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class TeacherPage : ContentPage

{

private Teacher teacher;

private CollegeDatabase database;

public ObservableCollection<string> GroupNames { get; set; }

public TeacherPage(Teacher teacher)

{

InitializeComponent();

Title = $"Добро пожаловать, {teacher.FullName}!";

this.teacher = teacher;

database = new CollegeDatabase(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "collegeDB"));

GroupNames = new ObservableCollection<string>();

GroupListView.ItemsSource = GroupNames;

}

protected override async void OnAppearing()

{

base.OnAppearing();

await LoadGroupNames();

}

private async Task LoadGroupNames()

{

var groupNames = await database.GetAllGroupNamesAsync();

GroupNames.Clear();

foreach (var groupName in groupNames)

{

GroupNames.Add(groupName);

}

}

async void Handle\_GroupSelected(object sender, SelectedItemChangedEventArgs e)

{

if (e.SelectedItem == null)

return;

var selectedGroupName = (string)e.SelectedItem;

await Navigation.PushModalAsync(new StudentsPage(selectedGroupName));

((ListView)sender).SelectedItem = null;

}

}

}

}

1. RegisterPage.xaml: Файл разметки XAML для регистрации

<?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="CollegeApp.RegisterPage"

StyleClass="contentpage">

<ContentPage.Resources>

<StyleSheet Source="./styles.css"/>

</ContentPage.Resources>

<StackLayout Orientation="Vertical" HorizontalOptions="Center">

<StackLayout Orientation="Horizontal" HorizontalOptions="Center">

<Label x:Name="header"

StyleId="header"

Text="Регистрация"

FontSize="25"/>

</StackLayout>

<StackLayout Orientation="Horizontal" HorizontalOptions="Center">

<Entry x:Name="fullNameEntry"

Placeholder="FullName"

StyleId="entry" Keyboard="Text"

HorizontalOptions="Center"

WidthRequest="400"/>

</StackLayout>

<StackLayout Orientation="Horizontal" HorizontalOptions="Center">

<DatePicker x:Name="dateOfBirthDatePicker"

StyleId="entry"

HorizontalOptions="Center"

WidthRequest="400"/>

</StackLayout>

<StackLayout Orientation="Horizontal" HorizontalOptions="Center">

<Entry x:Name="numberCourseEntry"

StyleId="entry"

Keyboard="Numeric"

HorizontalOptions="Center"

Placeholder="Number of course"

WidthRequest="400"/>

</StackLayout>

<StackLayout Orientation="Horizontal" HorizontalOptions="Center">

<Entry x:Name="loginEntry"

Placeholder="Login"

StyleId="entry"

HorizontalOptions="Center"

WidthRequest="400"/>

</StackLayout>

<StackLayout Orientation="Horizontal" HorizontalOptions="Center">

<Entry x:Name="passwordEntry"

Placeholder="Password"

IsPassword="True" StyleId="entry"

HorizontalOptions="Center"

WidthRequest="400"/>

</StackLayout>

<StackLayout Orientation="Horizontal" HorizontalOptions="Center">

<Entry x:Name="repitpasswordEntry"

Placeholder="Repit Password"

IsPassword="True"

StyleId="entry"

HorizontalOptions="Center"

WidthRequest="400"/>

</StackLayout>

<StackLayout Orientation="Horizontal" HorizontalOptions="Center">

<Entry x:Name="groupIdEntry"

Placeholder="Group ID"

StyleId="entry"

Keyboard="Numeric"

HorizontalOptions="Center"

WidthRequest="400"/>

</StackLayout>

<StackLayout Orientation="Horizontal" HorizontalOptions="Center">

<CheckBox x:Name="budgetCheckBox"

HorizontalOptions="Center"

Color="White"/>

<Label Text="Бюджет?"

HorizontalOptions="Center"

StyleId="label"/>

</StackLayout>

<StackLayout Orientation="Horizontal" HorizontalOptions="Center">

<Button x:Name="Regiter"

StyleId="button"

Text="Регистрация"

WidthRequest="400"

Clicked="Regiter\_Clicked"/>

</StackLayout>

<StackLayout Orientation="Horizontal" >

<Button x:Name ="BackToLogin"

Text="Назад" StyleId="button"

WidthRequest="400"

Clicked="BackToLogin\_Clicked"/>

</StackLayout>

</StackLayout>

</ContentPage>

1. RegisterPage.xaml.cs: Файл, содержащий логику и обработчики событий для страницы приемной комиссии:

using CollegeApp.Models;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace CollegeApp

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class RegisterPage : ContentPage

{

private CollegeDatabase \_database;

public RegisterPage ()

{

InitializeComponent ();

\_database = new CollegeDatabase(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "collegeDB"));

}

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

{

string fullName = fullNameEntry.Text;

string dateOfBirth = dateOfBirthDatePicker.Date.ToString("dd/MM/yyyy");

bool budget = budgetCheckBox.IsChecked;

string course = numberCourseEntry.Text;

string login = loginEntry.Text;

string password = passwordEntry.Text;

string repitPassword = repitpasswordEntry.Text;

Student student = await \_database.FindStudentByLoginAndPasswordAsync(login, password);

int groupID = int.Parse(groupIdEntry.Text);

if (string.IsNullOrEmpty(course) || string.IsNullOrEmpty(dateOfBirth) ||

string.IsNullOrEmpty(fullName) || string.IsNullOrEmpty(login) ||

string.IsNullOrEmpty(password) || string.IsNullOrEmpty(repitPassword) || string.IsNullOrEmpty(groupIdEntry.Text))

{

await DisplayAlert("Ошибка", "Заполните поля", "OK");

return;

}

else if (password != repitPassword)

{

await DisplayAlert("Ошибка", "Пароли не совпадают", "OK");

return;

}

if (student != null)

{

await DisplayAlert("Ошибка", "Студент с таким логином уже есть", "OK");

return;

}

else

{

Student studentAdd = new Student

{

FullName = fullName,

DateOfBirth = dateOfBirth,

Course = int.Parse(course),

Login = login,

Password = password,

GroupId = groupID,

IsBudget = budget,

};

await \_database.SaveStudentAsync(studentAdd);

await DisplayAlert("Информация", "Регистрация прошла успешно", "OK");

fullNameEntry.Text = "";

dateOfBirthDatePicker = null;

budgetCheckBox.IsChecked = false;

numberCourseEntry.Text = "";

loginEntry.Text = "";

passwordEntry.Text = "";

repitpasswordEntry.Text = "";

}

}

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

{

await Navigation.PushModalAsync(new MainPage());

}

}

}

1. StudentsPage.xaml: Файл разметки XAML для страницы списка студентов:

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

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

x:Class="CollegeApp.StudentsPage"

StyleClass="contentpage">

<ContentPage.Resources>

<StyleSheet Source="./styles.css"/>

</ContentPage.Resources>

<ContentPage.Content>

<StackLayout>

<ListView x:Name="StudentListView"

ItemsSource="{Binding Students}"

ItemSelected="Handle\_StudentSelected"

CachingStrategy="RecycleElement">

<ListView.ItemTemplate>

<DataTemplate>

<TextCell Text="{Binding FullName}" TextColor="White"/>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</StackLayout>

</ContentPage.Content>

</ContentPage>

1. StudentsPage.xaml.cs: Файл, содержащий логику и обработчики событий для страницы списка студентов:

using CollegeApp.Models;

using System;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace CollegeApp

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class StudentsPage : ContentPage

{

private string selectedGroupName;

private CollegeDatabase database;

public ObservableCollection<Student> Students { get; set; }

public StudentsPage(string selectedGroupName)

{

InitializeComponent();

Title = $"Студенты группы {selectedGroupName}";

this.selectedGroupName = selectedGroupName;

database = new CollegeDatabase(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "collegeDB"));

Students = new ObservableCollection<Student>();

StudentListView.ItemsSource = Students;

}

protected override async void OnAppearing()

{

base.OnAppearing();

await LoadStudents();

}

private async Task LoadStudents()

{

var students = await database.GetStudentsByGroupNameAsync(selectedGroupName);

Students.Clear();

foreach (var student in students)

{

Students.Add(student);

}

}

async void Handle\_StudentSelected(object sender, SelectedItemChangedEventArgs e)

{

if (e.SelectedItem == null) return;

var selectedStudent = (Student)e.SelectedItem;

await Navigation.PushAsync(new StudentPage(selectedStudent));

((ListView)sender).SelectedItem = null;

}

}

}

1. База данных:

public class Group

{

[PrimaryKey, AutoIncrement]

public int GroupId { get; set; }

public string Name { get; set; }

public int SpecializationId { get; set; }

}

public class Specialization

{

[PrimaryKey, AutoIncrement]

public int SpecializationId { get; set; }

public string Name { get; set; }

}

public class Student

{

[PrimaryKey, AutoIncrement]

public int StudentId { get; set; }

public string FullName { get; set; }

public string DateOfBirth { get; set; }

public bool IsBudget { get; set; }

public int Course { get; set; }

public int GroupId { get; set; }

public string Login { get; set; }

public string Password { get; set; }

}

public class StudentSpecialization

{

[PrimaryKey, AutoIncrement]

public int StudentSpecializationId { get; set; }

public int StudentId { get; set; }

public int SpecializationId { get; set; }

}

public class Teacher

{

[PrimaryKey, AutoIncrement]

public int TeacherId { get; set; }

public string FullName { get; set; }

public int Workload { get; set; }

public int SpecializationId { get; set; }

public string Login { get; set; }

public string Password { get; set; }

public bool IsAdmission { get; set; }

}

public class TeacherSpecialization

{

[PrimaryKey, AutoIncrement]

public int TeacherSpecializationId { get; set; }

public int TeacherId { get; set; }

public int SpecialtyId { get; set; }

}

1. Файл с инициализацией БД и с обращениями к БД:

using CollegeApp.Models;

using SQLite;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CollegeApp

{

public class CollegeDatabase

{

readonly SQLiteAsyncConnection connect;

public CollegeDatabase(string path)

{

connect = new SQLiteAsyncConnection(path);

connect.CreateTableAsync<Student>().Wait();

connect.CreateTableAsync<Group>().Wait();

connect.CreateTableAsync<Specialization>().Wait();

connect.CreateTableAsync<Teacher>().Wait();

connect.CreateTableAsync<StudentSpecialization>().Wait();

connect.CreateTableAsync<TeacherSpecialization>().Wait();

if (connect.Table<Student>().CountAsync().Result == 0)

{

var specialtyes = new List<Specialization>

{

new Specialization { Name = "Информационные системы и программирование" },

new Specialization { Name = "Технология аналитического контроля химических соединений" },

};

connect.InsertAllAsync(specialtyes).Wait();

var groups = new List<Group>

{

new Group { Name = "ПР-21", SpecializationId = 1 },

new Group { Name = "ДЗ-11", SpecializationId = 1 },

};

connect.InsertAllAsync(groups).Wait();

var teachers = new List<Teacher>

{

new Teacher { FullName = "Ватолина Татьяна Александровна", Workload = 100, Login = "Teacher1", IsAdmission = false, Password = "qwerty" },

new Teacher { FullName = "Мирошниченко Галина Викторовна", Workload = 200, Login = "Teacher2", IsAdmission = true, Password = "qwerty" },

};

connect.InsertAllAsync(teachers).Wait();

var students = new List<Student>

{

new Student { FullName = "Максимов Кирилл Эдуардович", DateOfBirth = "30.05.2005", IsBudget = true, Course = 2, GroupId = 1, Login = "Maksimov", Password = "qwerty"},

new Student { FullName = "Шайдулин Артём Ришатович", DateOfBirth = "26.04.2005", IsBudget = true, Course = 1, GroupId = 2, Login = "Student2", Password = "qwerty"},

};

connect.InsertAllAsync(students).Wait();

var studentSpecialties = new List<StudentSpecialization>

{

new StudentSpecialization { StudentId = 1, SpecializationId = 1 },

new StudentSpecialization { StudentId = 2, SpecializationId = 1 },

new StudentSpecialization { StudentId = 2, SpecializationId = 2 },

};

connect.InsertAllAsync(studentSpecialties).Wait();

var teacherSpecialties = new List<TeacherSpecialization>

{

new TeacherSpecialization { TeacherId = 1, SpecialtyId = 1 },

new TeacherSpecialization { TeacherId = 2, SpecialtyId = 1 },

};

connect.InsertAllAsync(teacherSpecialties).Wait();

}

}

public async Task<Student> FindStudentByLoginAndPasswordAsync(string login, string password)

{

return await connect.Table<Student>()

.Where(s => s.Login == login && s.Password == password)

.FirstOrDefaultAsync();

}

public async Task<Teacher> FindTeacherByLoginAndPasswordAsync(string login, string password)

{

return await connect.Table<Teacher>()

.Where(t => t.Login == login && t.Password == password)

.FirstOrDefaultAsync();

}

public Task<List<Specialization>> GetStudentSpecialtiesAsync(int studentId)

{

return connect.QueryAsync<Specialization>(

"SELECT Specialty.\* FROM Specialty " +

"JOIN StudentSpecialty ON Specialty.SpecialtyId = StudentSpecialty.SpecialtyId " +

"WHERE StudentSpecialty.StudentId = ?", studentId);

}

public Task<string> GetGroupNameAsync(int groupId)

{

return connect.ExecuteScalarAsync<string>(

"SELECT Name FROM [Group] WHERE GroupId = ?", groupId);

}

public async Task<List<string>> GetAllGroupNamesAsync()

{

var groupNames = await connect.Table<Group>().ToListAsync();

return groupNames.Select(group => group.Name).ToList();

}

public async Task<List<Student>> GetStudentsByGroupNameAsync(string groupName)

{

var group = await connect.Table<Group>().FirstOrDefaultAsync(g => g.Name == groupName);

if (group != null)

{

var students = await connect.Table<Student>().Where(s => s.GroupId == group.GroupId).ToListAsync();

return students;

}

return new List<Student>();

}

public async Task<List<Student>> GetStudentsAsync()

{

return await connect.Table<Student>().ToListAsync();

}

public async Task SaveStudentAsync(Student student)

{

if (student.StudentId != 0)

{

await connect.UpdateAsync(student);

}

else

{

await connect.InsertAsync(student);

}

}

public async Task DeleteStudentAsync(Student student)

{

if (student != null)

{

await connect.DeleteAsync(student);

}

}

public Task<int> SaveTeacherAsync(Teacher teacher)

{

if (teacher.TeacherId != 0)

{

return connect.UpdateAsync(teacher);

}

else

{

return connect.InsertAsync(teacher);

}

}

public Task<List<Teacher>> GetTeachersAsync()

{

return connect.Table<Teacher>().ToListAsync();

}

public async Task<List<Group>> GetGroupsAsync()

{

return await connect.Table<Group>().ToListAsync();

}

public async Task<Group> GetGroupByNameAsync(string groupName)

{

return await connect.Table<Group>().FirstOrDefaultAsync(g => g.Name == groupName);

}

}

}

Это основные файлы проекта, которые определены в коде.

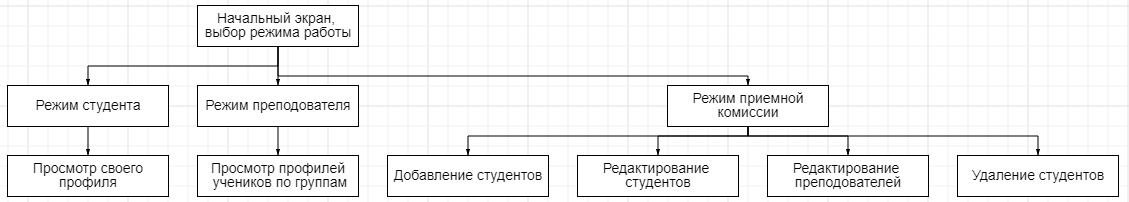
## Описание разработанных функций

Вот описание функций, реализованных в коде:

1. Метод **LoadStudentSpecialtiesAsync()**: Метод для загрузки специальности студента из базы данных и отображает их на странице.
2. Метод **LoadGroupNameAsync()**: Метод для загрузки названий группы студента из базы данных и отображает его на странице.
3. Метод **LoadStudentPhotoAsync()** Метод для загрузки фотографии студента и отображает ее на странице.
4. Метод **OnAppearing()**: Метод вызывается, когда страница TeacherPage становится видимой, инициирует загрузку названий групп из базы данных.
5. Метод **LoadGroupNames()**: Метод загружает и отображает названия групп из базы данных.
6. **Handle\_GroupSelected**(object sender, SelectedItemChangedEventArgs e) Обработчик события, вызывается при выборе элемента в списке групп. Открывает страницу StudentsPage для выбранной группы.
7. **LoadStudents()**: Метод для загрузки списка студентов из базы данных и отображения их в списке studentsListView. Также загружает список групп и устанавливает его в качестве источника данных для groupPicker.
8. **LoadTeachers()**: Метод для загрузки списка преподавателей из базы данных и отображения их в списке teachersListView.
9. **AddStudent\_Clicked**(object sender, EventArgs e): Обработчик события клика на кнопку "Добавить" для добавления нового студента. Метод проверяет заполненность всех необходимых полей ввода (полное имя, курс, логин и пароль) и выбор группы. Затем создает новый объект Student с заполненными значениями и сохраняет его в базе данных. После сохранения поля ввода очищаются, и список студентов обновляется.
10. **StudentsListView\_ItemSelected**(object sender, SelectedItemChangedEventArgs e): Обработчик события выбора элемента списка студентов. Метод сохраняет выбранного студента в переменной \_selectedStudent и заполняет поля редактирования соответствующими значениями.
11. **SaveStudent\_Clicked**(object sender, EventArgs e): Обработчик события клика на кнопку "Сохранить" при редактировании студента. Метод обновляет выбранного студента с новыми значениями из полей редактирования и сохраняет изменения в базе данных. Затем очищает поля редактирования и обновляет список студентов.
12. **DeleteStudent\_Clicked**(object sender, EventArgs e): Обработчик события клика на кнопку "Удалить" при удалении студента. Метод удаляет выбранного студента из базы данных, очищает поля редактирования и обновляет список студентов.
13. **DeleteStudentsListView\_ItemSelected**(object sender, SelectedItemChangedEventArgs e): Обработчик события выбора элемента списка студентов при удалении. Метод сохраняет выбранного студента в переменной \_selectedStudent.
14. **EditFilterEntry\_TextChanged**(object sender, TextChangedEventArgs e): Обработчик события изменения текста в поле фильтрации при редактировании студентов. Метод фильтрует список студентов на основе введенного фильтра и обновляет отображаемые элементы в списке.
15. **DeleteFilterEntry\_TextChanged**(object sender, TextChangedEventArgs e): Обработчик события изменения текста в поле фильтрации при удалении студентов. Метод фильтрует список студентов на основе введенного фильтра и обновляет отображаемые элементы в списке.
16. **SaveTeacher\_Clicked**(object sender, EventArgs e): Обработчик события клика на кнопку "Сохранить" при редактировании преподавателя. Метод обновляет выбранного преподавателя с новыми значениями из полей редактирования и сохраняет изменения в базе данных. Затем очищает поля редактирования и обновляет список преподавателей.
17. **DeleteTeacher\_Clicked**(object sender, EventArgs e): Обработчик события клика на кнопку "Удалить" при удалении преподавателя. Метод удаляет выбранного преподавателя из базы данных, очищает поля редактирования и обновляет список преподавателей.
18. **DeleteTeachersListView\_ItemSelected**(object sender, SelectedItemChangedEventArgs e): Обработчик события выбора элемента списка преподавателей при удалении. Метод сохраняет выбранного преподавателя в переменной \_selectedTeacher.
19. **EditTeacherFilterEntry\_TextChanged**(object sender, TextChangedEventArgs e): Обработчик события изменения текста в поле фильтрации при редактировании преподавателей. Метод фильтрует список преподавателей на основе введенного фильтра и обновляет отображаемые элементы в списке.
20. **DeleteTeacherFilterEntry\_TextChanged**(object sender, TextChangedEventArgs e): Обработчик события изменения текста в поле фильтрации при удалении преподавателей. Метод фильтрует список преподавателей на основе введенного фильтра и обновляет отображаемые элементы в списке.

Это основные функции, позволяющие загружать данные о студентах и преподавателях из базы данных, добавлять, редактировать и удалять записи, а также фильтровать списки по различным критериям. Обработчики событий реагируют на действия пользователя, например, нажатие кнопки или выбор элемента в списке, и выполняют соответствующие операции с данными.

## Алгоритм решения



## Используемые библиотеки

В проекте используются следующие библиотеки:

1. **Xamarin.Forms**.
2. **Xamarin.Forms.Xaml**.
3. **System**.
4. **System.Collections.Generic**;
5. **System.Linq**;
6. **SQLite**;
7. **System.IO**.

Это основные библиотеки, которые я использовал в проекте.

## Тестовые случаи

Имеется проверка на ввод пустых данных в поля.

## Используемые инструменты

* Язык программирования: C#
* Среда разработки: Xamarin
* Фреймворк: Xamarin.Forms

## Описание пользовательского интерфейса

Описание пользовательского интерфейса:

1. MainPage:

* Элементы на странице:
  + Label с текстом "College!" в шапке страницы.
  + Label с приветствием и инструкцией для выбора режима работы.
  + Picker (myPicker) для выбора режима работы (студент/преподаватель/приемная комиссия).
  + Entry для ввода логина (userNameEntry).
  + Entry для ввода пароля (passwordEntry).
  + Button (loginButton) для выполнения входа.

1. StudentPage:

* Название страницы: "Добро пожаловать, студент!"
* Элементы на странице:
  + Label с именем студента (FullName), расположенный по центру страницы.
  + Label с текстом "Дата рождения:" и соответствующим значением даты рождения студента (DateOfBirth).
  + Label с текстом "Форма обучения:" и соответствующим значением формы обучения студента (IsBudget).
  + Label с текстом "Обучение на курсе:" и соответствующим значением курса студента (Course).
  + Label с текстом "Группа:" и соответствующим названием группы студента (GroupName).
  + Label с текстом "Специальность(-и):" и соответствующими названиями специальностей студента (StudentSpecialties).

1. TeacherPage:

* Элементы на странице:
  + Label с приветствием, содержащим имя преподавателя (FullName).
  + ListView (GroupListView) со списком названий групп.
    - При выборе группы открывается страница со списком студентов этой группы.

1. ComissionPage:

* Элементы на странице:
  + ListView (studentsListView) со списком студентов, которые могут быть удалены.
  + ListView (deleteStudentsListView) со списком студентов, которые могут быть удалены.
  + Entry для ввода полного имени студента (fullNameEntry).
  + DatePicker для выбора даты рождения студента (dateOfBirthPicker).
  + Switch для выбора формы обучения студента (isBudgetSwitch).
  + Entry для ввода курса студента (courseEntry).
  + Picker (groupPicker) для выбора группы студента.
  + Entry для ввода логина студента (loginEntry).
  + Entry для ввода пароля студента (passwordEntry).
  + Button (addStudentButton) для добавления нового студента.
  + Entry для редактирования полного имени студента (fullNameEditEntry).
  + DatePicker для редактирования даты рождения студента (dateOfBirthEditPicker).
  + Picker (groupEditPicker) для выбора группы студента при редактировании.
  + Entry для редактирования курса студента (courseEditEntry).
  + Entry для редактирования логина студента (loginEditEntry).
  + Entry для редактирования пароля студента (passwordEditEntry).
  + Button (saveStudentButton) для сохранения отредактированных данных студента.
  + Button (deleteStudentButton) для удаления выбранного студента из списка.

1. GroupPage:

* Элементы на странице:
  + Label с названием группы (GroupName).
  + ListView (studentsListView) со списком студентов в группе.
    - При выборе студента открывается страница с подробной информацией о студенте.

1. StudentDetailsPage:

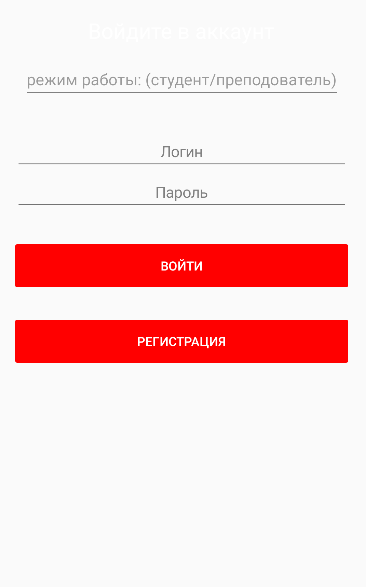
* Элементы на странице:
  + Label с именем студента (FullName).
  + Изображение студента (StudentPhoto).
  + Label с текстом "Дата рождения:" и соответствующим значением даты рождения студента (DateOfBirth).
  + Label с текстом "Форма обучения:" и соответствующим значением формы обучения студента (IsBudget).
  + Label с текстом "Обучение на курсе:" и соответствующим значением курса студента (Course).
  + Label с текстом "Группа:" и соответствующим названием группы студента (GroupName).
  + Label с текстом "Специальность(-и):" и соответствующими названиями специальностей студента (StudentSpecialties).
  + Button (editButton) для редактирования информации о студенте.

1. EditStudentPage:

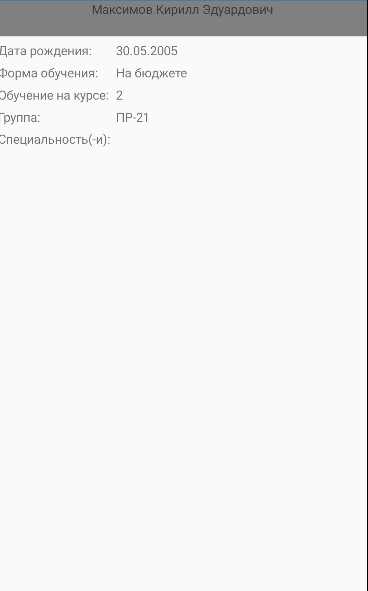
* Элементы на странице:
  + Entry для редактирования полного имени студента (fullNameEditEntry).
  + DatePicker для редактирования даты рождения студента (dateOfBirthEditPicker).
  + Switch для редактирования формы обучения студента (isBudgetEditSwitch).
  + Entry для редактирования курса студента (courseEditEntry).
  + Picker (groupEditPicker) для выбора группы студента.
  + Entry для редактирования логина студента (loginEditEntry).
  + Entry для редактирования пароля студента (passwordEditEntry).
  + Button (saveChangesButton) для сохранения отредактированных данных студента.

## Приложение (print screen)

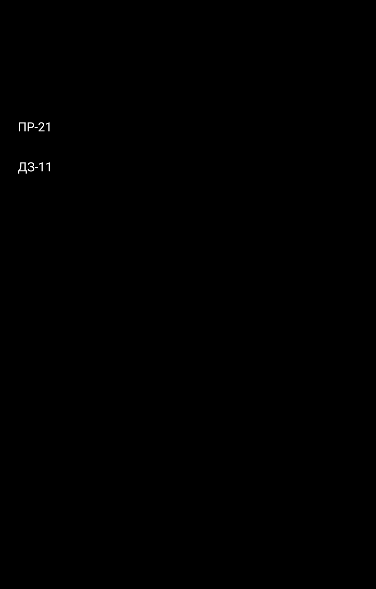
1. Главная страница



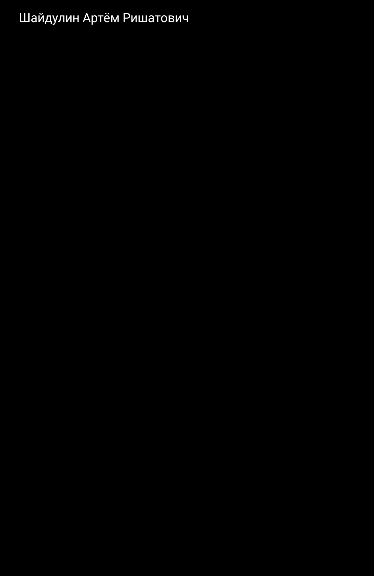
1. Режим студента:



1. Режим преподавателя (выбор группы):



1. Режим преподавателя (выбор студента в группе):



1. Регистрация:
2. 