Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по учебной практике

УП 01.01

Выполнил: Гордейчик Степан Алексеевич

Группа: ПР-21

Преподаватель: Мирошниченко Г.В.

2023

Содержание

[1. Индивидуальное задание №6.2 (вариант - 6) 3](#_Toc138711989)

[1.1 Описание задачи 3](#_Toc138711990)

[1.2 Структура проекта 3](#_Toc138711991)

[1.3 Описание разработанных функций 19](#_Toc138711992)

[1.4 Алгоритм решения 22](#_Toc138711993)

[1.5 Используемые библиотеки 22](#_Toc138711994)

[1.6 Тестовые случаи 22](#_Toc138711995)

[1.7 Используемые инструменты 22](#_Toc138711996)

[1.8 Описание пользовательского интерфейса 23](#_Toc138711997)

[1.9 Приложение (print screen) 26](#_Toc138711998)

# 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, FlyoutPage, 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">

<StackLayout>

<StackLayout.Resources>

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

</StackLayout.Resources>

<Frame BackgroundColor="#2196F3" Padding="24" CornerRadius="0">

<Label Text="College!"

HorizontalTextAlignment="Center"

TextColor="White" FontSize="36"/>

</Frame>

<Label Text="Доброго времени суток, выберите режим работы:"

FontSize="Title"

HorizontalOptions="Center"

VerticalOptions="Center"

Padding="30,10,30,0"

StyleId="welcomeLabel"/>

<Picker x:Name="myPicker"

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

HorizontalOptions="Center"

VerticalOptions="Center"

TextColor="White"

StyleId="welcomeLabel">

<Picker.ItemsSource>

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

<x:String>Студент</x:String>

<x:String>Преподователь</x:String>

<x:String>Приемная комиссия</x:String>

</x:Array>

</Picker.ItemsSource>

</Picker>

<Entry x:Name="userNameEntry"

FontSize="17"

Margin="80,30,80,0"

Grid.ColumnSpan="3"

Placeholder="Логин"

PlaceholderColor="Gray"

HorizontalTextAlignment="Center"

MaxLength="20"/>

<Entry x:Name="passwordEntry"

FontSize="17"

Margin="80,-5,80,0"

IsPassword="True"

Grid.ColumnSpan="3"

Placeholder="Пароль"

PlaceholderColor="Gray"

HorizontalTextAlignment="Center"

MaxLength="20"/>

<Button x:Name="loginButton"

Text="Войти"

Margin="80,30,80,0"

StyleId="button"

Clicked="loginButton\_Clicked"/>

</StackLayout>

</ContentPage>

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

namespace CollegeApp

{

public partial class MainPage : ContentPage

{

private CollegeDatabase \_database;

public MainPage ()

{

InitializeComponent();

\_database = new CollegeDatabase(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "Database.db3"));

}

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.PushAsync(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.PushAsync(new TeacherPage(teacher));

}

else if (selectedMode == "Приемная комиссия")

{

await Navigation.PushAsync(new ComissionPage());

}

}

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

}

}

}

}

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">

<ContentPage.Resources>

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

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

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

</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"/>

<Image Grid.Row="1" Grid.Column="0" x:Name="StudentPhoto" Grid.ColumnSpan="2" Source="{Binding Photo}" Aspect="AspectFit" HorizontalOptions="Center" VerticalOptions="Center" WidthRequest="150" HeightRequest="150" />

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

<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-bold" />

<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-bold" />

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

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

<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-bold" />

<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:

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), "Database.db3"));

this.student = student;

Title = "Добро пожаловать, студент!";

BindingContext = this.student;

\_ = LoadStudentSpecialtiesAsync();

\_ = LoadGroupNameAsync();

\_ = LoadStudentPhotoAsync();

}

private async Task LoadStudentSpecialtiesAsync()

{

List<Specialty> 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;

}

private Task LoadStudentPhotoAsync()

{

if (string.IsNullOrEmpty(student.Photo))

{

student.Photo = "default\_photo.png";

}

var imageSource = ImageSource.FromFile(student.Photo);

StudentPhoto.Source = imageSource;

return Task.CompletedTask;

}

}

}

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">

<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-bold" TextDecorations="Underline" />

<Label Text="Нажмите на нужную вам группу, чтобы увидеть список студентов." HorizontalOptions="Center" Margin="0,0,0,10" StyleClass="label-small" 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:

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), "Database.db3"));

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.PushAsync(new StudentsPage(selectedGroupName));

((ListView)sender).SelectedItem = null;

}

}

}

1. ComissionPage.xaml: Файл разметки XAML для страницы приемной комиссии:

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

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

x:Class="CollegeApp.ComissionPage"

NavigationPage.HasNavigationBar="False">

<CarouselPage.Resources>

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

</CarouselPage.Resources>

<ContentPage>

<StackLayout>

<Label Text="Добавление студентов" HorizontalTextAlignment="Center" StyleClass="label-bold"/>

<Entry Placeholder="ФИО" x:Name="fullNameEntry" />

<DatePicker x:Name="dateOfBirthPicker" />

<Label Text="Форма обучения платная:" HorizontalTextAlignment="Start" Margin="3,0,0,0"/>

<Switch x:Name="isBudgetSwitch" IsToggled="true" HorizontalOptions="Center" Margin="0,-27,0,0"/>

<Entry Placeholder="Курс" x:Name="courseEntry" />

<Picker x:Name="groupPicker" Title="Выберите группу" />

<Entry Placeholder="Логин" x:Name="loginEntry" />

<Entry Placeholder="Пароль" x:Name="passwordEntry" />

<Button Text="Добавить" Clicked="AddStudent\_Clicked" StyleId="button"/>

</StackLayout>

</ContentPage>

<ContentPage>

<StackLayout>

<Label Text="Редактирование студентов" HorizontalTextAlignment="Center" StyleClass="label-bold"/>

<StackLayout>

<Label Text="Фильтр:" Margin="3,0,0,0"/>

<Entry x:Name="editFilterEntry" TextChanged="EditFilterEntry\_TextChanged" />

</StackLayout>

<ListView x:Name="studentsListView" ItemSelected="StudentsListView\_ItemSelected">

<ListView.ItemTemplate>

<DataTemplate>

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

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

<Entry Placeholder="ФИО" x:Name="fullNameEditEntry" />

<DatePicker x:Name="dateOfBirthEditPicker" />

<Label Text="Форма обучения платная:" HorizontalTextAlignment="Start" Margin="3,0,0,0"/>

<Switch x:Name="isBudgetEditSwitch" HorizontalOptions="Center" Margin="0,-27,0,0"/>

<Entry Placeholder="Курс" x:Name="courseEditEntry" />

<Entry Placeholder="ID группы" x:Name="groupIdEditEntry" />

<Entry Placeholder="Логин" x:Name="loginEditEntry" />

<Entry Placeholder="Пароль" x:Name="passwordEditEntry" />

<Button Text="Сохранить" Clicked="SaveStudent\_Clicked" StyleId="button"/>

</StackLayout>

</ContentPage>

<ContentPage>

<StackLayout>

<Label Text="Редактирование преподавателей" HorizontalTextAlignment="Center" StyleClass="label-bold"/>

<StackLayout>

<Label Text="Фильтр:" Margin="3,0,0,0"/>

<Entry x:Name="editTeacherFilterEntry" TextChanged="EditTeacherFilterEntry\_TextChanged" />

</StackLayout>

<ListView x:Name="teachersListView" ItemSelected="TeachersListView\_ItemSelected">

<ListView.ItemTemplate>

<DataTemplate>

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

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

<Entry Placeholder="ФИО" x:Name="fullNameEditTeacherEntry\_2" />

<Entry Placeholder="Нагрузка" x:Name="workloadEditEntry" />

<Entry Placeholder="Логин" x:Name="loginEditEntry\_2" />

<Entry Placeholder="Пароль" x:Name="passwordEditEntry\_2" />

<Label Text="Сотрудник приемной комиссии:" HorizontalTextAlignment="Start" Margin="3,0,0,0"/>

<Switch x:Name="isAdmissionEditSwitch" HorizontalOptions="Center" Margin="50,-27,0,0"/>

<Button Text="Сохранить" Clicked="SaveTeacher\_Clicked" StyleId="button"/>

</StackLayout>

</ContentPage>

<ContentPage>

<StackLayout>

<Label Text="Удаление студентов" HorizontalTextAlignment="Center" StyleClass="label-bold"/>

<StackLayout>

<Label Text="Фильтр:" Margin="3,0,0,0"/>

<Entry x:Name="deleteFilterEntry" TextChanged="DeleteFilterEntry\_TextChanged" />

</StackLayout>

<ListView x:Name="deleteStudentsListView" ItemSelected="DeleteStudentsListView\_ItemSelected">

<ListView.ItemTemplate>

<DataTemplate>

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

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

<Button Text="Удалить" Clicked="DeleteStudent\_Clicked" StyleId="button"/>

</StackLayout>

</ContentPage>

</CarouselPage>

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

namespace CollegeApp

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class ComissionPage : CarouselPage

{

private readonly CollegeDatabase \_database;

private List<Student> \_students;

private List<Teacher> \_teachers;

private Student \_selectedStudent;

private Teacher \_selectedTeacher;

public ComissionPage()

{

InitializeComponent();

\_database = new CollegeDatabase(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "Database.db3"));

LoadStudents();

LoadTeachers();

}

private async void LoadStudents()

{

\_students = await \_database.GetStudentsAsync();

studentsListView.ItemsSource = \_students;

deleteStudentsListView.ItemsSource = \_students;

var groups = await \_database.GetGroupsAsync();

groupPicker.ItemsSource = groups.Select(g => g.Name).ToList();

}

private async void LoadTeachers()

{

\_teachers = await \_database.GetTeachersAsync();

teachersListView.ItemsSource = \_teachers;

}

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

{

if (string.IsNullOrEmpty(fullNameEntry.Text) || string.IsNullOrEmpty(courseEntry.Text) || string.IsNullOrEmpty(loginEntry.Text) || string.IsNullOrEmpty(passwordEntry.Text))

{

\_ = DisplayAlert("Внимание пользователь!", "Заполните все поля для ввода.", "OK");

return;

}

else if (groupPicker.SelectedIndex == -1)

{

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

return;

}

else if (int.Parse(courseEntry.Text) <= 0 || int.Parse(courseEntry.Text) > 4)

{

\_ = DisplayAlert("Внимание пользователь!", "Введите верное значение курса.", "OK");

return;

}

var selectedGroupName = groupPicker.SelectedItem as string;

var selectedGroup = await \_database.GetGroupByNameAsync(selectedGroupName);

var student = new Student

{

FullName = fullNameEntry.Text,

DateOfBirth = dateOfBirthPicker.Date,

IsBudget = isBudgetSwitch.IsToggled,

Course = int.Parse(courseEntry.Text),

GroupId = selectedGroup.GroupId,

Photo = null,

Login = loginEntry.Text,

Password = passwordEntry.Text

};

await \_database.SaveStudentAsync(student);

fullNameEntry.Text = string.Empty;

dateOfBirthPicker.Date = DateTime.Now;

isBudgetSwitch.IsToggled = false;

courseEntry.Text = string.Empty;

loginEntry.Text = string.Empty;

passwordEntry.Text = string.Empty;

LoadStudents();

}

private void StudentsListView\_ItemSelected(object sender, SelectedItemChangedEventArgs e)

{

\_selectedStudent = e.SelectedItem as Student;

fullNameEditEntry.Text = \_selectedStudent?.FullName;

dateOfBirthEditPicker.Date = \_selectedStudent?.DateOfBirth ?? DateTime.Now;

isBudgetEditSwitch.IsToggled = \_selectedStudent?.IsBudget ?? false;

courseEditEntry.Text = \_selectedStudent?.Course.ToString();

groupIdEditEntry.Text = \_selectedStudent?.GroupId.ToString();

loginEditEntry.Text = \_selectedStudent?.Login;

passwordEditEntry.Text = \_selectedStudent?.Password;

}

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

{

if (\_selectedStudent != null)

{

\_selectedStudent.FullName = fullNameEditEntry.Text;

\_selectedStudent.DateOfBirth = dateOfBirthEditPicker.Date;

\_selectedStudent.IsBudget = isBudgetEditSwitch.IsToggled;

\_selectedStudent.Course = int.Parse(courseEditEntry.Text);

\_selectedStudent.GroupId = int.Parse(groupIdEditEntry.Text);

\_selectedStudent.Login = loginEditEntry.Text;

\_selectedStudent.Password = passwordEditEntry.Text;

await \_database.SaveStudentAsync(\_selectedStudent);

fullNameEditEntry.Text = string.Empty;

dateOfBirthEditPicker.Date = DateTime.Now;

isBudgetEditSwitch.IsToggled = false;

courseEditEntry.Text = string.Empty;

groupIdEditEntry.Text = string.Empty;

loginEditEntry.Text = string.Empty;

passwordEditEntry.Text = string.Empty;

LoadStudents();

}

}

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

{

if (\_selectedStudent != null)

{

await \_database.DeleteStudentAsync(\_selectedStudent);

fullNameEditEntry.Text = string.Empty;

dateOfBirthEditPicker.Date = DateTime.Now;

isBudgetEditSwitch.IsToggled = false;

courseEditEntry.Text = string.Empty;

groupIdEditEntry.Text = string.Empty;

loginEditEntry.Text = string.Empty;

passwordEditEntry.Text = string.Empty;

LoadStudents();

}

}

private void DeleteStudentsListView\_ItemSelected(object sender, SelectedItemChangedEventArgs e)

{

\_selectedStudent = e.SelectedItem as Student;

}

private void EditFilterEntry\_TextChanged(object sender, TextChangedEventArgs e)

{

var filter = editFilterEntry.Text.ToLowerInvariant();

var filteredStudents = \_students.Where(s => s.FullName.ToLowerInvariant().Contains(filter)).ToList();

studentsListView.ItemsSource = filteredStudents;

}

private void DeleteFilterEntry\_TextChanged(object sender, TextChangedEventArgs e)

{

var filter = deleteFilterEntry.Text.ToLowerInvariant();

var filteredStudents = \_students.Where(s => s.FullName.ToLowerInvariant().Contains(filter)).ToList();

deleteStudentsListView.ItemsSource = filteredStudents;

}

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

{

if (\_selectedTeacher != null)

{

\_selectedTeacher.FullName = fullNameEditTeacherEntry\_2.Text;

\_selectedTeacher.Workload = int.Parse(workloadEditEntry.Text);

\_selectedTeacher.Login = loginEditEntry\_2.Text;

\_selectedTeacher.Password = passwordEditEntry\_2.Text;

\_selectedTeacher.IsAdmission = isAdmissionEditSwitch.IsToggled;

await \_database.SaveTeacherAsync(\_selectedTeacher);

fullNameEditTeacherEntry\_2.Text = string.Empty;

workloadEditEntry.Text = string.Empty;

loginEditEntry\_2.Text = string.Empty;

passwordEditEntry\_2.Text = string.Empty;

isAdmissionEditSwitch.IsToggled = false;

LoadTeachers();

}

}

private void TeachersListView\_ItemSelected(object sender, SelectedItemChangedEventArgs e)

{

\_selectedTeacher = e.SelectedItem as Teacher;

fullNameEditTeacherEntry\_2.Text = \_selectedTeacher?.FullName;

workloadEditEntry.Text = \_selectedTeacher?.Workload.ToString();

loginEditEntry\_2.Text = \_selectedTeacher?.Login;

passwordEditEntry\_2.Text = \_selectedTeacher?.Password;

isAdmissionEditSwitch.IsToggled = \_selectedTeacher?.IsAdmission ?? false;

}

private void EditTeacherFilterEntry\_TextChanged(object sender, TextChangedEventArgs e)

{

var filter = editTeacherFilterEntry.Text.ToLowerInvariant();

var filteredTeachers = \_teachers.Where(t => t.FullName.ToLowerInvariant().Contains(filter)).ToList();

teachersListView.ItemsSource = filteredTeachers;

}

}

}

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">

<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:

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), "Database.db3"));

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.PushModalAsync(new StudentPage(selectedStudent));

((ListView)sender).SelectedItem = null;

}

}

}

1. База данных:

public class TeacherSpecialty

{

[PrimaryKey, AutoIncrement]

public int TeacherSpecialtyId { get; set; }

public int TeacherId { get; set; }

public int SpecialtyId { get; set; }

}

public class Teacher

{

[PrimaryKey, AutoIncrement]

public int TeacherId { get; set; }

public string FullName { get; set; }

public int Workload { get; set; }

public int SpecialtyId { get; set; }

public string Login { get; set; }

public string Password { get; set; }

public bool IsAdmission { get; set; }

}

public class StudentSpecialty

{

[PrimaryKey, AutoIncrement]

public int StudentSpecialtyId { get; set; }

public int StudentId { get; set; }

public int SpecialtyId { get; set; }

}

public class Student

{

[PrimaryKey, AutoIncrement]

public int StudentId { get; set; }

public string FullName { get; set; }

public DateTime DateOfBirth { get; set; }

public bool IsBudget { get; set; }

public int Course { get; set; }

public int GroupId { get; set; }

public string Photo { get; set; }

public string Login { get; set; }

public string Password { get; set; }

}

public class Specialty

{

[PrimaryKey, AutoIncrement]

public int SpecialtyId { get; set; }

public string Name { get; set; }

}

public class Group

{

[PrimaryKey, AutoIncrement]

public int GroupId { get; set; }

public string Name { get; set; }

public int SpecialtyId { get; set; }

}

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

namespace CollegeApp

{

public class CollegeDatabase

{

readonly SQLiteAsyncConnection \_database;

public CollegeDatabase(string dbPath)

{

\_database = new SQLiteAsyncConnection(dbPath);

\_database.CreateTableAsync<Student>().Wait();

\_database.CreateTableAsync<Group>().Wait();

\_database.CreateTableAsync<Specialty>().Wait();

\_database.CreateTableAsync<Teacher>().Wait();

\_database.CreateTableAsync<StudentSpecialty>().Wait();

\_database.CreateTableAsync<TeacherSpecialty>().Wait();

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

{

var specialtyes = new List<Specialty>

{

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

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

};

\_database.InsertAllAsync(specialtyes).Wait();

var groups = new List<Group>

{

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

new Group { Name = "ХТ-11", SpecialtyId = 1 },

};

\_database.InsertAllAsync(groups).Wait();

var teachers = new List<Teacher>

{

new Teacher { FullName = "Сидоров Сидор Сидорович", Workload = 100, Login = "Sidorov", IsAdmission = false, Password = "123456" },

new Teacher { FullName = "Козлова Анна Ивановна", Workload = 200, Login = "Kozlova", IsAdmission = true, Password = "654321" },

};

\_database.InsertAllAsync(teachers).Wait();

var students = new List<Student>

{

new Student { FullName = "Гордейчик Степан Алексеевич", DateOfBirth = new DateTime(2005, 3, 31), IsBudget = true, Course = 2, GroupId = 1, Login = "Stepan", Password = "123456789", Photo = null },

new Student { FullName = "Иванов Иван Иванович", DateOfBirth = new DateTime(2004, 12, 1), IsBudget = true, Course = 1, GroupId = 2, Login = "Ivan", Password = "4321", Photo = "Ivan.JPG" },

};

\_database.InsertAllAsync(students).Wait();

var studentSpecialties = new List<StudentSpecialty>

{

new StudentSpecialty { StudentId = 1, SpecialtyId = 1 },

new StudentSpecialty { StudentId = 2, SpecialtyId = 1 },

new StudentSpecialty { StudentId = 2, SpecialtyId = 2 },

};

\_database.InsertAllAsync(studentSpecialties).Wait();

var teacherSpecialties = new List<TeacherSpecialty>

{

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

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

};

\_database.InsertAllAsync(teacherSpecialties).Wait();

}

}

// методы для поиска и группировки данных

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

{

return await \_database.Table<Student>()

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

.FirstOrDefaultAsync();

}

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

{

return await \_database.Table<Teacher>()

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

.FirstOrDefaultAsync();

}

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

{

return \_database.QueryAsync<Specialty>(

"SELECT Specialty.\* FROM Specialty " +

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

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

}

public Task<string> GetGroupNameAsync(int groupId)

{

return \_database.ExecuteScalarAsync<string>(

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

}

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

{

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

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

}

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

{

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

if (group != null)

{

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

return students;

}

return new List<Student>();

}

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

{

return await \_database.Table<Student>().ToListAsync();

}

public async Task SaveStudentAsync(Student student)

{

if (student.StudentId != 0)

{

await \_database.UpdateAsync(student);

}

else

{

await \_database.InsertAsync(student);

}

}

public async Task DeleteStudentAsync(Student student)

{

if (student != null)

{

await \_database.DeleteAsync(student);

}

}

public Task<int> SaveTeacherAsync(Teacher teacher)

{

if (teacher.TeacherId != 0)

{

return \_database.UpdateAsync(teacher);

}

else

{

return \_database.InsertAsync(teacher);

}

}

public Task<List<Teacher>> GetTeachersAsync()

{

return \_database.Table<Teacher>().ToListAsync();

}

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

{

return await \_database.Table<Group>().ToListAsync();

}

public async Task<Group> GetGroupByNameAsync(string groupName)

{

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

}

}

}

Это основные файлы проекта, которые определены в коде.

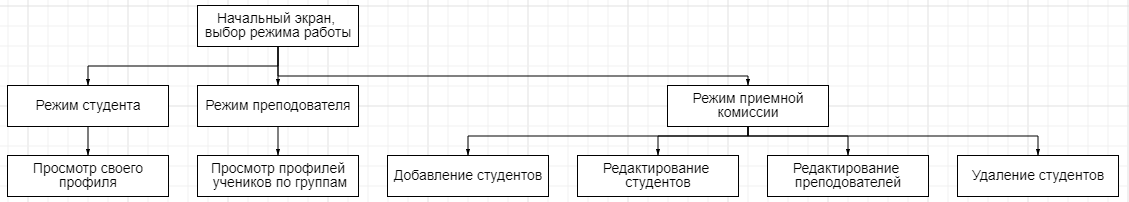
## Описание разработанных функций

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.
  + Изображение студента (StudentPhoto), отображающее фотографию студента.
  + 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:

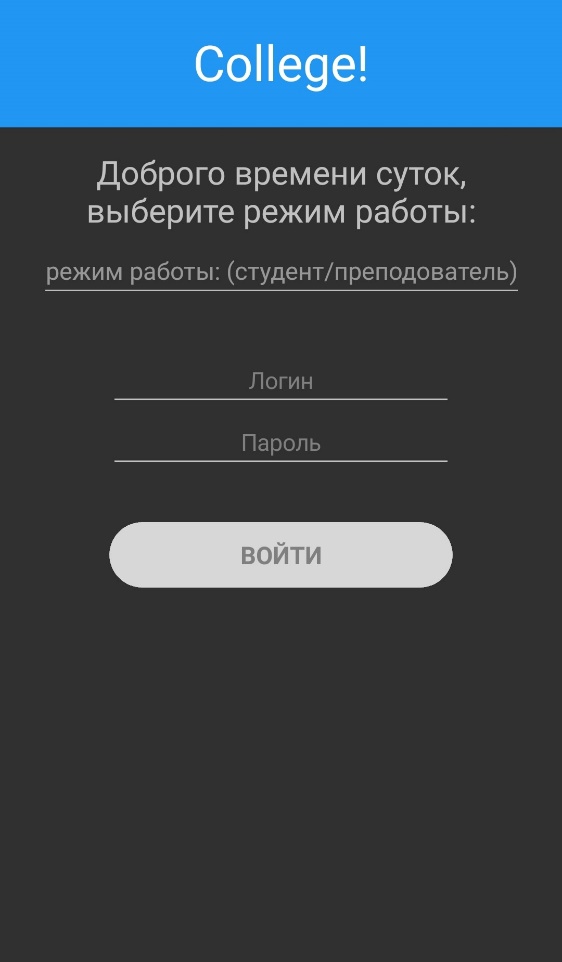
* Элементы на странице:
  + Label с именем студента (FullName).
  + Изображение студента (StudentPhoto).
  + Label с текстом "Дата рождения:" и соответствующим значением даты рождения студента (DateOfBirth).
  + Label с текстом "Форма обучения:" и соответствующим значением формы обучения студента (IsBudget).
  + 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:

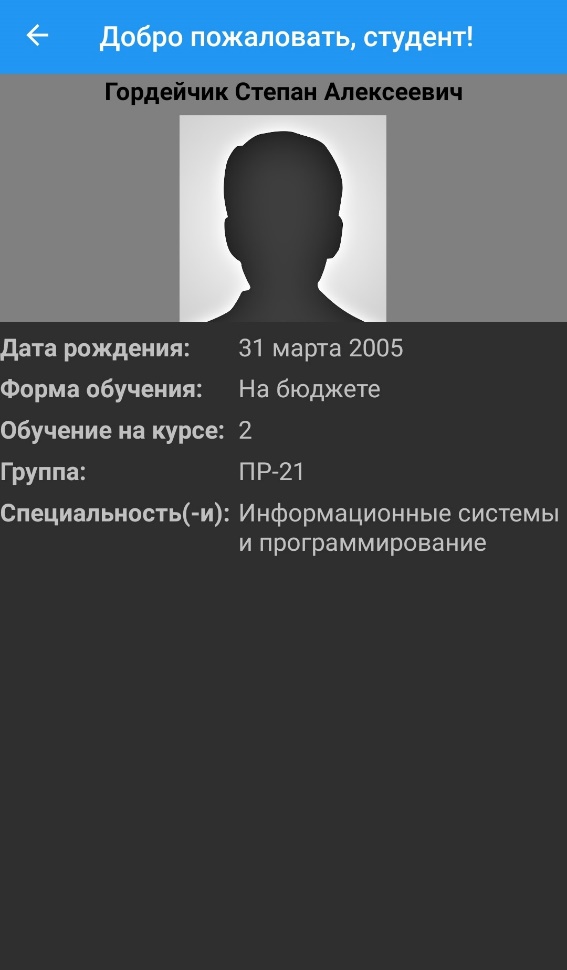
* Элементы на странице:
  + Entry для редактирования полного имени студента (fullNameEditEntry).
  + DatePicker для редактирования даты рождения студента (dateOfBirthEditPicker).
  + Switch для редактирования формы обучения студента (isBudgetEditSwitch).
  + Entry для редактирования курса студента (courseEditEntry).
  + 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)

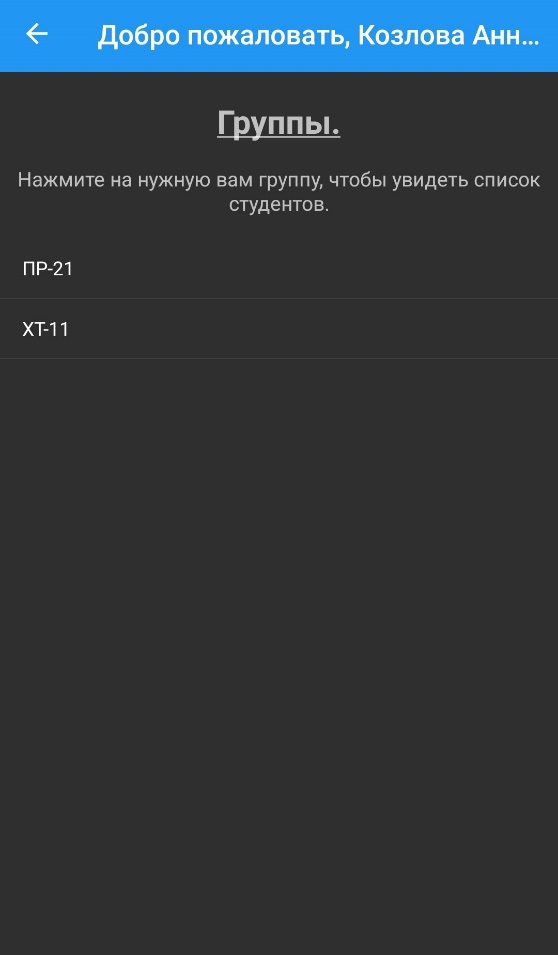
1. Главная страница



1. Режим студента:



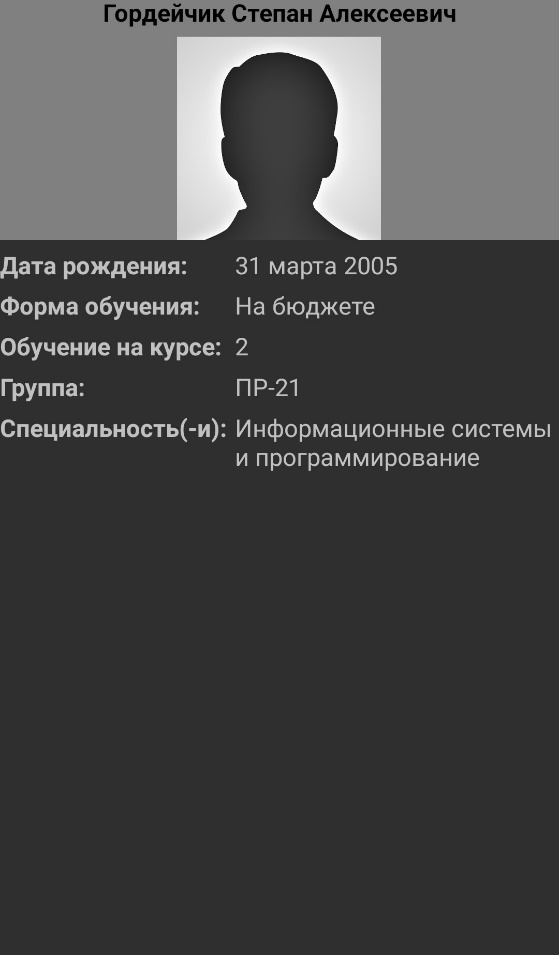
1. Режим преподавателя (выбор группы):



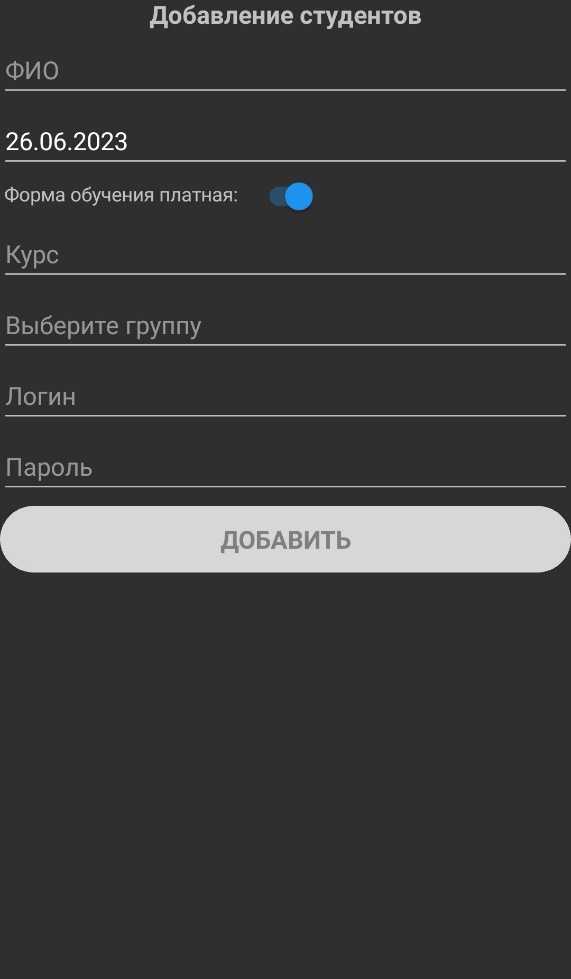
1. Режим преподавателя (выбор студента в группе):



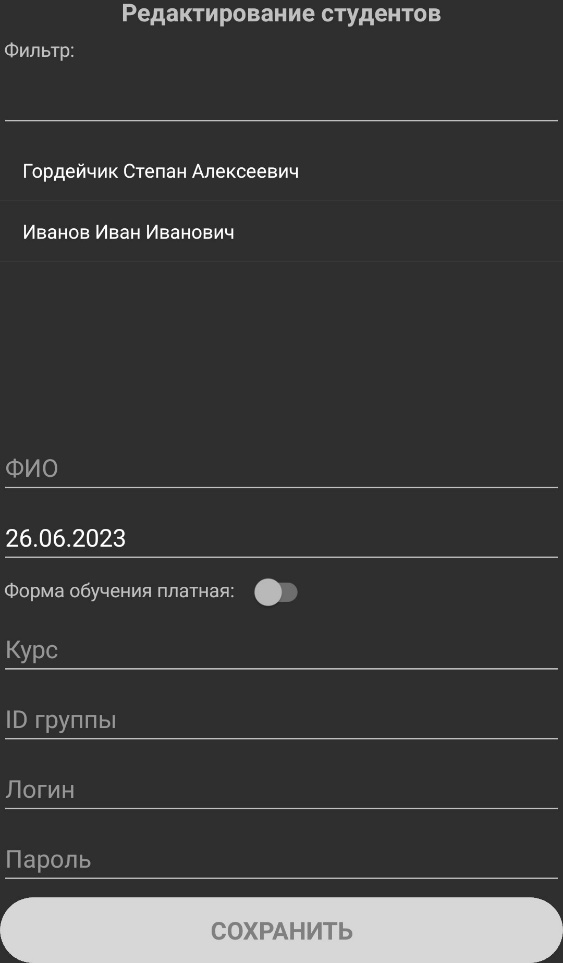
1. Режим преподавателя (просмотр профиля студента):



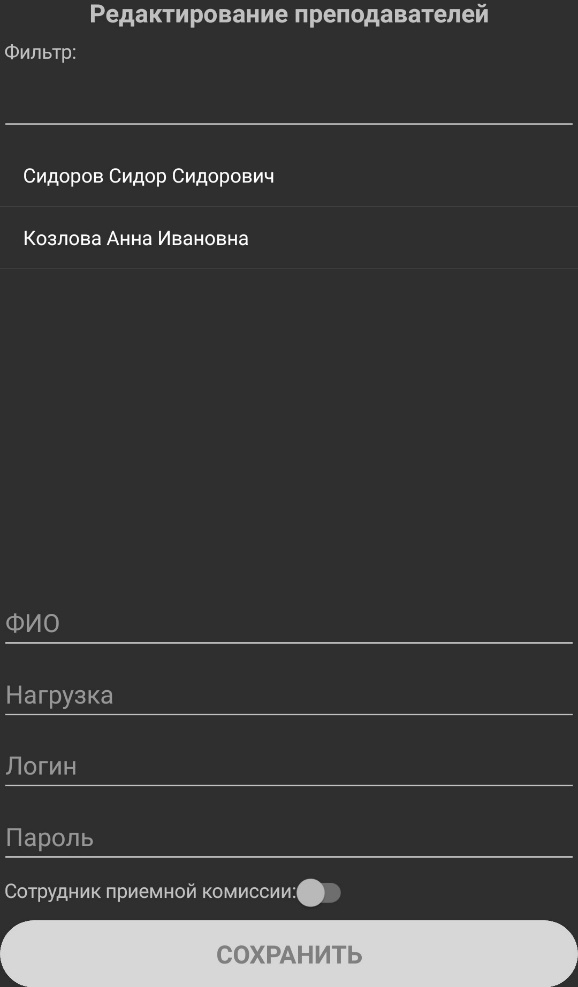
1. Режим приемной комиссии (добавление студентов):



1. Режим приемной комиссии (редактирование студентов):



1. Режим приемной комиссии (редактирование преподавателей):



1. Режим приемной комиссии (удаление студентов):

