МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ижевский государственный технический университет имени М. Т. Калашникова»

(ФГБОУ «ИжГТУ имени М. Т. Калашникова»)

Кафедра «Программное обеспечение»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к выпускной квалификационной работе бакалавра на тему:

«Разработка мобильного приложения

«Личный кабинет преподавателя»

|  |  |
| --- | --- |
| Дипломник:  Студент группы Б08-191-2 | М.Д. Ложкин |
| Руководитель: | Р.О. Султанов |
| Нормоконтролер: | В.П. Соболева |
| Зав. кафедрой ПО: | А.В. Коробейников |

Ижевск 2020

РЕФЕРАТ

Пояснительная записка к выпускной квалификационной работе на тему «Мобильное приложение «Личный кабинет преподавателя» оформлена на 87 листах, содержит 40 рисунков, 4 таблицы и 9 использованных источников.

Целью работы является разработка мобильного приложения для использования личного кабинета сайта ИжГТУ.

Программный продукт разрабатывается с целью получения удобного доступа к личному кабинету сайта ИжГТУ с мобильного устройства без использования браузера. Приложение позволяет просматривать и редактировать информацию аналогично функционалу личного кабинета на сайте ИжГТУ.

Работа приложения основана на запросах к API сайта ИжГТУ, соответственно оно не представляет угрозы серверу и личным данным пользователей. Особенностью приложения является разделение функционала личного кабинета преподавателя и студента в одном месте.

В реализации приложения используется язык программирования C# с дополнением для разработки мобильных приложений Xamarin Forms. Разработка выполняется в среде Microsoft Visual Studio.

В результате разработки получено мобильное приложение предоставляющие планированные возможности.

СОДЕРЖАНИЕ

[Введение 6](#_Toc43824449)

[1. Анализ разработки приложения «Личный кабинет преподавателя» 8](#_Toc43824450)

[1.1. Обоснование целесообразности разработки системы 8](#_Toc43824451)

[1.1.1. Назначение системы 8](#_Toc43824452)

[1.1.2. Обоснование цели системы 8](#_Toc43824453)

[1.1.3. Обоснование состава автоматизируемых задач 9](#_Toc43824454)

[1.2. Аналитический обзор 9](#_Toc43824455)

[1.3. Основные требования к системе 11](#_Toc43824456)

[1.3.1. Состав автоматизируемых задач 11](#_Toc43824457)

[1.3.2. Функциональное назначение системы 11](#_Toc43824458)

[1.3.3. Особенности системы, условия эксплуатации, основные определяющие требования к системе 12](#_Toc43824459)

[1.3.4. Требования к функциональной структуре системы 12](#_Toc43824460)

[1.3.5. Типовые проектные решения и пакеты прикладных программ, применяемые в системе 16](#_Toc43824461)

[1.3.6. Требования к техническому обеспечению 16](#_Toc43824462)

[1.3.7. Требования к программному обеспечению 16](#_Toc43824463)

[1.3.8. Перспективность системы, возможность её развития 16](#_Toc43824464)

[1.4. Основные технические решения проекта системы 16](#_Toc43824465)

[1.4.2. Безопасность разрабатываемой системы 17](#_Toc43824466)

[1.4.3. Описание системы программного обеспечения 18](#_Toc43824467)

[2. Разработка мобильного приложения «Личный кабинет преподавателя» 19](#_Toc43824468)

[2.1. Описание постановки задачи 19](#_Toc43824469)

[2.1.1. Характеристика задачи 19](#_Toc43824470)

[2.1.2. Входная информация 19](#_Toc43824471)

[2.1.3. Выходная информация 19](#_Toc43824472)

[2.2. Разработка функционала приложения для взаимодействия с сервером 20](#_Toc43824473)

[2.2.1. Назначение и характеристика 20](#_Toc43824474)

[2.2.2. Входная информация 20](#_Toc43824475)

[2.2.3. Выходная информация 20](#_Toc43824476)

[2.2.4. Алгоритм решения 20](#_Toc43824477)

[2.2.5. Описание контрольного примера 23](#_Toc43824478)

[2.2.5.1. Назначение 23](#_Toc43824479)

[2.2.5.2. Исходные данные 23](#_Toc43824480)

[2.2.5.3. Результаты расчета 23](#_Toc43824481)

[2.2.5.4. Результаты испытания программы 24](#_Toc43824482)

[2.3 Разработка функционала преподавателя в мобильном приложении для просмотра информации и взаимодействия с ней 24](#_Toc43824483)

[2.3.1 Назначение и характеристика 24](#_Toc43824484)

[2.3.2. Входная информация 24](#_Toc43824485)

[2.3.3. Выходная информация 24](#_Toc43824486)

[2.3.4. Алгоритм решения 25](#_Toc43824487)

[2.3.5. Описание контрольного примера 27](#_Toc43824488)

[2.3.5.1. Назначение 27](#_Toc43824489)

[2.3.5.2. Исходные данные 27](#_Toc43824490)

[2.3.5.3. Результаты расчета 27](#_Toc43824491)

[2.3.5.4. Результаты испытания программы 31](#_Toc43824492)

[2.4. Разработка системы важных объявлений от преподавателя студентам 31](#_Toc43824493)

[2.4.1. Характеристика задачи 31](#_Toc43824494)

[2.4.2. Входная информация 31](#_Toc43824495)

[2.4.3. Выходная информация 31](#_Toc43824496)

[2.4.4. Алгоритм решения 31](#_Toc43824497)

[2.4.5. Описание контрольного примера 33](#_Toc43824498)

[2.4.5.1. Назначение 33](#_Toc43824499)

[2.4.5.2. Исходные данные 33](#_Toc43824500)

[2.4.5.3. Результаты расчета 33](#_Toc43824501)

[2.4.5.4. Результаты испытания программы 34](#_Toc43824502)

[Заключение 35](#_Toc43824503)

[Список использованных источников 36](#_Toc43824504)

Приложение 1. [Текст программы 37](#_Toc43824505)

Приложение 2. [Результаты работы программы 78](#_Toc43824509)

Приложение 3. [Руководство пользователя приложения «Личный кабинет преподавателя» 82](#_Toc43824510)

# Введение

В настоящее время популярность рынка мобильных устройств быстро растет. Это обусловлено тем, что люди могут использовать их везде. Соответственно, растет и рынок приложений для этих устройств. Выпущено много приложений, дублирующих функционал крупных сервисов в удобном интерфейсе, которые не требуют от пользователя использования браузера. В связи с этим мобильные версии такие приложения завоевывают популярность на рынке.

Также тема использования мобильных устройств актуальна и для преподавателей и студентов, которые часто перемещаются между аудиториями или корпусами без возможности использовать компьютер для использования личного кабинета.

Некоторые университеты также имеют собственные приложения для доступа к личному кабинету, например, такие как УдГУ, СВФУ, УрФУ, КФУ и другие. На данный момент ИжГТУ не имеет аналогичного приложения, доступ к личному кабинету с мобильного устройства осуществляется лишь с использованием браузера. Данный способ использования личного кабинета не всегда удобен, это сильно заметно для пользователей бюджетных устройств. Мобильное приложение не требует загрузки целой страницы и взаимодействие с интерфейсом происходит быстрее.

Большинство преподавателей имеют мобильное устройство с доступом в интернет и во время передвижения между аудиториями или иногда даже корпусами не имеют возможности использовать компьютер для использования личного кабинета. В этой ситуации пользователь может использовать мобильное устройство, но использование браузера удобно не в любой ситуации, поскольку это занимает больше времени и

Приложение, разрабатываемое в работе призвано для предоставления быстрого и удобного доступа к личному кабинету преподавателя с мобильного устройства без использования браузера. Работа приложения основана на API запросах к сайту ИжГТУ с целью исключения угроз для данных пользователей и БД.

# 1. Анализ разработки приложения «Личный кабинет преподавателя»

## 1.1. Обоснование целесообразности разработки системы

### 1.1.1. Назначение системы

Система предназначена для предоставления доступа преподавателю к наиболее используемой информации с сайта ИжГТУ. Например, редактирование контактных данных или просмотра и редактирования расписания консультаций.

### 1.1.2. Обоснование цели системы

Рынок мобильных устройств растет в последнее время очень быстро. Соответственно, растет рынок приложений для мобильных устройств и средств для их разработки.

Часто для веб-сервисов разрабатываются мобильные приложения для быстрого доступа с мобильного устройства, так как пользователь не всегда имеет доступ к персональному компьютеру. Также не всегда удобно использовать браузер мобильного устройства для таких сервисов, так как мобильное приложение работает быстрее и не требует загрузки целой страницы. В магазинах приложений для мобильных устройств первые места занимают приложения таких же веб сервисов, например, топ бесплатных приложений в Google Play Market: Zoom, Вконтакте, личные кабинеты мобильных операторов и т.д.

Преподаватель не всегда может иметь доступ к компьютеру для использования личного кабинета ИжГТУ, соответственно, нужно предоставить ему удобный доступ в приложении с мобильного устройства, которое будет выполнять некоторые основные функции, которые имеются в веб-версии личного кабинета.

### 1.1.3. Обоснование состава автоматизируемых задач

Главное назначение приложения – предоставление быстрого и удобного доступа для преподавателя к информации личного кабинета ИжГТУ связанной со взаимодействием со студентами.

## 1.2. Аналитический обзор

Многие преподаватели не всегда имеют возможность использовать полную версию личного кабинета. Например, когда от преподавателя требуется мобильность, он часто перемещается между аудиториями или корпусами и компьютера или ноутбука просто нет под рукой. Часто удобнее использовать мобильное устройство для доступа к личному кабинету. Использование мобильного браузера также удобно не для всех. В такой момент нужно приложение с более узким применением избавляющее пользователя от использования браузера. Мобильное приложение позволяет преподавателю получить быстрый и удобный доступ к основным функциям личного кабинета в любое время и в любом месте. Преимущества использования приложения перед использованием браузера в том, что оно требует только получения ответов от сервера на посылаемые запросы вместо загрузки целых страниц с разметкой, стилями и скриптами. Такая разница будет заметна больше всего на бюджетных устройствах.

Главная особенность приложения – функционал личного кабинета преподавателя и студента в одном месте. Достаточно ввести свои данные для авторизации.

Другие университеты имеют аналогичные приложения для использования личного кабинета, но они несколько отличаются от разрабатываемого приложения, например, в них нет разделения на студента и сотрудника в одном приложении. Сравнение с другими приложениями приведено в таблице 1.1.

Таблица 1.1

Сравнение аналогов приложения в других ВУЗах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Приложения | Функционал преподавателя | | | Функционал студента |
| Личная информация | Редактирование расписания консультаций | Система объявлений для студентов |
| ISTU Mobile | + | + | + | + |
| Istudent (УрФУ) | - | - | - | + |
| vBSU (БСУ) | - | - | - | + |
| Личный кабинет студента БГМУ | - | - | - | + |
| Личный кабинет студента СВФУ | - | - | - | + |

Как видно из таблицы сравнения, аналогичные приложения имеют функционал личного кабинета студента, но не имеют функционала для преподавателя. Это главная особенность приложения, для реализации которой был разработан механизм авторизации и разделения ролей. Также все приложения разработаны конкретно для определенных университетов, а это значит, что ни одно приложение не может заменить разрабатываемую в работе систему.

Для разработки мобильных и кроссплатформенных приложений существует множество фреймворков, в работе был выбран Xamarin Forms, так как он основан на языке программирования C# и прост в освоении. Также Xamarin имеет большое сообщество разработчиков и удобную документацию от Microsoft. Соответственно, будет легко найти людей, которые смогут заниматься сопровождением и развитием программного продукта. Помимо этого, Xamarin используется в удобной среде, где приложение отлаживается непосредственно в эмуляторе. А также среду Visual Studio позволяет устанавливать в проект дополнительные пакеты и расширения, которые упрощают процесс разработки.

## 1.3. Основные требования к системе

### 1.3.1. Состав автоматизируемых задач

Основной задачей разработки является быстрое предоставление доступа к наиболее запрашиваемой информации из личного кабинета с использованием мобильного устройства.

Приложение должно отвечать следующим требованиям:

* приложение должно предоставлять доступ как для студента, так и для преподавателя, а также иметь раздельный функционал;
* доступ к информации личного кабинета должен быть осуществлен посредством API сайта ИжГТУ, используя полученные от разработчика сайта запросы во избежание угроз для безопасности сайта и баз данных;
* архитектура должна позволять другим разработчикам знакомым с инструментами разработки легко производить доработку системы и добавление новых модулей;
* данные должны отображаться в удобной форме и без лишних недоступных пользователю элементов.

### 1.3.2. Функциональное назначение системы

Автоматизация системы должна отвечать следующим пунктам:

* вывод личной информации о пользователе (ФИО, занимаемые должности, информация об обучении если она имеется, контактные данные);
* редактирование контактных данных при необходимости;
* возможность коммуникации со студентами.

Просмотр расписания консультаций, а также их добавление и удаление.

### 1.3.3. Особенности системы, условия эксплуатации, основные определяющие требования к системе

Система предназначена для эксплуатации на мобильном устройстве пользователя (планшетный ПК/смартфон). Приложение не требует наличия больших вычислительных ресурсов и может работать на бюджетных устройствах. Определяющих требований как таковых система не имеет, кроме ограниченности поддерживаемых операционных систем.

### 1.3.4. Требования к функциональной структуре системы

Система призвана решить следующие задачи:

* взаимодействие с личной информацией пользователя:
  1. входные данные: логин и пароль для авторизации;
  2. выходные данные: информация о пользователе;
* редактирование контактных данных:
  1. входные данные: параметры для авторизации, обновленные контактные данные;
  2. выходные данные: сохранение новых контактных данных;
* просмотр и редактирование расписания консультаций:
  1. входные данные: параметры для авторизации;
  2. выходные данные: список консультаций;
* добавление и редактирование консультаций:
  1. входные данные: параметры авторизации, обновленный или заполненный экземпляр модели данных консультации;
  2. выходные данные: добавление новой или обновление существующей консультации;
* чат со студентами:
  1. входные данные: параметры для авторизации, ввод новых сообщений;
  2. выходные данные: получение списка сообщений для чата и отправка запросов для добавления новых;
* объявления для студентов:
  1. входные данные: параметры авторизации, выбор списка групп и ввод текста объявлений;
  2. выходные данные: получение списка существующих объявлений и добавление новых

Для доступа к главному меню приложения от пользователя требуются данные для авторизации. Дальнейшая схема программы показана на структурной схеме (рис. 1.1).

Структурная диаграмма программы

Рис. 1.1

Из структурной диаграммы видно, что модуль личного кабинета преподавателя отделен от личного кабинета студента, общими моментами являются только форма авторизации и меню настроек. Соответственно, логика приложения разделена по ролям пользователей, а общие элементы такие как, например, форма авторизации доступны всем пользователям.

Часть приложения, отвечающая за функционал студента, разрабатывалась отдельно и в работе затронута не будет. Часть приложения для преподавателя содержит в себе следующие формы:

* главное меню приложения,
* форма личной информации для просмотра и редактирования данных,
* форма просмотра списка консультаций,
* форма добавления консультаций,
* форма просмотра списка публикаций,
* форма просмотра списка групп студентов для выбора группы или добавления объявлений,
* форма добавления объявления,
* форма просмотра списка студентов группы для использования чата,
* форма для просмотра списка.

Помимо форм приложение содержит набор моделей данных и отдельный класс, выполняющий отправку запросов на сервер. Диаграмма зависимостей типов показана на рисунке 1.2.

Логику выполнения запросов и отправки данных на сервер выполняет статический класс GlobalMethods с использованием данных авторизации из класса GlobalVariables. Стартовой формой является AuthorizationPage, которая в свою очередь распределяет роли и открывает следующее окно главного меню студента или преподавателя в зависимости от роли пользователя. Главное меню для каждой роли имеет свой набор разделов. Формы для преподавателя сгруппированы по следующим разделам: личная информация, консультации, публикации, список студентов и объявления.

Диаграмма зависимостей типов

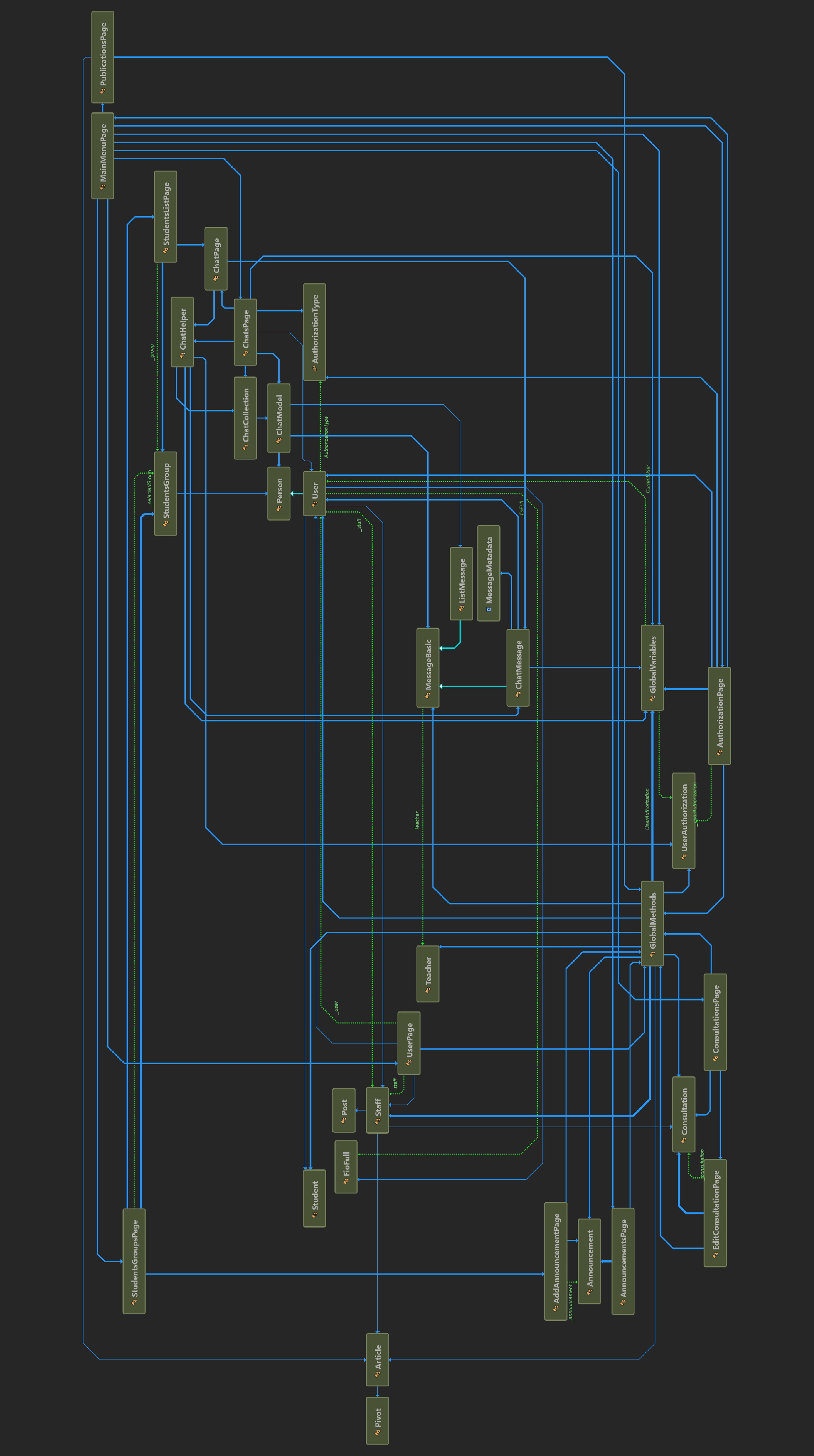


Рис. 1.2

## 1.3.5. Типовые проектные решения и пакеты прикладных программ, применяемые в системе

Для создания системы применяется среда разработки «Visual Studio 2017», а также инструмент для разработки мобильных приложений «Xamarin Forms», являющийся дополнительным загружаемым компонентом среды «Visual Studio». Для повышения удобства разработки использован «ReSharper», а для построения графиков, схем и ведения документации «Microsoft Office».

### 1.3.6. Требования к техническому обеспечению

Для использования приложения на устройстве должна быть учтены следующие требования:

* устройство должно использовать операционную систему Android 5 или iOS 6, а также возможно использование версий выше для обеих ОС,
* на устройстве должен быть доступ к интернету (Wi-Fi или мобильная сеть).

### 1.3.7. Требования к программному обеспечению

Приложение требует использования на устройстве операционной системы Android 6 или iOS 6, либо версий выше. Также в будущем в случае публикации приложения личного кабинета в магазинах приложений для ОС Android может потребоваться Google Play Market или аналог (Yandex.Store и др.), а для iOS AppStore.

### 1.3.8. Перспективность системы, возможность её развития

В будущем система может дополниться новыми функциями, такими как чат между студентом и преподавателем, сообщения от преподавателей для групп и другие существующие или новые функции веб-версии личного кабинета.

## 1.4. Основные технические решения проекта системы

1.4.1. Среда разработки системы

Существует множество фреймворков для разработки мобильных приложений на разных языках программирования, которые используют различные IDE. В таблице 1.2 приведено сравнение нескольких из самых популярных фреймворков разработки мобильных приложений

Таблица 1.2

Сравнение фреймворков разработки мобильных приложений

|  |  |  |  |
| --- | --- | --- | --- |
| Фреймворк | Язык | Среда разработки | Доступные платформы |
| Xamarin Forms | C# | Visual Studio | iOS, Android, UWP |
| React Native | JavaScript, Java, Objective-C, Swift | Visual Studio Code, Notepad++ | iOS, Android |
| Ionic | HTML, CSS, TypeScript, JavaScript | IonicPro | iOS, Android |
| Kotlin | Java, Kotlin | JetBrains Intellij Idea | Android |
|  | Swift | XCode | iOS |
|  | Objective-C | XCode | iOS |

Как видно из таблицы, Xamarin Forms является единственным фреймворком, который использует среду Visual Studio и использует язык C#. При выборе фреймворка эта особенность играла ключевую роль, поскольку студенты, принимающие участие в разработке приложения, имеют больший опыт работы с языком C#, чем с другими языками программирования. Среда разработки Visual Studio является одной из самых удобных, поскольку предоставляет разработчику множество возможностей, например, загрузку пакетов дополнений из открытого каталога NuGet. Также отладка мобильных приложений в среде Visual Studio проходит в уже запущенном эмуляторе мобильного устройства под управлением ОС Android/iOS. Сборка приложений также происходит в среде Visual Studio. Помимо перечисленных выше преимуществ Visual Studio имеет расширение для использования GitHub и Team Foundation Server, что упрощает совместную разработку приложения несколькими людьми. В процессе разработки приложения личного кабинета была использована система GitHub для упрощения процесса внесения совместных изменений

Помимо удобства среды разработки большим плюсом Xamarin является использование языка программирования C# так как он имеет удобную документацию на официальном сайте Microsoft, что помогает разработчику быстро найти нужную информацию о тех или иных компонентах самого языка и компонентов фреймворка. Также C# является одним из самых популярных языков программирования и на данный момент изучается студентами в ВУЗах, что упрощает поиск людей для дальшейшего сопровождения и развития приложения.

### 1.4.2. Безопасность разрабатываемой системы

Приложение не представляет угрозы сайту и базам данных которые он использует так как оно построено на взаимодействии API сервера где он развернут.

### 1.4.3. Описание системы программного обеспечения

Для функционирования системы на мобильном устройстве должна быть развернута операционная система Android 5 или iOS 6, а также версии выше. Также для использования приложения необходимо подключение к сети интернет посредством Wi-Fi или мобильной сети.

В процессе разработки системы использованы следующие компоненты:

* Microsoft Visual Studio 2017,
* набор инструментов Xamarin Forms для разработки мобильных приложений [1],
* JetBrains ReSharper,
* Microsoft Android Emulator.

# 2. Разработка мобильного приложения «Личный кабинет преподавателя»

## 2.1. Описание постановки задачи

### 2.1.1. Характеристика задачи

Задача – разработка мобильного приложения личного кабинета ИжГТУ для преподавателя.

Назначение приложения – предоставление удобного доступа к личному кабинету сайта ИжГТУ с мобильного устройства. Приложение совмещает в себе функционал личного кабинета преподавателя и студента и использует раздельный интерфейс.

Для разработки приложения поставлены следующие цели:

1. разработка функционала приложения, для взаимодействия с сервером;
2. разработка функционала преподавателя в приложении для отображения информации и взаимодействия с ней;
3. разработка системы важных объявлений от преподавателя студентам.

### 2.1.2. Входная информация

От пользователя при использовании приложения требуется следующая информация:

* логин (электронная почта) пользователя;
* пароль аккаунта;
* ввод информации для заполнения различных форм.

### 2.1.3. Выходная информация

Пользователь получает мобильное приложение, предоставляющее следующие функции:

* основная личная информация пользователя;
* просмотр, добавление, удаление, редактирование расписания консультаций;
* просмотр списка публикаций, в которых пользователь принимал участие;
* просмотр списка групп и студентов, которых обучает преподаватель;
* использование чата со студентами и отправка объявлений для них.

## 2.2. Разработка функционала приложения для взаимодействия с сервером

### 2.2.1. Назначение и характеристика

Задача – разработка системы, отправляющей запросы на сервер сайта ИжГТУ. Система предназначена для получения существующих данных с сервера и отправления новых данных на сервер.

Система использует HTTP запросы к API сервера, при разработке используется компонент .NET HttpClient.

### 2.2.2. Входная информация

Для отправки запросов на сервер используется информация, введенная пользователем и полученный с сервера токен авторизации. Каждый запрос имеет дополнительные параметры определяющие получаемые данные.

### 2.2.3. Выходная информация

В результате работы запросов система получает ответы от сервера для получения информации или отправляет новые записи на сервер для сохранения в БД. Все запросы выполняются с использованием JSON строк, которые сериализуются при отправке и десериализуются при получении. В системе данные представлены в виде моделей.

### 2.2.4. Алгоритм решения

Часть проекта, отвечающая за взаимодействием отделена в отдельное пространство имён DataHelpers, которое содержит два класса – GlobalMethods для выполнения логики и отправки запросов и GlobalVariables для хранения данных о текущей сессии (данные авторизации и пользователя). Работа всех запросов начинается с использования клиента для отправки запросов на сервер. Каждый запрос описанный в отдельном методе использует новый экземпляр класса HttpClient.

Класс GlobalMethods содержит следующие общие методы, предназначенные для всех пользователей:

1. UserAuthorisation – метод авторизации для получения токена;
2. GetUser – метод получения стандартных данных о пользователе;
3. GetMessages – для получения списка сообщений чата по пользователю;
4. GetAnnotations – метод получения списка объявлений от преподавателя, используется для просмотра как студентами, так и преподавателями;
5. GetConsultations – метод для получения списка консультаций, доступен так же для студентов (с применением Id преподавателя) и преподавателей;
6. GetStaff – метод получения информации, связанной с преподавателем, содержит такие данные, как информация об учетной степени, занимаемых должностях;
7. AddMessage – метод отправки нового сообщения.

Для студентов имеются отдельные методы:

1. GetSession – для получения данных о сессиях студента;
2. GetDebts – метод для получения информации об оплате (предназначен для студентов обучающихся платно);
3. GetTeachers – метод для получения списка преподавателей.

Для преподавателей существуют следующие методы:

1. GetArticles – метод получения списка публикаций пользователя;
2. UpdateStaff – метод для обновления личной информации пользователя, например, контактных данных;
3. AddOrUpdateConsultation – метод добавления новой консультации или обновления существующей;
4. AddAnnotation – метод добавления нового объявления для групп студентов.

Примеры схем работы алгоритмов запросов показаны на рисунках 2.1 и 2.2.

Пример вызова запроса получения данных



Рис. 2.1

Схема работы запроса отправки данных



Рис. 2.2

### 2.2.5. Описание контрольного примера

#### 2.2.5.1. Назначение

Назначение системы – упрощение выполнения HTTP запросов к API сервера для получения различных данных личного кабинета.

#### 2.2.5.2. Исходные данные

Для метода авторизации требуется вызвать метод с параметрами логина и пароля пользователя. Методы запроса информации требуют данных, полученных на этапе авторизации и в некоторых случаях требуются данные для уточнения запросов, например, номер группы студентов или ID пользователя для использования чата с ним. Для запросов добавления или обновления новой информации требуется использование метода с применением заполненных данных.

#### 2.2.5.3. Результаты расчета

В результате работы системы представлены примеры работы некоторых из запросов:

Сведения о пользователе

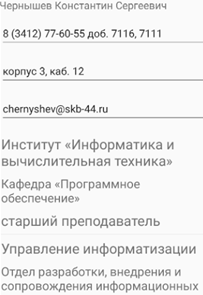


Рис. 2.3

Список публикаций

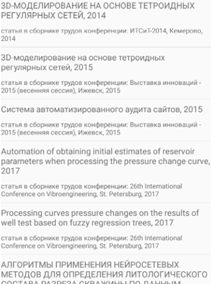


Рис. 2.4

Список консультаций

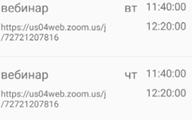


Рис. 2.5

#### 2.2.5.4. Результаты испытания программы

В результате работы программы получены ожидаемые ответы от сервера. Данные успешно сериализуются для использования в мобильном приложении и десериализуются для отправки на сервер.

## 2.3 Разработка функционала преподавателя в мобильном приложении для просмотра информации и взаимодействия с ней

### 2.3.1 Назначение и характеристика

Задача – разработка функционала приложения, который должен предоставлять пользователю (преподавателю) удобный доступ ко всем основным элементам и функциям личного кабинета сайта ИжГТУ. А также группировать информацию если она предоставляется в виде списков.

### 2.3.2. Входная информация

Авторизованный пользователь должен выбрать нужный пункт в меню нажав соответствующую кнопку. Помимо главного меню пользователь может производить следующие действия:

* добавить, редактировать или удалить консультации в соответствующем разделе;
* редактировать личные контактные данные;
* оставлять объявления для студентов в специальном разделе;
* отвечать студентам в чате,
* просматривать остальную информацию, такую как: список консультаций, отправленные объявления, личная информация, список публикаций.

Интерфейс личного кабинета должен иметь приятный вид и быть интуитивно понятным для пользователя. Диаграмма состояний при переходах между формами показана на рисунке 2.6.

Как видно из рисунка диаграммы состояний, пользователя встречает форма авторизации, где нужно ввести логин и пароль от личного кабинета, после чего пользователь переходит на страницу главного меню. Главное меню предоставляет пользователю 5 кнопок: личная информация, консультации, публикации, студенты и объявления. Каждая кнопка отвечает за открытие определенного раздела, который соответствует тексту на кнопке. В разделе с личной информацией пользователь видит основную информацию и контактные данные, которые может редактировать и сохранять. Раздел консультаций предназначен для просмотра списка консультаций преподавателя: время, место, комментарий. Также в разделе с консультациями имеется возможность удаления и редактирования существующих записей и добавления новых. За эти действия отвечают отдельные кнопки формы. В разделе публикаций пользователь может просматривать список публикаций, в которых он является одним из авторов. Раздел студентов предназначен для добавления объявлений и использования чата со студентами. При открытии раздела пользователь видит список групп, из которых может выбрать несколько для того, чтобы оставить для них текстовое объявление, также в данном разделе можно открыть список студентов определенной группы и перейти в раздел чата со студентом. Последним разделом является список объявлений, оставленных пользователем для студентов.

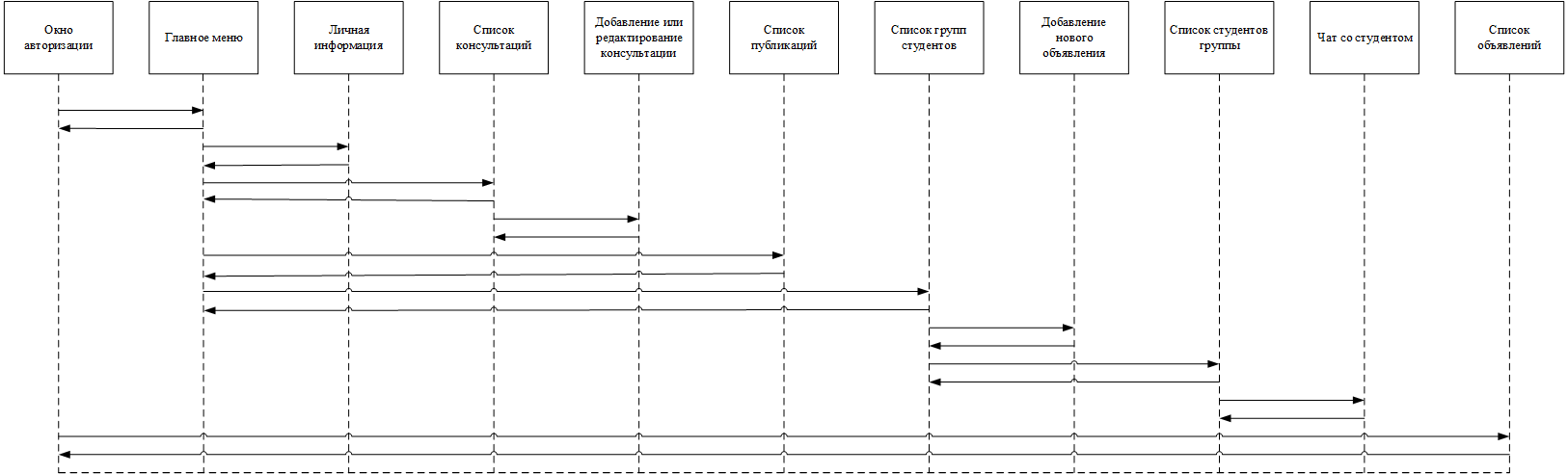
Диаграмма состояний приложения при переходах между формами

Рис. 2.6

### 2.3.3. Выходная информация

В ходе взаимодействия с интерфейсом программы пользователь получает следующие результаты:

* просмотр информации такой как личные данные, список публикаций, расписание консультаций, оставленные для студентов объявления, списки студентов, история сообщений в чате;
* редактирование расписания консультаций, личных контактных данных, добавление объявлений для студентов, отправка сообщений в чате.

### 2.3.4. Алгоритм решения

Работа приложения начинается с окна авторизации. в окне авторизации пользователь вводит данные – логин и пароль. После авторизации пользователь может взаимодействовать с главным меню. Меню содержит следующие пункты: личная информация, консультации и публикации.

Каждый из разделов в главном меню содержит информацию по соответствующему запросу. Далее представлены представления приложения и методы, которые они используют:

1. AuthorizationPage – представление авторизации, используется для ввода логина и пароля. Использует метод Authorize(email, password) для проверки корректности введенных данных и получения токена;
2. MainMenuPage – представление главного меню, используется для перехода в необходимый пользователю пункт, например, личные сведения, консультации, публикации и т.д. Вызывает некоторые формы представленные далее;
3. ConsultationsPage – представление списка консультаций, используется для вывода списка консультаций преподавателя. Использует метод GetConsultations() для получения списка консультаций;
4. UserPage – представление для просмотра личных сведений пользователя, таких как данные для связи, сведения об образовании, ученой степени, занимаемых должностях в ВУЗе и др. Использует метод GetUser() для получения модели пользователя, которая содержит отображаемую информацию;
5. PublucationsPage – представление для просмотра публикаций, при издании которых пользователь принимал участие. Использует метод GetPublications().

Ниже на рисунках 2.7-2.11 показаны схемы работы функционала преподавателя в приложении.

Использование раздела информации о пользователе



Рис. 2.7

Использование раздела публикаций

Рис. 2.8

Использование раздела консультаций



Рис. 2.9

Использование раздела взаимодействия со студентами



Рис. 2.10

Использование раздела объявлений



Рис. 2.11

### 2.3.5. Описание контрольного примера

#### 2.3.5.1. Назначение

Назначение системы – отображение запрашиваемой пользователем информации, а также её добавление, удаление и редактирование.

#### 2.3.5.2. Исходные данные

В результате работы программы будут указаны примеры скриншотов страниц приложения, выбранных пользователем с заполненными данными.

#### 2.3.5.3. Результаты расчета

На указанных ниже скриншотах (рис. 2.12-2.22) продемонстрирована работа приложения.

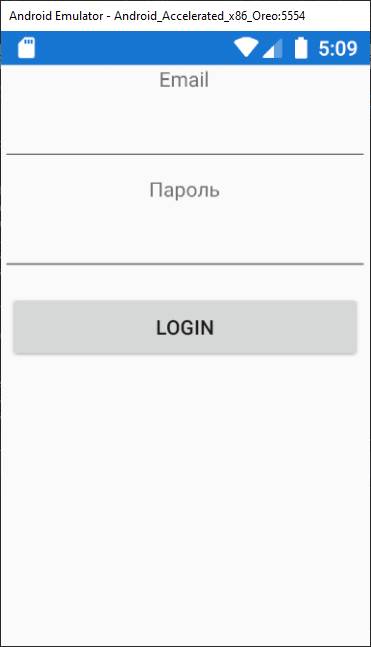
Меню авторизации

Рис. 2.12

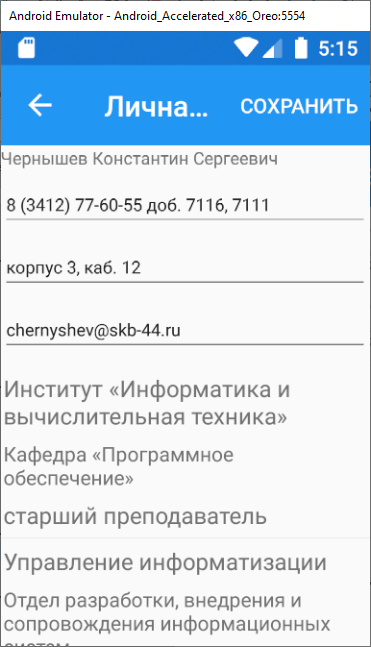
Меню личной информации о преподавателе

Рис. 2.14

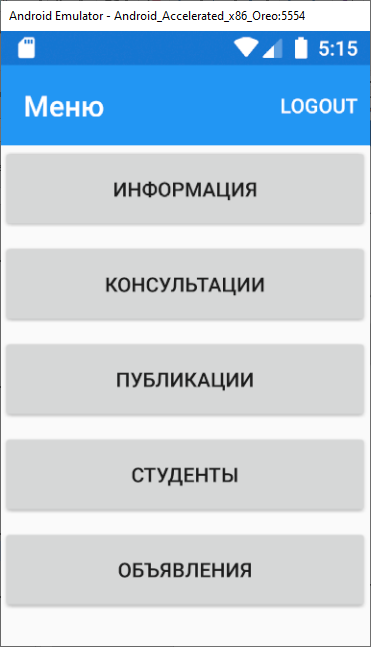
Главное меню преподавателя

Рис. 2.13

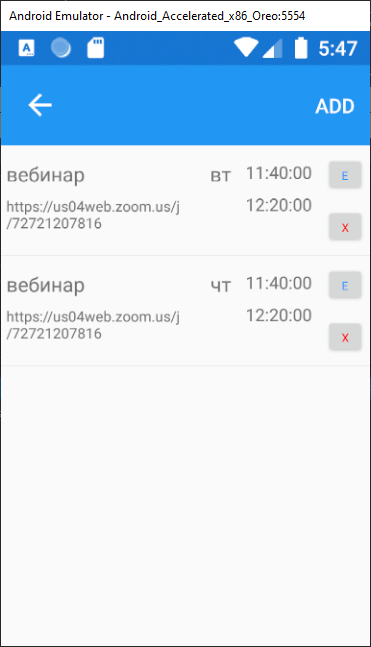
Список консультаций преподавателя

Рис. 2.15

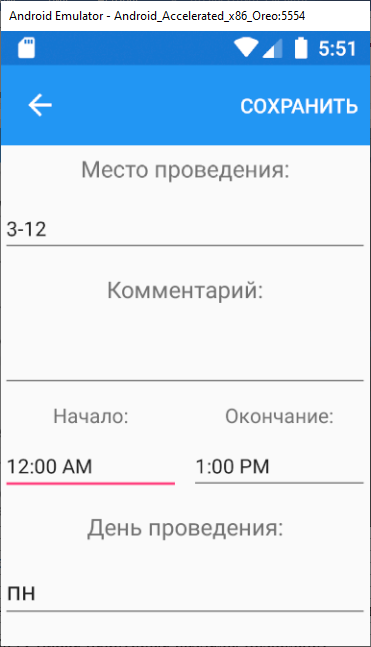
Форма добавления новой консультации

Рис. 2.16

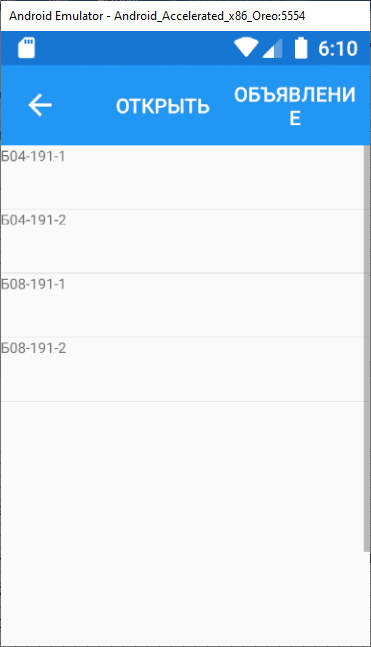
Список групп

Рис. 2.18

Список публикаций преподавателя

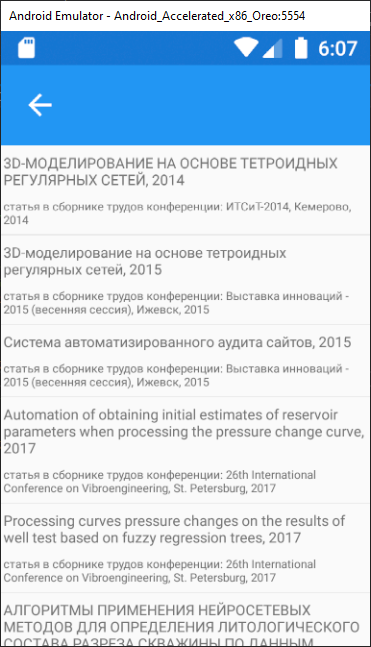


Рис. 2.17

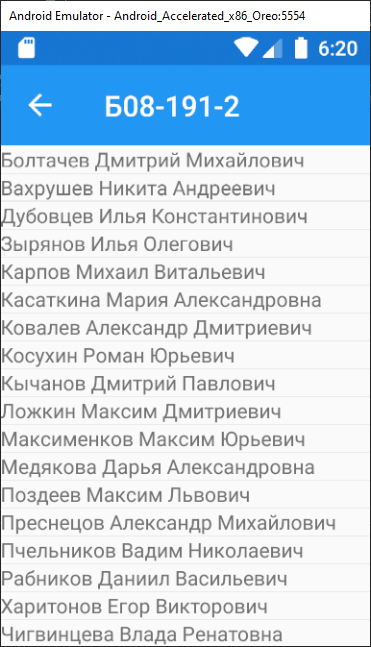
Список студентов в группе

Рис. 2.19

Использование чата со студентом

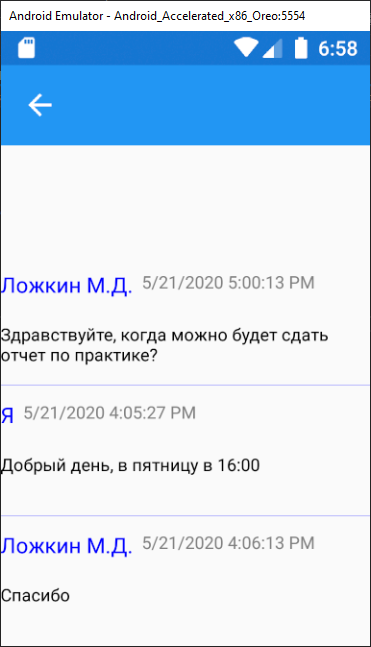


Рис. 2.20

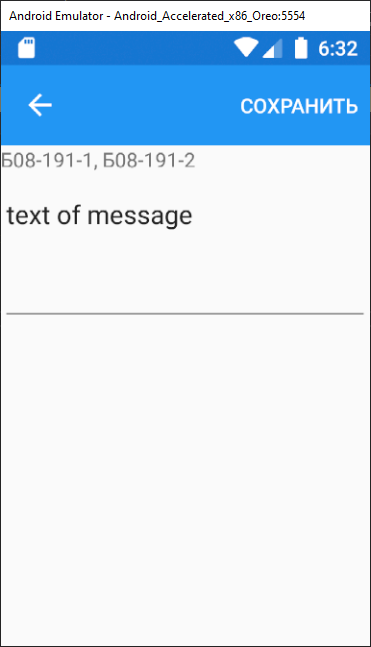
Форма добавления объявления

Рис. 2.21

Просмотр объявлений

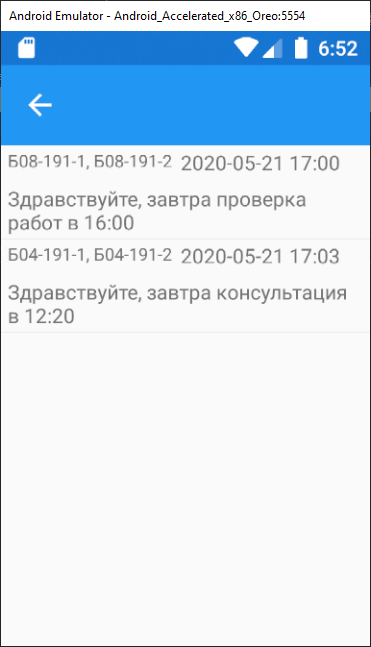


Рис. 2.22

#### 

#### 2.3.5.4. Результаты испытания программы

Из скриншотов можно увидеть, что пользователь может просматривать необходимую информацию. А также некоторые разделы позволяют редактирование информации.

## 2.4. Разработка системы важных объявлений от преподавателя студентам

### 2.4.1. Характеристика задачи

Задача – разработка системы отправки объявлений от преподавателя студентам. Система предназначена для сообщения важной информации сразу большому количеству студентов не используя чат с каждым из них и не передавая эту информацию одному студенту для её распространения.

### 2.4.2. Входная информация

В разделе, содержащем список групп и студентов преподаватель может отметить нужные ему группы и написать сообщение для студентов выбранных групп и отправить его. От пользователя требуется отметить нужные группы в списке и ввести текст сообщения.

### 2.4.3. Выходная информация

При открытии формы пользователю предоставляется список групп, которых он обучает, далее пользователь может открыть список нужной ему группы для того, чтобы написать студенту сообщение в чате или отметить нужные группы и оставить им сообщение.

После того, как пользователь отмечает группы и нажимает кнопку открытия формы сообщения, ему предоставляется текстовое поле, куда можно написать нужный текст.

### 2.4.4. Алгоритм решения

При открытии списка групп приложение посылает запрос на сервер, в ответ на которое получает список студентов. После заполнения текста сообщения пользователь отправляет его, а приложение отправляет данные на сервер.

Преподаватель выбирает в списке групп, которые он обучает несколько или выбирает только одну и пишет текст объявления, после чего сохраняет сообщение, и оно отправляется на сервер. Далее указаны используемые формы и методы:

1. StudentsGroups – форма, содержащая связанные с преподавателем группы студентов. В форме можно отметить несколько групп и нажать кнопку «Написать», после чего открывается форма заполнения сообщения;
2. AddAnnotationPage – форма для написания сообщения студентам, содержит метод AddAnnotation() для отправки запроса с сообщением на сервер;
3. AnnotationsPage – форма просмотра списка отправленных объявлений.

Далее на рисунке 18 показана схема добавления объявлений. Схема просмотра объявлений представлена на рисунке 2.23.

Схема добавления новых объявлений



Рис. 2.23

### 2.4.5. Описание контрольного примера

#### 2.4.5.1. Назначение

Назначение системы – отправка преподавателем объявлений для студентов целых групп.

#### 2.4.5.2. Исходные данные

Для добавления объявления от пользователя требуется выбрать группы и ввести текст сообщения. Для просмотра списка объявлений пользователь должен перейти в соответствующий раздел в главном меню.

#### 2.4.5.3. Результаты расчета

В результате пользователь должен получить возможность добавления новых объявлений нескольким группам студентов. После отправки сообщения должны появляться в соответствующем разделе в главном меню приложения. Скриншоты работы программы приведены на рисунках 2.24 и 2.25.

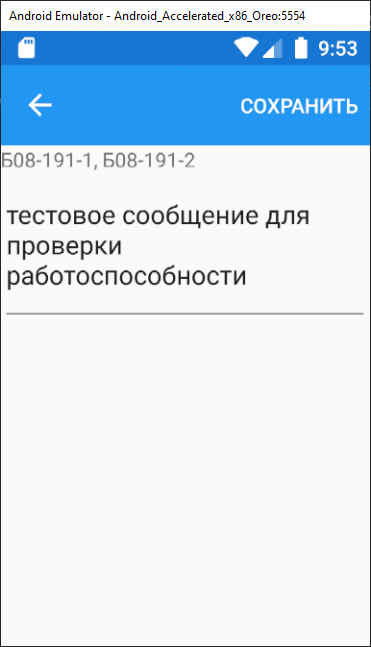
Тестирование ввода сообщения

Рис. 2.24

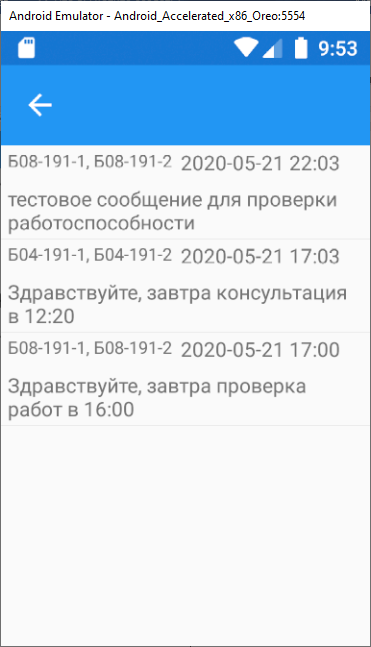
Просмотр добавленного объявления в списке

Рис. 2.25

#### 2.4.5.4. Результаты испытания программы

В результате работы программы мы видим, как пользователь может добавить объявление для группы студентов и увидеть его в списке отправленных объявлений. ­

# Заключение

В ходе работы было разработано приложение, предоставляющее преподавателю удобный доступ к личному кабинету ИжГТУ с мобильного устройства.

Полученное приложение отвечает всем ожидаемым требованиям, а именно:

* приложение предоставляет доступ студентам и преподавателем с разделением ролей и функционала в зависимости от роли пользователя;
* приложение использует API-запросы к серверу сайта ИжГТУ;
* архитектура позволяет новым разработчикам быстро разобраться в системе для её доработки и сопровождения;
* интерфейс пользователя выполнен в удобной форме без лишних элементов.

и поставленным целям:

* функционал приложения для взаимодействия с сервером;
* функционал приложения для использования преподавателем;
* система важных объявлений студентам от преподавателя.

# Список использованных источников

1. Metanit Руководство по программированию для Xamarin Forms [Электронный ресурс], URL: <https://metanit.com/sharp/xamarin/> .
2. Эндрю Троелсен. Язык программирования C# 7 и платформы .NET и .NET Core. М.: изд-во Издательский дом «Вильямс», 2018г.
3. ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению. – М.: Издательство стандартов, 1980.
4. ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению. – М.: Издательство стандартов, 1980.
5. ГОСТ 19.402-78 ЕСПД. Описание программы. – М.: Издательство стандартов, 1980.
6. ГОСТ 19.505-79 ЕСПД. Руководство оператора. Требования к содержанию и оформлению. – М.: Издательство стандартов, 1980.
7. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М.: Издательство стандартов, 1991.
8. ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению. – М.: Издательство стандартов, 1980.
9. Методические указания по оформлению курсовых и выпускных квалификационных работ для студентов, обучающихся на кафедре «Программное обеспечение». – Ижевск: Издательство ИжГТУ, 2018.

ПРИЛОЖЕНИЕ 1

# Текст программы

П. 1.1. Разметка форм

MainMenuPage.xaml

<?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="TryXamarin.Views.TeacherViews.MainMenu.MainMenuPage" Title="Меню">

<ContentPage.ToolbarItems>

<ToolbarItem x:Name="LogOutButton" Clicked="LogOutButton\_OnClicked" Text="LogOut" />

</ContentPage.ToolbarItems>

<ContentPage.Content>

<StackLayout>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="60" />

<RowDefinition Height="60" />

<RowDefinition Height="60" />

<RowDefinition Height="60" />

<RowDefinition Height="60" />

</Grid.RowDefinitions>

<Button Grid.Row="0" x:Name="UserButton" Text="Информация" Clicked="UserButton\_OnClicked" />

<Button Grid.Row="1" x:Name="ConsultationsButton" Text="Консультации"

Clicked="ConsultationsButton\_OnClicked" />

<Button Grid.Row="2" x:Name="PublicationsButton" Text="Публикации"

Clicked="PublicationsButton\_OnClicked" />

<Button Grid.Row="3" x:Name="StudentsButtonButton" Text="Студенты"

Clicked="StudentsButtonButton\_OnClicked" />

<Button Grid.Row="4" x:Name="AnnotationsButton" Text="Объявления"

Clicked="AnnotationsButton\_OnClicked" />

<Button Grid.Row="4" x:Name="MessagesButton" Text="Студентота"

Clicked="MessagesButton\_OnClicked" />

</Grid>

</StackLayout>

</ContentPage.Content>

</ContentPage>

MainMenuPage.xaml.cs

<?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="TryXamarin.Views.TeacherViews.MainMenu.MainMenuPage" Title="Меню">

<ContentPage.ToolbarItems>

<ToolbarItem x:Name="LogOutButton" Clicked="LogOutButton\_OnClicked" Text="LogOut" />

</ContentPage.ToolbarItems>

<ContentPage.Content>

<StackLayout>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="60" />

<RowDefinition Height="60" />

<RowDefinition Height="60" />

<RowDefinition Height="60" />

<RowDefinition Height="60" />

</Grid.RowDefinitions>

<Button Grid.Row="0" x:Name="UserButton" Text="Информация" Clicked="UserButton\_OnClicked" />

<Button Grid.Row="1" x:Name="ConsultationsButton" Text="Консультации"

Clicked="ConsultationsButton\_OnClicked" />

<Button Grid.Row="2" x:Name="PublicationsButton" Text="Публикации"

Clicked="PublicationsButton\_OnClicked" />

<Button Grid.Row="3" x:Name="StudentsButtonButton" Text="Студенты"

Clicked="StudentsButtonButton\_OnClicked" />

<Button Grid.Row="4" x:Name="AnnotationsButton" Text="Объявления"

Clicked="AnnotationsButton\_OnClicked" />

<Button Grid.Row="4" x:Name="MessagesButton" Text="Студентота"

Clicked="MessagesButton\_OnClicked" />

</Grid>

</StackLayout>

</ContentPage.Content>

</ContentPage>

UserPage.xaml

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

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

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

xmlns:dataGrid="clr-namespace:DevExpress.Mobile.DataGrid;assembly=DevExpress.Mobile.Grid.v18.2"

x:Class="TryXamarin.Views.TeacherViews.User.UserPage"

Title="Личная информация">

<ContentPage.ToolbarItems>

<ToolbarItem x:Name="Save" Clicked="Save\_OnClicked" Text="Сохранить" />

</ContentPage.ToolbarItems>

<ContentPage.Content>

<StackLayout>

<Label x:Name="FioLabel" FontSize="12"/>

<Entry x:Name="PhoneEntry" PropertyChanged="Staff\_OnChanged" FontSize="12" />

<Entry x:Name="AddressEntry" PropertyChanged="Staff\_OnChanged" FontSize="12"/>

<Entry x:Name="EmailEntry" PropertyChanged="Staff\_OnChanged" FontSize="12"/>

<ScrollView>

<ListView x:Name="PostsView" HasUnevenRows="True" />

</ScrollView>

</StackLayout>

</ContentPage.Content>

</ContentPage>

UserPage.xaml.cs

using System;

using System.ComponentModel;

using TryXamarin.DataHelpers;

using TryXamarin.Models.Teacher;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using PropertyChangingEventArgs = Xamarin.Forms.PropertyChangingEventArgs;

namespace TryXamarin.Views.TeacherViews.User

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class UserPage : ContentPage

{

private bool \_isChanged;

private readonly Staff \_staff;

private readonly Models.Common.User \_user;

public UserPage()

{

\_isChanged = false;

\_staff = GlobalMethods.GetStaff();

\_user = GlobalMethods.GetUser();

InitializeComponent();

FillInfo();

}

private void FillInfo()

{

FioLabel.SetBinding(Label.TextProperty, new Binding {Source = \_staff, Path = "Fio"});

PhoneEntry.SetBinding(Entry.TextProperty, new Binding {Source = \_staff, Path = "Phone"});

AddressEntry.SetBinding(Entry.TextProperty, new Binding {Source = \_staff, Path = "Address"});

EmailEntry.SetBinding(Entry.TextProperty, new Binding {Source = \_staff, Path = "Email"});

PostsView.ItemsSource = \_user.Staff.Posts;

PostsView.ItemTemplate = new DataTemplate(() =>

{

var parentDepTitle = new Label {FontSize = 16};

parentDepTitle.SetBinding(Label.TextProperty, "ParentDepartmentTitle");

var depTitle = new Label {FontSize = 14};

depTitle.SetBinding(Label.TextProperty, "DepartmentTitle");

var postTitle = new Label {FontSize = 16};

postTitle.SetBinding(Label.TextProperty, "PostTitle");

var postsLayout = new StackLayout

{

Padding = new Thickness(2, 5),

Orientation = StackOrientation.Vertical,

Children = {parentDepTitle, depTitle, postTitle}

};

return new ViewCell

{

View = postsLayout

};

});

}

private void Staff\_OnChanged(object sender, PropertyChangedEventArgs e)

{

\_isChanged = true;

}

private void Save\_OnClicked(object sender, EventArgs e)

{

if (\_isChanged) GlobalMethods.UpdateStaff(\_staff);

}

}

}

AnnouncementsPage.xaml

<?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="TryXamarin.Views.TeacherViews.Announcements.AnnouncementsPage">

<ContentPage.Content>

<ScrollView>

<ListView x:Name="AnnotationsListView" HasUnevenRows="True"/>

</ScrollView>

</ContentPage.Content>

</ContentPage>

AnnouncementsPage.xaml.cs

using System.Collections.Generic;

using System.Linq;

using TryXamarin.DataHelpers;

using TryXamarin.Models.Common;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace TryXamarin.Views.TeacherViews.Announcements

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class AnnouncementsPage : ContentPage

{

private List<Annotation> \_annotations;

public AnnouncementsPage()

{

\_annotations = GlobalMethods.GetAnnotations();

InitializeComponent();

FillData();

}

private void FillData()

{

AnnotationsListView.ItemsSource = \_annotations;

AnnotationsListView.ItemTemplate = new DataTemplate(() =>

{

var groupsLabel = new Label { FontSize = 12 };

groupsLabel.SetBinding(Label.TextProperty, "GroupsLine");

var timeLabel = new Label {FontSize = 14};

timeLabel.SetBinding(Label.TextProperty, "CreatedAt");

var rowStack = new StackLayout

{

Orientation = StackOrientation.Horizontal,

Children = {groupsLabel, timeLabel}

};

var textLabel = new Label {FontSize = 14};

textLabel.SetBinding(Label.TextProperty, "Text");

var stackLayout = new StackLayout

{

Padding = new Thickness(5, 2),

Orientation = StackOrientation.Vertical,

Children = { rowStack, textLabel }

};

return new ViewCell

{

View = stackLayout

};

});

}

}

}

AddAnnouncementsPage.xaml

<?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="TryXamarin.Views.TeacherViews.Announcements.AddAnnouncementPage">

<ContentPage.ToolbarItems>

<ToolbarItem x:Name="Save" Text="Сохранить" Clicked="Save\_OnClicked"/>

</ContentPage.ToolbarItems>

<ContentPage.Content>

<StackLayout>

<Label x:Name="GroupsLabel"/>

<Editor x:Name="TextEditor" HeightRequest="100"/>

</StackLayout>

</ContentPage.Content>

</ContentPage>

AddAnnouncementPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using TryXamarin.DataHelpers;

using TryXamarin.Models.Common;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace TryXamarin.Views.TeacherViews.Announcements

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class AddAnnouncementPage : ContentPage

{

private Annotation \_annotation;

private string \_groups;

public AddAnnouncementPage(string[] groups)

{

\_groups = string.Join(", ", groups);

InitializeComponent();

FillInfo();

}

private void FillInfo()

{

GroupsLabel.Text = \_groups;

}

private void Save\_OnClicked(object sender, EventArgs e)

{

if (!string.IsNullOrEmpty(TextEditor.Text) && !string.IsNullOrWhiteSpace(TextEditor.Text))

{

GlobalMethods.AddAnnotation(new Annotation

{

TargetGroups = \_groups.Split(',')

.Select(x => x.Replace(' ', (char)0)).ToList(),

Text = TextEditor.Text

});

}

}

}

}

ConsultationsPage.xaml

<?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="TryXamarin.Views.TeacherViews.Consultations.ConsultationsPage">

<ContentPage.ToolbarItems>

<!--<ToolbarItem x:Name="Edit" Clicked="Edit\_OnClicked" Text="Edit" />

<ToolbarItem x:Name="Remove" Clicked="Remove\_OnClicked" Text="[X]" />-->

<ToolbarItem x:Name="Add" Clicked="Add\_OnClicked" Text="Add" />

</ContentPage.ToolbarItems>

<ContentPage.Content>

<StackLayout>

<ListView x:Name="ConsultationsListView" HasUnevenRows="True" />

</StackLayout>

</ContentPage.Content>

</ContentPage>

ConsultationsPage.xaml.cs

using System;

using System.Collections.Generic;

using TryXamarin.DataHelpers;

using TryXamarin.Models.Common;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace TryXamarin.Views.TeacherViews.Consultations

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class ConsultationsPage : ContentPage

{

private List<Consultation> \_consultations;

public ConsultationsPage()

{

\_consultations = GlobalMethods.GetConsultations();

InitializeComponent();

FillInfo();

}

private Consultation \_selectedConsultation { get; set; }

private void FillInfo()

{

ConsultationsListView.ItemsSource = \_consultations;

ConsultationsListView.SetBinding(ListView.SelectedItemProperty, new Binding("\_selectedConsultation", BindingMode.TwoWay));

ConsultationsListView.SelectionMode = ListViewSelectionMode.Single;

ConsultationsListView.ItemTemplate = new DataTemplate(() =>

{

var commentLabel = new Label {FontSize = 10};

commentLabel.SetBinding(Label.TextProperty, "Comment");

var addressLabel = new Label {FontSize = 14};

addressLabel.SetBinding(Label.TextProperty, "Address");

var timeStartLabel = new Label {FontSize = 12};

timeStartLabel.SetBinding(Label.TextProperty, "TimeStart");

var timeEndLabel = new Label {FontSize = 12};

timeEndLabel.SetBinding(Label.TextProperty, "TimeEnd");

var dayLabel = new Label

{

FontSize = 14,

HorizontalTextAlignment = TextAlignment.Center,

Margin = new Thickness(2, 5)

};

dayLabel.SetBinding(Label.TextProperty, "Day");

var removeButton = new Button

{

WidthRequest = 30,

HeightRequest = 30,

Text = "X",

FontSize = 8,

TextColor = Color.Red

};

removeButton.Clicked += Remove\_OnClicked;

var editButton = new Button

{

WidthRequest = 30,

HeightRequest = 30,

Text = "E",

FontSize = 8,

TextColor = Color.DodgerBlue

};

editButton.Clicked += Edit\_OnClicked;

var buttonsStack = new StackLayout

{

Orientation = StackOrientation.Vertical,

Children = {editButton, removeButton}

};

var detailsView = new StackLayout

{

Padding = new Thickness(2, 5),

Orientation = StackOrientation.Vertical,

Children = {addressLabel, commentLabel}

};

var timeView = new StackLayout

{

Padding = new Thickness(2, 5),

Orientation = StackOrientation.Vertical,

Children = {timeStartLabel, timeEndLabel}

};

var gridView = new Grid

{

Padding = new Thickness(2, 5),

RowDefinitions = new RowDefinitionCollection

{

new RowDefinition

{

Height = GridLength.Auto

}

},

ColumnDefinitions = new ColumnDefinitionCollection

{

new ColumnDefinition {Width = GridLength.Star},

new ColumnDefinition {Width = GridLength.Auto},

new ColumnDefinition {Width = GridLength.Auto},

new ColumnDefinition {Width = GridLength.Auto}

}

};

gridView.Children.Add(detailsView, 0, 0);

gridView.Children.Add(dayLabel, 1, 0);

gridView.Children.Add(timeView, 2, 0);

gridView.Children.Add(buttonsStack, 3, 0);

return new ViewCell

{

View = gridView

};

});

}

private void Edit\_OnClicked(object sender, EventArgs e)

{

if (\_selectedConsultation != null)

Navigation.PushAsync(new EditConsultationPage(\_selectedConsultation));

}

private void Add\_OnClicked(object sender, EventArgs e)

{

Navigation.PushAsync(new EditConsultationPage());

}

private void Remove\_OnClicked(object sender, EventArgs e)

{

if (ConsultationsListView.SelectedItem != null)

\_consultations.Remove(\_selectedConsultation);

}

}

}

EditConsultationPage.xaml

<?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="TryXamarin.Views.TeacherViews.Consultations.EditConsultationPage">

<ContentPage.ToolbarItems>

<ToolbarItem x:Name="Save" Text="Сохранить" Clicked="Save\_OnClicked" />

</ContentPage.ToolbarItems>

<ContentPage.Content>

<StackLayout>

<Label Text="Место проведения:" Margin="5" FontSize="16" HorizontalTextAlignment="Center" />

<Entry x:Name="AddressEntry" FontSize="14" />

<Label Text="Комментарий:" Margin="5" FontSize="16" HorizontalTextAlignment="Center" />

<Editor x:Name="CommentEditor" FontSize="14" HeightRequest="50" />

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

</Grid.RowDefinitions>

<Label Grid.Row="0" Grid.Column="0" Text="Начало:" FontSize="14" HorizontalTextAlignment="Center" />

<Label Grid.Row="0" Grid.Column="1" Text="Окончание:" FontSize="14" HorizontalTextAlignment="Center" />

<TimePicker x:Name="StartTimePicker" Grid.Row="1" Grid.Column="0" FontSize="14" />

<TimePicker x:Name="EndTimePicker" Grid.Row="1" Grid.Column="1" FontSize="14" />

</Grid>

<Label Text="День проведения:" Margin="5" FontSize="16" HorizontalTextAlignment="Center" />

<Picker x:Name="DayPicker" />

</StackLayout>

</ContentPage.Content>

</ContentPage>

EditConsultationPage.xaml.cs

using System;

using System.Collections.Generic;

using TryXamarin.DataHelpers;

using TryXamarin.Models.Common;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace TryXamarin.Views.TeacherViews.Consultations

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class EditConsultationPage : ContentPage

{

private readonly Consultation \_consultation;

public EditConsultationPage()

{

\_consultation = new Consultation();

InitializeComponent();

FillData();

}

public EditConsultationPage(Consultation consultation)

{

\_consultation = consultation;

InitializeComponent();

FillData();

}

private void FillData()

{

DayPicker.ItemsSource = new List<string> {"пн", "вт", "ср", "чт", "пт", "сб", "вс"};

AddressEntry.Text = \_consultation.Address;

CommentEditor.Text = \_consultation.Comment;

DayPicker.SelectedItem = DayFormat(\_consultation.Day);

StartTimePicker.Time = TimeSpan.Parse(\_consultation.TimeStart);

EndTimePicker.Time = TimeSpan.Parse(\_consultation.TimeEnd);

}

private void Save\_OnClicked(object sender, EventArgs e)

{

if (!string.IsNullOrEmpty(\_consultation.Address))

{

\_consultation.Address = AddressEntry.Text;

\_consultation.Comment = CommentEditor.Text;

\_consultation.Day = DayPicker.SelectedItem.ToString();

\_consultation.TimeEnd = EndTimePicker.Time.ToString("hh:mm:00");

\_consultation.TimeStart = StartTimePicker.Time.ToString("hh:mm:00");

\_consultation.UpdatedAt = DateTime.Now.ToString("yyyy-MM-dd HH:mm:cc");

if (\_consultation.CreatedAt == null)

\_consultation.CreatedAt = DateTime.Now.ToString("yyyy-MM-dd HH:mm:cc");

}

GlobalMethods.AddOrUpdateConsultation(\_consultation);

}

private string DayFormat(string day)

{

return day;

}

}

}

PublicationsPage.xaml

<?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="TryXamarin.Views.TeacherViews.Publications.PublicationsPage">

<ContentPage.Content>

<ScrollView>

<ListView x:Name="PublicationsListView" HasUnevenRows="True"/>

</ScrollView>

</ContentPage.Content>

</ContentPage>

PublicationsPage.xaml.cs

using System.Collections.Generic;

using TryXamarin.DataHelpers;

using TryXamarin.Models.Common;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace TryXamarin.Views.TeacherViews.Publications

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class PublicationsPage : ContentPage

{

private List<Article> \_articles;

public PublicationsPage()

{

\_articles = GlobalMethods.GetArticles();

InitializeComponent();

FillData();

}

private void FillData()

{

PublicationsListView.ItemsSource = \_articles;

PublicationsListView.ItemTemplate = new DataTemplate(() =>

{

var titleLabel = new Label { FontSize = 10 };

titleLabel.SetBinding(Label.TextProperty, "TitleAndYear");

var infoLabel = new Label {FontSize = 8};

infoLabel.SetBinding(Label.TextProperty, "Info");

var stackLayout = new StackLayout

{

Padding = new Thickness(2, 5),

Orientation = StackOrientation.Vertical,

Children = {titleLabel, infoLabel}

};

return new ViewCell

{

View = stackLayout

};

});

}

}

}

StudentsGroupsPage.xaml

<?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="TryXamarin.Views.TeacherViews.StundentsGroups.StudentsGroupsPage">

<ContentPage.ToolbarItems>

<ToolbarItem x:Name="OpenGroup" Text="Открыть" Clicked="OpenGroup\_OnClicked"/>

<ToolbarItem x:Name="AddAnnotation" Text="Объявление" Clicked="AddAnnotation\_OnClicked"/>

</ContentPage.ToolbarItems>

<ContentPage.Content>

<ScrollView>

<ListView x:Name="GroupsListView"/>

</ScrollView>

</ContentPage.Content>

</ContentPage>

StudentsGroupsPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using TryXamarin.Models.Common;

using TryXamarin.Models.Teacher;

using TryXamarin.Views.TeacherViews.Announcements;

using TryXamarin.Views.TeacherViews.Stundents;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace TryXamarin.Views.TeacherViews.StundentsGroups

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class StudentsGroupsPage : ContentPage

{

private List<StudentsGroup> \_groups;

private StudentsGroup \_selectedGroup;

public StudentsGroupsPage()

{

\_groups = new List<StudentsGroup>

{

new StudentsGroup

{

Name = "Б04-191-1"

},

new StudentsGroup

{

Name = "Б04-191-2"

},

new StudentsGroup

{

Name = "Б08-191-1"

},

new StudentsGroup

{

Name = "Б08-191-2"

},

};

InitializeComponent();

FillData();

}

private void FillData()

{

GroupsListView.ItemsSource = \_groups;

GroupsListView.ItemTemplate = new DataTemplate(() =>

{

var groupLabel = new Label { FontSize = 10 };

groupLabel.SetBinding(Label.TextProperty, "Name");

return new ViewCell

{

View = groupLabel

};

});

}

private void OpenGroup\_OnClicked(object sender, EventArgs e)

{

Navigation.PushAsync(new StudentsListPage(\_selectedGroup));

}

private void AddAnnotation\_OnClicked(object sender, EventArgs e)

{

Navigation.PushAsync(new AddAnnouncementPage());

}

}

}

StudentsListPage.xaml

<?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="TryXamarin.Views.TeacherViews.Stundents.StudentsListPage">

<ContentPage.Content>

<ScrollView>

<ListView x:Name="StudentsListView" HasUnevenRows="True"/>

</ScrollView>

</ContentPage.Content>

</ContentPage>

StudentsListPage.xaml.cs

using System;

using System.Linq;

using TryXamarin.Models.Common;

using TryXamarin.Models.Student;

using TryXamarin.Models.Teacher;

using TryXamarin.Views.CommonViews;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace TryXamarin.Views.TeacherViews.Stundents

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class StudentsListPage : ContentPage

{

private StudentsGroup \_group;

public StudentsListPage(StudentsGroup group)

{

\_group = group;

Title = \_group.Name;

InitializeComponent();

FillData();

}

private void FillData()

{

StudentsListView.ItemsSource = \_group.Students;

StudentsListView.ItemTemplate = new DataTemplate(() =>

{

var fioLabel = new Label { FontSize = 14 };

fioLabel.SetBinding(Label.TextProperty, "Fio");

return new ViewCell

{

View = fioLabel

};

});

}

private void ChatWithStudent\_OnClicked(object sender, EventArgs e)

{

Navigation.PushAsync(new ChatPage(1));

}

}

}

П. 1.2. Глобальные статические классы

GlobalMethods.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Security.Authentication;

using Android.Media;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

using TryXamarin.Models.Common;

using TryXamarin.Models.Student;

using TryXamarin.Models.Teacher;

namespace TryXamarin.DataHelpers

{

public class GlobalMethods

{

#region Common

public static UserAuthorization Authorize(string email, string password)

{

var baseAdress = "https://istu.ru/api/auth/";

var content = new Dictionary<string, string>

{

{"email", $"{email}"},

{"password", $"{password}"},

{"ContentType", "application/json"}

};

var postAsync = App.HttpClient.PostAsync(new Uri($"{baseAdress}login"), new FormUrlEncodedContent(content))

.Result;

if (postAsync.StatusCode == HttpStatusCode.OK)

{

GlobalVariables.UserAuthorization =

JsonConvert.DeserializeObject<UserAuthorization>(postAsync.Content.ReadAsStringAsync().Result);

return GlobalVariables.UserAuthorization;

}

throw new AuthenticationException();

}

public static User GetUser()

{

var httpClient = App.HttpClient;

//httpClient.BaseAddress = new Uri("https://istu.ru/api");

var baseAdress = "https://istu.ru/api/auth/";

var request = new HttpRequestMessage

{

Method = HttpMethod.Get,

RequestUri = new Uri($"{baseAdress}user")

};

httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

request.Headers.Add("Authorization", $"Bearer {GlobalVariables.UserAuthorization.AccessToken}");

var httpResponseMessage = httpClient.SendAsync(request).Result;

if (httpResponseMessage.IsSuccessStatusCode)

{

var ans = httpResponseMessage.Content.ReadAsStringAsync().Result;

var user = JsonConvert.DeserializeObject<User>(ans);

return user;

}

return null;

}

public static List<Consultation> GetConsultations()

{

var httpClient = App.HttpClient;

var baseAdress = "https://istu.ru/api/mobile/staff";

var request = new HttpRequestMessage

{

Method = HttpMethod.Get,

RequestUri = new Uri($"{baseAdress}?staff\_id={GlobalVariables.CurrentUser.Staff.Id}&consultation")

};

httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

request.Headers.Add("Authorization", $"Bearer {GlobalVariables.UserAuthorization.AccessToken}");

var httpResponseMessage = httpClient.SendAsync(request).Result;

if (httpResponseMessage.IsSuccessStatusCode)

{

var answer = httpResponseMessage.Content.ReadAsStringAsync().Result;

var staff = JsonConvert

.DeserializeObject<Staff>(answer, new JsonSerializerSettings

{

NullValueHandling = NullValueHandling.Ignore,

MissingMemberHandling = MissingMemberHandling.Ignore

});

return staff.Consultations;

}

return null;

}

public static List<Announcement> GetAnnouncements(bool forTeacher)

{

var httpClient = App.HttpClient;

var baseAdress = "https://istu.ru/api/mobile/announcement";

var request = new HttpRequestMessage();

if (forTeacher)

{

request = new HttpRequestMessage

{

Method = HttpMethod.Get,

RequestUri = new Uri($"{baseAdress}?from\_userid={GlobalVariables.CurrentUser.Staff.Id}")

};

}

else

{

request = new HttpRequestMessage

{

Method = HttpMethod.Get,

RequestUri = new Uri($"{baseAdress}?for\_group={GlobalVariables.CurrentUser.Students.First().Group}")

};

}

httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

request.Headers.Add("Authorization", $"Bearer {GlobalVariables.UserAuthorization.AccessToken}");

var httpResponseMessage = httpClient.SendAsync(request).Result;

if (httpResponseMessage.IsSuccessStatusCode)

{

var answer = httpResponseMessage.Content.ReadAsStringAsync().Result;

var announcements = JsonConvert

.DeserializeObject<List<Announcement>>(answer, new JsonSerializerSettings

{

NullValueHandling = NullValueHandling.Ignore,

MissingMemberHandling = MissingMemberHandling.Ignore

});

return announcements;

}

return null;

}

public static Staff GetStaff()

{

var httpClient = App.HttpClient;

var baseAdress = "https://istu.ru/api/mobile/staff";

var request = new HttpRequestMessage

{

Method = HttpMethod.Get,

RequestUri = new Uri($"{baseAdress}?staff\_id={GlobalVariables.CurrentUser.Staff.Id}&article")

};

httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

request.Headers.Add("Authorization", $"Bearer {GlobalVariables.UserAuthorization.AccessToken}");

var httpResponseMessage = httpClient.SendAsync(request).Result;

if (httpResponseMessage.IsSuccessStatusCode)

{

var answer = httpResponseMessage.Content.ReadAsStringAsync().Result;

var staff = JsonConvert

.DeserializeObject<Staff>(answer, new JsonSerializerSettings

{

NullValueHandling = NullValueHandling.Ignore,

MissingMemberHandling = MissingMemberHandling.Ignore

});

return staff;

}

return null;

}

public static List<MessageBasic> GetMessages(Guid teacherId)

{

return new List<MessageBasic>();

}

public static void AddMessage(MessageBasic messageBasic)

{

}

#endregion

#region Student

public static List<StudentsMark> GetSession()

{

var httpClient = App.HttpClient;

var baseAdress = "https://istu.ru/api/mobile/session";

var request = new HttpRequestMessage

{

Method = HttpMethod.Get,

RequestUri = new Uri($"{baseAdress}?student\_id={GlobalVariables.CurrentUser.Students.First().Id}")

};

httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

request.Headers.Add("Authorization", $"Bearer {GlobalVariables.UserAuthorization.AccessToken}");

var httpResponseMessage = httpClient.SendAsync(request).Result;

if (httpResponseMessage.IsSuccessStatusCode)

{

var answer = httpResponseMessage.Content.ReadAsStringAsync().Result;

var marks = JsonConvert

.DeserializeObject<List<StudentsMark>>(answer, new JsonSerializerSettings

{

NullValueHandling = NullValueHandling.Ignore,

MissingMemberHandling = MissingMemberHandling.Ignore

});

return marks;

}

return null;

}

public static StudentDebt GetDebts()

{

try

{

var httpClient = App.HttpClient;

var baseAdress = "https://istu.ru/api/mobile/student";

var request = new HttpRequestMessage

{

Method = HttpMethod.Get,

RequestUri =

new Uri($"{baseAdress}?student\_id={GlobalVariables.CurrentUser.Students.First().Id}&finance")

};

httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

request.Headers.Add("Authorization", $"Bearer {GlobalVariables.UserAuthorization.AccessToken}");

var httpResponseMessage = httpClient.SendAsync(request).Result;

if (httpResponseMessage.IsSuccessStatusCode)

{

var answer = httpResponseMessage.Content.ReadAsStringAsync().Result;

var jObject = JObject.Parse(answer);

var debt = jObject["contract"].ToObject<StudentDebt>();

return debt;

}

}

catch (Exception ex)

{

return null;

}

return null;

}

public static List<Teacher> GetTeachers()

{

return new List<Teacher>();

}

#endregion

#region Teacher

public static List<Article> GetArticles()

{

var httpClient = App.HttpClient;

var baseAdress = "https://istu.ru/api/mobile/staff";

var request = new HttpRequestMessage

{

Method = HttpMethod.Get,

RequestUri = new Uri($"{baseAdress}?staff\_id={GlobalVariables.CurrentUser.Staff.Id}&article")

};

httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

request.Headers.Add("Authorization", $"Bearer {GlobalVariables.UserAuthorization.AccessToken}");

var httpResponseMessage = httpClient.SendAsync(request).Result;

if (httpResponseMessage.IsSuccessStatusCode)

{

var answer = httpResponseMessage.Content.ReadAsStringAsync().Result;

var staff = JsonConvert

.DeserializeObject<Staff>(answer, new JsonSerializerSettings

{

NullValueHandling = NullValueHandling.Ignore,

MissingMemberHandling = MissingMemberHandling.Ignore

});

return staff.Articles;

}

return null;

}

public static void UpdateStaff(Staff staff)

{

var httpClient = App.HttpClient;

var baseAdress = "https://istu.ru/api/mobile/updStaff";

httpClient.PostAsync(baseAdress, new StringContent(JsonConvert.SerializeObject(staff,

new JsonSerializerSettings

{

NullValueHandling = NullValueHandling.Include

})));

}

public static void AddOrUpdateConsultation(Consultation consultation)

{

var httpClient = App.HttpClient;

var baseAdress = "https://istu.ru/api/mobile/updConsult";

httpClient.PostAsync(baseAdress, new StringContent(JsonConvert.SerializeObject(consultation,

new JsonSerializerSettings

{

NullValueHandling = NullValueHandling.Include

})));

}

public static void AddAnnouncement(Announcement announcement)

{

var httpClient = App.HttpClient;

var baseAdress = "https://istu.ru/api/mobile/addAnnoucement";

httpClient.PostAsync(baseAdress, new StringContent(JsonConvert.SerializeObject(announcement,

new JsonSerializerSettings

{

NullValueHandling = NullValueHandling.Include

})));

}

#endregion

}

}

GlobalVariables.cs

using TryXamarin.Models.Common;

namespace TryXamarin.DataHelpers

{

public static class GlobalVariables

{

public static UserAuthorization UserAuthorization;

public static User CurrentUser;

}

}

П. 1.3. Модели данных

Post.cs

using Newtonsoft.Json;

namespace TryXamarin.Models.Teacher

{

public class Post

{

[JsonProperty("Department\_Title")]

public string DepartmentTitle { get; set; }

[JsonProperty("Parent\_Department\_Title")]

public string ParentDepartmentTitle { get; set; }

[JsonProperty("Post\_Title")]

public string PostTitle { get; set; }

}

}

Staff.cs

using System;

using System.Collections.Generic;

using System.Text;

using Newtonsoft.Json;

using TryXamarin.Annotations;

using TryXamarin.Models.Common;

namespace TryXamarin.Models.Teacher

{

public class Staff

{

public string Id { get; set; }

[JsonProperty("Academic\_Degree")]

public string AcademicDegree { get; set; }

[JsonProperty("Academic\_Title")]

public string AcademicTitle { get; set; }

public string Code { get; set; }

[CanBeNull]

public bool IsPps { get; set; }

public List<Post> Posts { get; set; }

public string Fio { get; set; }

public string Photo { get; set; }

public string Email { get; set; }

public string Phone { get; set; }

public string Address { get; set; }

public List<Article> Articles { get; set; }

public List<Consultation> Consultations { get; set; }

}

}

StudentsGroup.cs

using System.Collections.Generic;

using TryXamarin.Models.Common;

namespace TryXamarin.Models.Teacher

{

public class StudentsGroup

{

public string Name { get; set; }

public List<Person> Students { get; set; }

}

}

Announcement.cs

using System;

using System.Collections.Generic;

namespace TryXamarin.Models.Common

{

public class Announcement

{

public string Text { get; set; }

public List<string> TargetGroups { get; set; }

public string CreatedAt { get; set; }

public string GroupsLine

{

get

{

return String.Join(", ", TargetGroups);

}

}

public DateTime CreatedAtDateTime

{

get { return DateTime.Parse(CreatedAt); }

set { CreatedAt = value.ToString("yyyy-MM-dd HH:mm:cc"); }

}

}

}

Article.cs

using System;

using System.Collections.Generic;

using System.Text;

using Newtonsoft.Json;

namespace TryXamarin.Models.Common

{

public class Article

{

public string Id { get; set; }

[JsonProperty("Genre\_Id")]

public string GenreId { get; set; }

[JsonProperty("Type\_Id")]

public string TypeId { get; set; }

public string Title { get; set; }

public int Year { get; set; }

[JsonProperty("Created\_At")]

public string CreatedAt { get; set; }

[JsonProperty("Updated\_At")]

public string UpdatedAt { get; set; }

public string Code { get; set; }

public string Info { get; set; }

public Pivot Pivot { get; set; }

public string TitleAndYear

{

get { return string.Format(Title + ", " + Year); }

}

}

}

ChatCollection.cs

using System;

using System.Collections.Generic;

using System.Text;

using Newtonsoft.Json;

namespace TryXamarin.Models.Common

{

public class ChatCollection

{

[JsonProperty("staffs")]

public List<ChatModel> Teachers { get; set; }

[JsonProperty("students")]

public List<ChatModel> Students { get; set; }

}

}

ChatModel.cs

using System;

using System.Collections.Generic;

using System.Text;

using Android.Media;

using Java.Util.Jar;

using Newtonsoft.Json;

using TryXamarin.Models.Teacher;

namespace TryXamarin.Models.Common

{

public class ChatModel

{

[JsonProperty("id")]

public int Id { get; set; }

[JsonProperty("staff\_id")]

public string StaffId { get; set; }

[JsonProperty("student\_id")]

public string StudentId { get; set; }

[JsonProperty("latest\_message")]

public ListMessage LatestMessageBasic { get; set; }

[JsonProperty("unread\_messages\_count")]

public string UnreadMessagesCount { get; set; }

[JsonProperty("staff")]

public Person Teacher { get; set; }

[JsonProperty("student")]

public Person Student { get; set; }

[JsonProperty("created\_at")]

public DateTime CreatedAt { get; set; }

[JsonProperty("updated\_at")]

public DateTime UpdatedAt { get; set; }

public string Name

{

get

{

if (Teacher != null)

return Teacher.Fio;

return Student.Fio;

}

}

public string LatestMessageText

{

get

{

if (LatestMessageBasic != null)

return LatestMessageBasic.Text;

return null;

}

}

}

}

Consultation.cs

using System;

using System.Collections.Generic;

using System.Text;

using Java.Util;

using Newtonsoft.Json;

namespace TryXamarin.Models.Common

{

public class Consultation

{

public Consultation()

{

Day = "пн";

TimeStart = "12:00:00";

TimeEnd = "12:00:00";

Address = "";

Comment = "";

CreatedAt = DateTime.Now.ToString("yyyy-MM-dd HH:mm:cc");

UpdatedAt = DateTime.Now.ToString("yyyy-MM-dd HH:mm:cc");

}

public string Id;

[JsonProperty("Staff\_Id")]

public string StaffId;

public string Day { get; set; }

[JsonProperty("Time\_Start")]

public string TimeStart { get; set; }

[JsonProperty("Time\_End")]

public string TimeEnd { get; set; }

public string Address { get; set; }

public string Comment { get; set; }

[JsonProperty("Created\_At")]

public string CreatedAt;

[JsonProperty("Updated\_At")]

public string UpdatedAt;

}

}

FioFull.cs

namespace TryXamarin.Models.Common

{

public class FioFull

{

public string Family;

public string Name;

public string Patronymic;

}

}

MessageBasic.cs

using System;

using Newtonsoft.Json;

using TryXamarin.DataHelpers;

namespace TryXamarin.Models.Common

{

public class MessageBasic

{

[JsonProperty("id")]

public string Id { get; set; }

[JsonProperty("user\_id")]

public string UserId { get; set; }

[JsonProperty("message")]

public string Text { get; set; }

[JsonProperty("isread")]

public bool IsRead { get; set; }

[JsonProperty("created\_at")]

public DateTime CreatedAt { get; set; }

[JsonProperty("updated\_at")]

public DateTime UpdatedAt { get; set; }

public Student.Teacher Teacher;

}

public class ChatMessage : MessageBasic

{

[JsonProperty("from")]

public MessageMetadata From { get; set; }

[JsonProperty("to")]

public MessageMetadata To { get; set; }

public string AuthorName

{

get

{

if (From.Id == GlobalVariables.CurrentUser.Id)

return "Я";

return From.Name;

}

}

}

public class ListMessage : MessageBasic

{

[JsonProperty("from")]

public int From { get; set; }

[JsonProperty("to")]

public int To { get; set; }

}

public struct MessageMetadata

{

public int Id { get; set; }

public string Name { get; set; }

}

}

Person.cs

using System;

using System.Collections.Generic;

using System.Text;

using Newtonsoft.Json;

namespace TryXamarin.Models.Common

{

public class Person

{

[JsonProperty("user\_id")]

public string UserId { get; set; }

public string Family { get; set; }

public string Name { get; set; }

public string Patronymic { get; set; }

public string Fio

{

get

{

return $"{Family} {Name} {Patronymic}";

}

}

}

}

Pivot.cs

using System;

using System.Collections.Generic;

using System.Text;

using Newtonsoft.Json;

namespace TryXamarin.Models.Common

{

public class Pivot

{

[JsonProperty("User\_Id")]

public string UserId;

[JsonProperty("Article\_Id")]

public string ArticleId;

}

}

User.cs

using System.Collections.Generic;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using Newtonsoft.Json;

using TryXamarin.Models.Teacher;

namespace TryXamarin.Models.Common

{

public class User : Person, INotifyPropertyChanged

{

private int \_id;

public int Id

{

get { return \_id; }

set

{

if (\_id != value)

{

\_id = value;

OnPropertyChanged("Id");

}

}

}

private string \_fio;

public string Fio

{

get { return \_fio; }

set

{

if (\_fio != value)

{

\_fio = value;

OnPropertyChanged("Fio");

}

}

}

private FioFull \_fioFull;

[JsonProperty("Fio\_Full")]

public FioFull FioFull

{

get { return \_fioFull; }

set

{

if (\_fioFull != value)

{

\_fioFull = value;

OnPropertyChanged("FioFull");

}

}

}

private string \_email;

public string Email

{

get { return \_email; }

set

{

if (\_email != value)

{

\_email = value;

OnPropertyChanged("Email");

}

}

}

private bool \_isPPS;

public bool IsPPS

{

get { return \_isPPS; }

set

{

if (\_isPPS != value)

{

\_isPPS = value;

OnPropertyChanged("IsPps");

}

}

}

public Staff \_staff;

public Staff Staff

{

get { return \_staff; }

set

{

if (\_staff != value)

{

\_staff = value;

OnPropertyChanged("Staff");

}

}

}

[JsonProperty("Student")]

public List<Student.Student> Students;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

public AuthorizationType AuthorizationType;

}

public enum AuthorizationType

{

Teacher,

Student

}

}

UserAuthorization.cs

using System;

using Newtonsoft.Json;

namespace TryXamarin.Models.Common

{

public class UserAuthorization

{

[JsonProperty("user\_id")]

public string Id;

[JsonProperty("isVerify")]

public bool isVerify;

[JsonProperty("access\_token")]

public string AccessToken;

[JsonProperty("token\_type")]

public string Token\_type;

[JsonProperty("expires\_at")]

public DateTime ExpiresAt;

}

}

ПРИЛОЖЕНИЕ 2

# Результаты работы программы

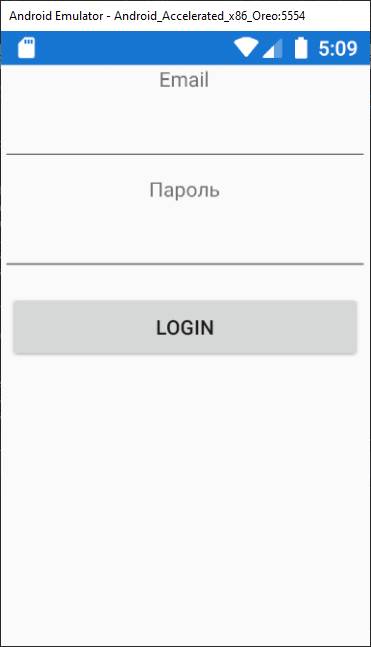
Окно авторизации

Рис. П.2.1

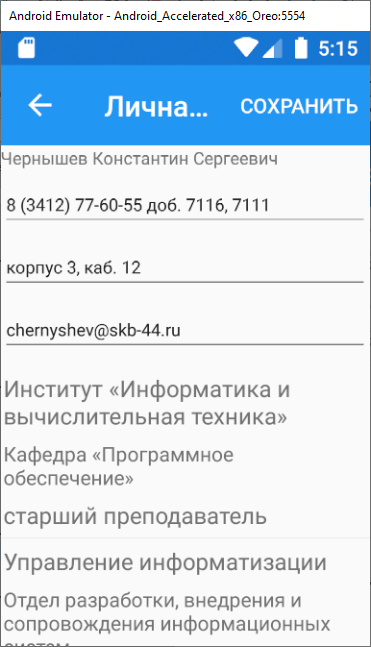
Форма личной информации преподавателя

Рис. П.2.3

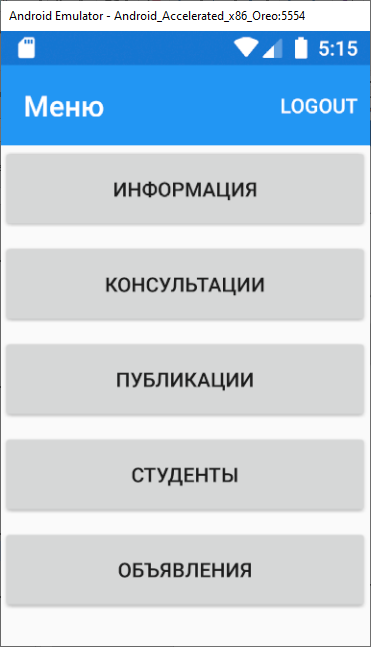
Главное меню приложения

Рис. П.2.2

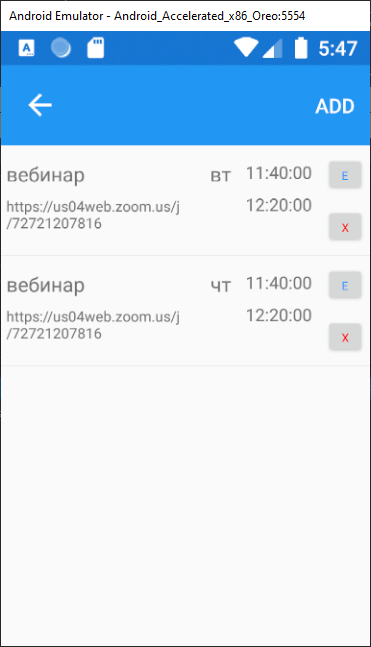
Список консультаций преподавателя

Рис. П.2.4

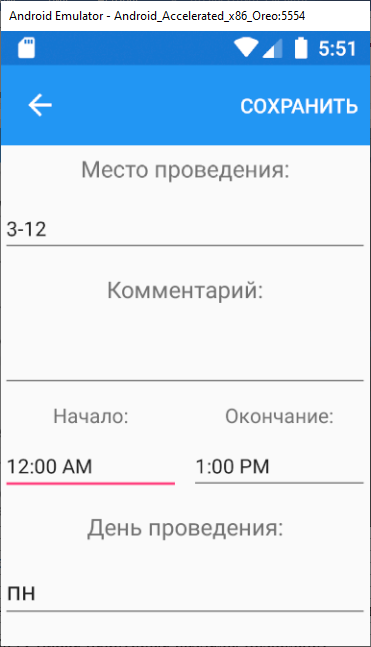
Форма добавления консультации

Рис. П.2.5

Список групп

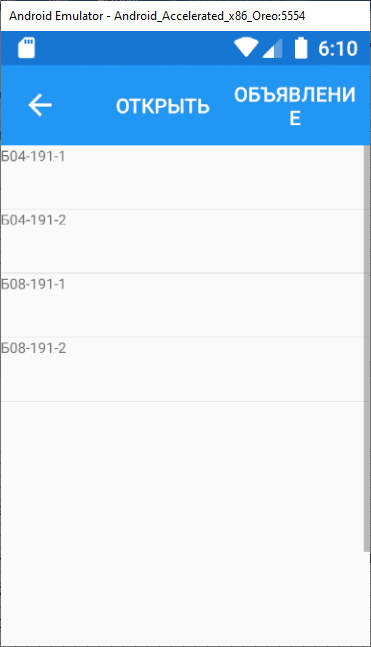


Рис. П.2.7

Список публикаций преподавателя

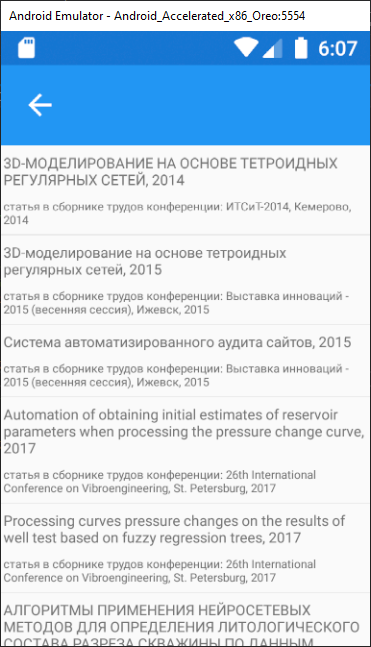


Рис. П.2.6

Список студентов группы

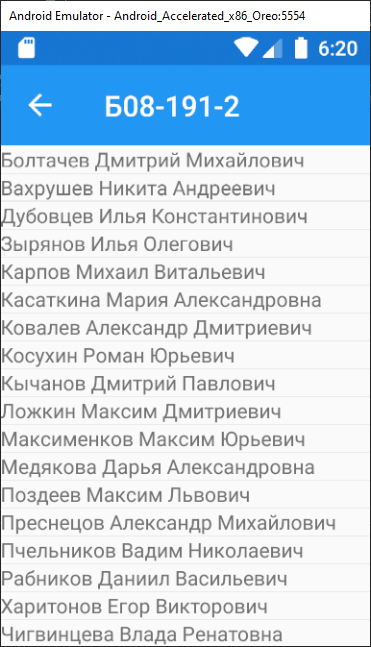


Рис. П.2.8

Чат со студентом

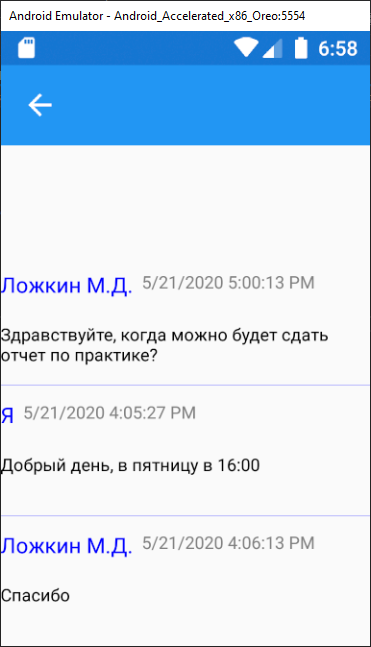


Рис. П.2.9

Добавление объявления

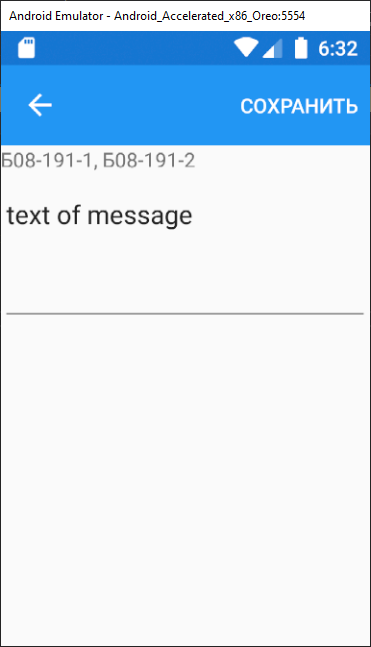


Рис. П.2.10

Объявления для студентов

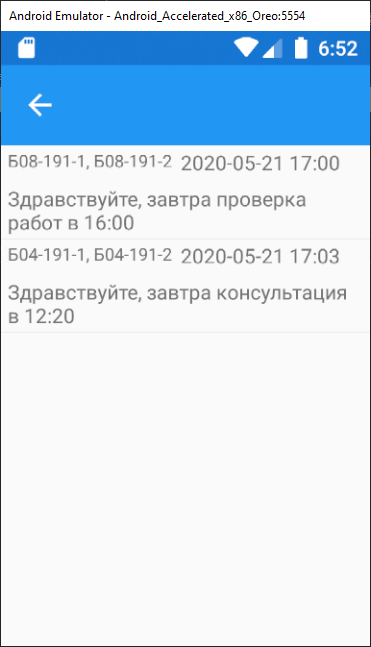


Рис. П.2.11

ПРИЛОЖЕНИЕ 3

# Руководство пользователя приложения «Личный кабинет преподавателя»

П. 3.1. Назначение программы

Часть приложения для преподавателя предназначена для просмотра пользователем основной информации которая может понадобиться в процессе коммуникации со студентами и их обучения. Приложение содержит основные разделы браузерной версии личного кабинета и новую функцию объявлений.

Главное меню состоит из следующих пунктов:

1. информация;
2. консультации
3. публикации;
4. студенты;
5. объявления.

П. 3.2. Условия выполнения программы

Требования к техническому обеспечению устройства пользователя приложения:

* оперативная память не менее 256 МБ;
* не менее 32 МБ свободного пространства на накопителе смартфона;
* приложение клавиатуры или внешняя клавиатура.

Информационная система предъявляет следующие требования к программным средствам.

* операционная система семейства Android или iOS,
* наличие подключения к сети Интернет.

П. 3.3. Пуск программы

Для запуска программы нужно выбрать приложение в меню приложений смартфона нажав соответствующую иконку.

П. 3.4. Команды оператора

Для перехода к главному меню нужно пройти авторизацию, введя логин и пароль, а затем нажать кнопку «Login». Форма авторизации представлена на рис. П.3.1.

Форма авторизации

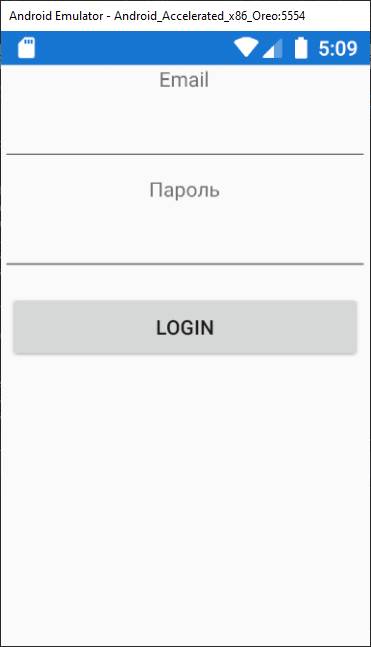


Рис. П.3.1

После авторизации пользователю станет доступно главное меню. Главное меню представлено на рис. П.3.2.

Главное меню

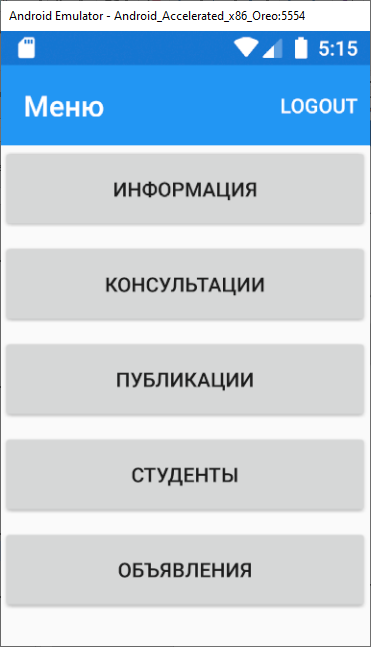


Рис. П.3.2

Для перехода к требуемому разделу, нужно нажать пальцем на соответствующую кнопку в главном меню. Описания разделов представлены в таблице П.3.1.

Таблица П.3.1

|  |  |
| --- | --- |
| Пункт главного меню | Описание раздела |
| Информация | Личная информация пользователя: ФИО, контактные данные, данные об образовании и занимаемых должностях |
| Консультации | Раздел со списком консультаций преподавателя с возможностью добавления и удаления консультаций |
| Публикации | Раздел со списком публикаций (статьи, издания) в принимал участие пользователь |
| Студенты | Список групп студентов которых обучает преподаватель для отправки объявлений и открытия чата |
| Объявления | Объявления отправленные пользователем для одной или нескольких групп студентов |

П. 3.5 Сообщения оператору

Сообщения, выдаваемые оператору, приведены в таблице П.3.2.

Таблица П.3.2

|  |  |
| --- | --- |
| Сообщение выводимое оператору | Действие оператора в случае возникновения сообщения |
| Введен неправильный логин или пароль | Проверить корректность ввода логина и пароля |
| Нет связи с сервером | Проверить подключение к сети интернет |