МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

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

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

Кафедра «Программное обеспечение»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к выпускной квалификационной работе бакалавра на тему:

«Мобильное приложение. «Личный кабинет студента»»

|  |  |
| --- | --- |
| Дипломник  студент группы Б08-191-2 | А.Д. Ковалев |
| Руководитель | Р.В. 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 студента"».

Целью работы было создание подсистемы приложения личного кабинета ИжГТУ для студента.

Для реализации поставленной задачи использовались данные, полученные из материалов по среде разработки .NET, web-технологиям и мобильной разработке на фреймворке Xamarin Forms.

В результате проделанной работы был разработан функционал приложения личного кабинета студента.

## СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc42802238)

[1.АНАЛИЗ РАЗРАБОТКИ СИСТЕМЫ «ЛИЧНЫЙ КАБИНЕТ СТУДЕНТА» 8](#_Toc42802239)

[1.1. Обоснование целесообразности разработки системы 8](#_Toc42802240)

[1.1.1. Назначение системы 8](#_Toc42802241)

[1.1.2. Обоснование цели системы 8](#_Toc42802242)

[1.1.3. Обоснование состава автоматизируемых задач 8](#_Toc42802243)

[1.2. Аналитический обзор 9](#_Toc42802244)

[1.3. Основные требования к системе 11](#_Toc42802245)

[1.3.1. Обоснование состава автоматизируемых задач 11](#_Toc42802246)

[1.3.2. Функциональное назначение системы 11](#_Toc42802247)

[1.3.3. Особенности системы, условия эксплуатации, определяющие основные требования к системе 12](#_Toc42802248)

[1.3.4. Требования к функциональной структуре системы 12](#_Toc42802249)

[1.3.5. Типовые проектные решения и (или) пакеты прикладных программ, применяемых в системе 16](#_Toc42802250)

[1.3.6. Требования к техническому обеспечению 16](#_Toc42802251)

[1.3.7. Требования к программному обеспечению 16](#_Toc42802252)

[1.3.8. Перспективность системы, возможности ее развития 16](#_Toc42802253)

[1.4 Основные технические решения проекта системы 16](#_Toc42802254)

[1.4.2 Описание системы программного обеспечения 18](#_Toc42802255)

[2. Разработка мобильного приложения личного кабинета студента 19](#_Toc42802256)

[2.1. Описание постановки задачи 19](#_Toc42802257)

[2.1.1 Характеристика задачи 19](#_Toc42802258)

[2.2. Алгоритм авторизации и определения роли пользователя 20](#_Toc42802259)

[2.2.1. Назначение и характеристики 20](#_Toc42802260)

[2.2.2. Входная информация 20](#_Toc42802261)

[2.2.3. Выходная информация 20](#_Toc42802262)

[2.2.4. Алгоритм решения 21](#_Toc42802263)

[2.2.5. Описание контрольного примера 23](#_Toc42802264)

[2.2.5.1. Назначение 23](#_Toc42802265)

[2.2.5.2. Исходные данные 23](#_Toc42802266)

[2.2.5.3. Результаты 23](#_Toc42802267)

[2.3 Разработка функционала отображения информационных разделов студента 25](#_Toc42802269)

[2.3.1 Характеристика задачи 25](#_Toc42802270)

[2.3.2. Входная информация 25](#_Toc42802271)

[2.3.3. Выходная информация 26](#_Toc42802272)

[2.3.4. Результаты решения 26](#_Toc42802273)

[2.3.5. Описание контрольного примера 29](#_Toc42802274)

[2.3.5.1. Назначение 29](#_Toc42802275)

[2.3.5.2. Исходные данные 29](#_Toc42802276)

[2.3.5.3. Результаты 29](#_Toc42802277)

[2.3.5.4 Результаты испытания отображения информационных разделов студента 30](#_Toc42802278)

[2.4 Разработка чата с преподавателем 30](#_Toc42802279)

[2.4.1 Характеристика задачи 30](#_Toc42802280)

[2.4.2 Входная информация 31](#_Toc42802281)

[2.4.5 Описание контрольного примера 34](#_Toc42802282)

[2.4.5.1. Назначение 34](#_Toc42802283)

[2.3.5.2. Исходные данные 34](#_Toc42802284)

[2.4.5.3. Результаты 34](#_Toc42802285)

[2.4.5.4 Результаты испытания функционала связи с преподавателем 36](#_Toc42802286)

[ЗАКЛЮЧЕНИЕ 37](#_Toc42802287)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 38](#_Toc42802288)

ПРИЛОЖЕНИЕ 1 [ТЕКСТЫ ПРОГРАММЫ 39](#_Toc42802289)

ПРИЛОЖЕНИЕ 2 [РЕЗУЛЬТЬТАТЫ РАБОТЫ ПРОГРАММЫ 76](#_Toc42802290)

ПРИЛОЖЕНИЕ 3 [РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПРИЛОЖЕНИЯ 79](#_Toc42802291)

## ВВЕДЕНИЕ

В настоящее время широкой популярностью пользуются мобильные версии известных ресурсов. Это связано с широким распространением мобильных устройств, так называемых смартфонов, под управлением ОС Android или iOS. Данные устройства позволяют быстро получить доступ к ресурсу вдали от стационарного компьютера и удобнее, чем использовать ноутбук, особенно в общественных местах, транспорте, по пути куда-либо, когда нет возможности выйти в интернет с помощью другого устройства, кроме как с помощью смартфона.

Данная тема актуальна и в условиях учебы в университете. Так как сейчас у подавляющего числа студентов и преподавателей имеется смартфон, способный запускать приложения с доступом у сети интернет. Студентам же часто требуется средство связи с преподавателем как для координации проведения обычных занятий и консультаций, так и для онлайн консультаций при выполнении домашних заданий наподобие курсовых и лабораторных работ. Также у университета имеется сайт с функциональным личным кабинетом как для преподавателя, так и для студента. Обе версии имеют свой уникальный функционал по доступу к нужной в учебе или работе информации. Общим разделом для обеих ролей пользователей является внутренний чат сайта, который служит для связи преподавателей и студентов, список пользователей фильтруется в зависимости от роли пользователя.

Данное приложение разрабатывается с целью повышения удобства доступа студента к личной и другой полезной в учебе информации, такой как список контактов преподавателей, зачетная книжка и прочее, а также связь с преподавателем через чат.

Новизна данной работы заключается в объединении функционала приложения для студента и преподавателя в одном приложении, при том, что пользователю дается лишь функционал, определенный его ролью.

Приложение использует стандартные, а также специально разработанные для него запросы к серверу ИжГТУ, а потому является абсолютно безопасным как для сервера ИжГТУ, так и для конечного пользователя приложения, так как все выполняемые запросы заранее прописаны в коде приложения и согласованы с разработчиками настольной версии сайта.

В ходе работы изучен фреймворк для мобильной разработки Xamarin Forms, закреплен опыт распределенной разработки с использованием системы контроля версий Git и приложением GitHub Desktop.

## 1. АНАЛИЗ РАЗРАБОТКИ СИСТЕМЫ «Личный кабинет студента»

### 1.1. Обоснование целесообразности разработки системы

### 1.1.1. Назначение системы

Система должна предоставлять студенту удобный доступ к наиболее часто запрашиваемой информации с сайта ИжГТУ, которая может понадобиться при оформлении некоторых документов или учебе, такой как личная информация о студенте, преподавательский состав и оценки за сессии.

### 1.1.2. Обоснование цели системы

В последнее десятилетие большими темпами идет рост рынка мобильных приложений. Это стало возможным благодаря росту числа владельцев смартфонов (за прошлый год отгружено около 1миллиарда смартфонов), повышению ценности быстрого доступа к информации в условиях, когда воспользоваться настольным компьютером не представляется возможным.

Так же для упрощения ведения поддержки и развития отдельных программ и ресурсов популярно разделение клиентов сайтов на приложение для персонального компьютера и мобильное приложение. Примерами такого разделения могут послужить сервис YouTube и социальная сеть «Вконтакте».

В последнее время, особенно для работающих студентов, актуально получать информацию об оценках, расписании или контактах преподавателей как можно быстрее и порой в пути между работой, местом учебы и/или домом. Это позволяет экономить время и силы для других задач студента, улучшить координацию студента и преподавателя, снизить нагрузку на деканат от вопросов, которые регулярно задают студенты.

### 1.1.3. Обоснование состава автоматизируемых задач

Главной задачей системы является обеспечение быстрого и удобного доступа к информации, необходимой студенту во время учебы, связанной с основными данными об успеваемости, преподавателях, данными об оплате учебы в университете и связи с преподавателями. Мобильное приложение же переносит функционал сайта в мобильную версию, тем самым

### 1.2. Аналитический обзор

Область мобильной разработки активно развивается вместе с рынком мобильных устройств[1]. В 2018 году рост рынка мобильных устройств составил 35% в 2018 году, а рынок мобильных приложений составил 101 миллиард долларов США, что составило 75% рост рынка по сравнению с 2016 годом. Более 80% компаний, вышедших на IPO в США в 2018 году, сосредоточены на мобильной разработке.

Не стоит забывать так же о необходимости доступа к информации у преподавателей. При разработке приложения данное требование было учтено, поэтому пользователь, имеющий учетную запись преподавателя, может войти как преподаватель и пользоваться уже функционалом преподавателя

Мобильные приложения созданы у таких университетов, как УрФУ, ИТМО и БГУ. Они также реализуют функционал личного кабинета студента, как и разрабатываемое приложение. Приложение УрФУ имеет систему микротранзакций – денежных платежей, в обмен на которые пользователь получает возможность использования премиум доступа к приложению, который заключается в загрузке и хранении некоторых данных на устройстве пользователя и использованием хранимых данных при отсутствии связи с сервером. Однако аналоги не реализуют функционала преподавателей и, соответственно, не имеют разделения функционала между студентом и преподавателем, что и является отличительной особенностью системы.

Сравнение аналогов приложения представлено в таблице 1.1

Таблица 1.1

Сравнение аналогов приложения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Приложение | Университет | Функционал студента | Наличие функционала преподавателя | Наличие  микротранзакций |
| Личный кабинет студента БГМУ | БГМУ | Просмотр объявлений и обратная связь к ним | - | - |
| iStudent.UrFU | УрФУ | Расписание, оценки, заметки | - | + |
| Личный кабинет студента СВФУ | СВФУ | Расписание, личная информация, оценки | - | - |
| ISTU Mobile | ИжГТУ | Личная информация, оценки, список преподавателей, чат с преподавателями, объявления | + | - |

Как видно из таблицы, ключевым отличием разрабатываемого приложения от аналогов является наличие в том же приложении функционала для преподавателей, для которого и был создан способ единой авторизации и автоматического определения ролей. Также плюсом в сравнении со многими аналогами является и чат между студентом и преподавателем, которого нет в найденных аналогах. Его создание стало возможным благодаря тесному сотрудничеству с разработчиками сайта ИжГТУ, позволившим с некоторыми ограничениями не только просматривать, но и загружать информацию на сервер. Отдельно стоит отметить, что все решения, представленные в таблице, работают только внутри университета, для которого были разработаны, а единого приложения, которое бы поддерживало связь с любым из учебных заведений пользователя пока не существует, что связано с различиями в разработках своих сайтов внутри каждого университета. Поэтому из представленных в таблице приложений с университетом ИжГТУ работает только разрабатываемое приложение.

### 1.3. Основные требования к системе

### 1.3.1. Обоснование состава автоматизируемых задач

Основной целью создания системы является предоставление доступа к наиболее часто запрашиваемой информации из личного кабинета ИжГТУ.

В связи с этим требуется выполнять следующие требования:

* приложение должно иметь раздельный функционал для студента и преподавателя;
* доступ к информации должен быть осуществлен через API сайта ИжГТУ и только через согласованные с разработчиком сайта запросы для сохранения информационной безопасности как сайта, так и пользователя;
* архитектура системы должна позволять легко добавлять новый функционал по необходимости другими разработчиками, знакомыми с технологией Xamarin Forms;
* итоговые данные должны отображаться в удобном для восприятия формате строго по той информации, которая доступна текущему авторизованному пользователю.

### 1.3.2. Функциональное назначение системы

В составе задач автоматизации входят следующие пункты:

* вывод Личной информации о студенте(-ах);
* вывод результатов пройденных сессий;
* информация о преподавателях;
* информация о задолженности и оплата образовательных услуг.

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

Разрабатываемая система будет эксплуатироваться на личном смартфоне пользователя, но приложение не требует больших вычислительных ресурсов, поэтому оно способно работать и на бюджетных устройствах. Также требуется создать установочный файл, который будет устанавливать систему и, если требуется, другое необходимое ПО.

### 1.3.4. Требования к функциональной структуре системы

Система предназначена для решения следующих задач:

1. повышение удобства работы с личной информацией

* входные данные: набор параметров для авторизации (логин и пароль личного кабинета);
* выходные данные: информация о студенте;

1. информирование о результатах сессии;

* входные данные: набор параметров для авторизации (логин и пароль личного кабинета);
* выходные данные: список предметов и оценок по ним у студента, сгруппированный по учебному году и семестру:
  + 1. название предмета;
    2. форма контроля;
    3. оценка при наличии;

1. информация о преподавателях.

* входные данные: набор параметров для авторизации (логин и пароль личного кабинета);
* выходные данные: список преподавателей, ведущих предметы студента:
  + 1. ФИО;
    2. Контакты;
    3. Фото при наличии;

1. Чат с преподавателем:
2. входные данные: выбранный преподаватель, текст отправляемого сообщения
3. выходные данные: сообщения из чата с преподавателем.
4. информация о задолженностях оплаты и оплата

* входные данные: набор параметров для авторизации (логин и пароль личного кабинета);
* выходные данные: сумма задолженности при наличии и информация о способах оплаты, qr-код для оплаты через приложение мобильного банка

Связь модулей программы представлена на рис. 1.1

Связь модулей программы

Рис.1.1

Как видно из представленного рисунка, модули преподавателя и студента отделены друг от друга, общие модули, классы и модели находятся в отдельном пространстве. Так, в модуле общего функционала находятся окно авторизации и чат между преподавателем и студентом, чьи реализации работают равно как от лица преподавателя, так и студента. В отдельный функционал студента отведен такой функционал как просмотр личной информации обучающегося, результатов прошедших и текущих сессий, объявлений от преподавателей, информации о преподавателях и оплаты услуг. Модели данных и функционал, используемый при работе с приложением только от лица преподавателя, был выделен в отдельный модуль и в ходе данной работы затронут не был.

Информация, необходимая для работы независимо от роли пользователя или используемого в данный момент функционала выделена в отдельный статический класс GlobalVariables и включает в себя информацию о пользователе, необходимую при построении HTTP запросов к сайту для выбора нужной информации о пользователе на сервере и подкласс UserAuthrization, хранящий в себе такие данные, как сгенерированный сервером электронный ключ и идентификатор пользователя, используемые в запросах к серверу для безопасности.

Методы, используемые при получении данных объединены в классе StudentGlobalMethods и используются страницами функционала студента для получения данных от сервера.

Архитектура рассчитана на

Диаграмма классов, предоставляющая информацию о зависимости классов и типов, представлена на рис. 1.2

Диаграмма классов

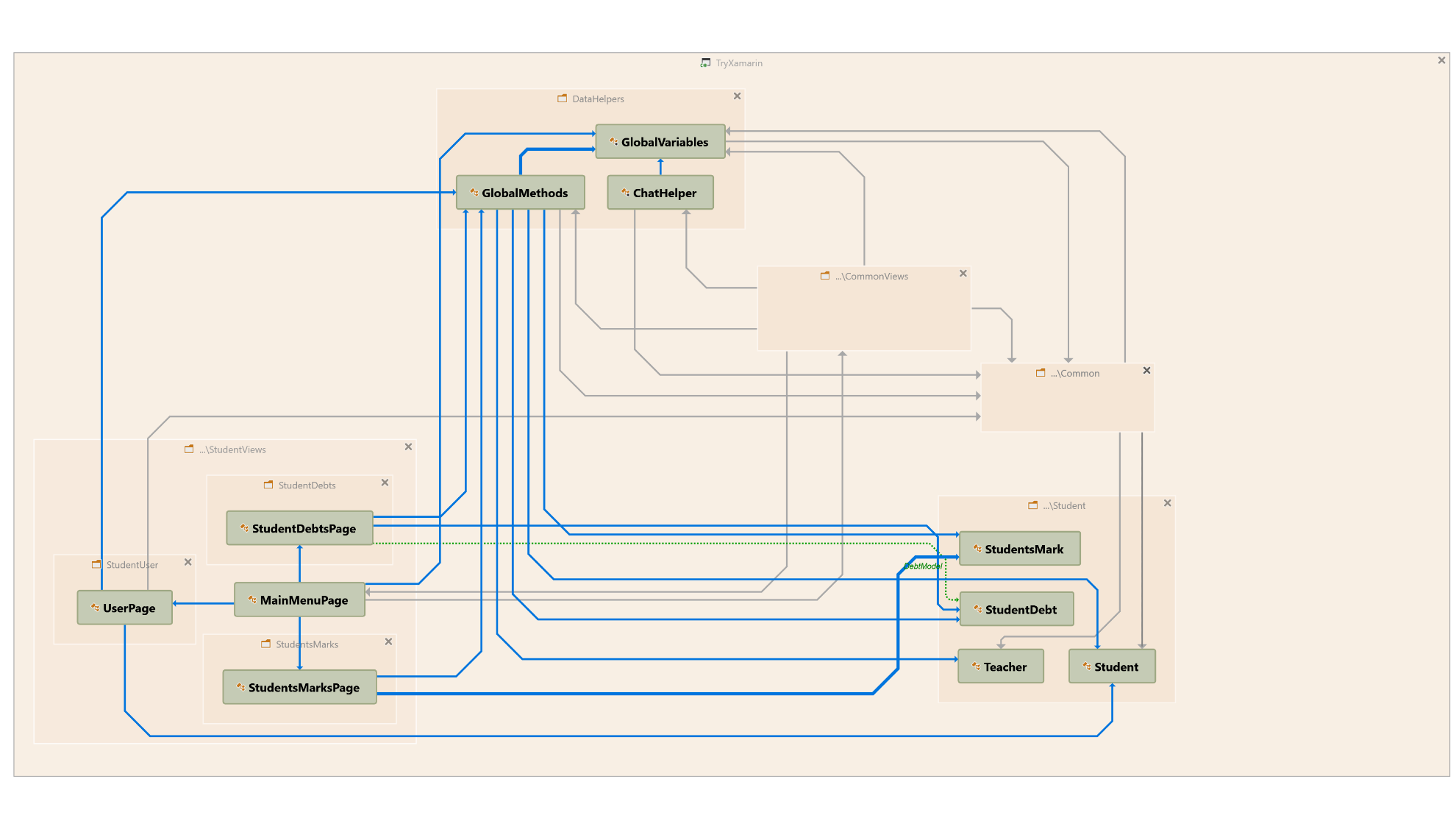


Рис.1.2

Как видно из автоматически сгенерированной средствами расширения Resharper, классы, реализующие функционал студента

### 1.3.5. Типовые проектные решения и (или) пакеты прикладных программ, применяемых в системе

В проекте применяется среда разработки языка C# Visual Studio 2017 с установленным пакетом для разработки Xamarin Forms, а также надстройка ReSharper. Для отладки приложения без отрыва от компьютера разработчика был использован Microsoft Android Emulator, входящий в пакет мобильной разработки Visual Studio.

### 1.3.6. Требования к техническому обеспечению

Требования к техническому обеспечению рабочего места пользователя системы:

1. смартфон под управлением операционной системы Android 5.0.0, iOS 6 или старше,
2. доступ смартфона к сети интернет

### 1.3.7. Требования к программному обеспечению

На смартфоне должно быть установлено следующее программное обеспечение:

1. Android 5.0.0, iOS 6 или старшеl;
2. Google Play для установки.

### 1.3.8. Перспективность системы, возможности ее развития

В перспективе система может дополняться различным функционалом, таким как, объявления от преподавателей, просмотр домашних заданий и прогресса их выполнения. Используемый язык достаточно популярен и прост в освоении, что позволяет достаточно просто найти разработчика для поддержки и модификации даже средствами отдельных способных студентов, если они имеют доступ к исходному коду систем ИжГТУ.

### 1.4 Основные технические решения проекта системы

Все данные приложение получает от сервера, так что на устройстве сохраняются лишь данные для авторизации. На самом устройстве сохраняется лишь пароль и логин при необходимости.

Для разработки данного приложения из нескольких вариантов средств для мобильной разработки выбран набор инструментов Visual Studio для мобильной разработки на платформе .NET и языке программирования C#. Хоть данная среда и не является единственным средством кроссплатформенной разработки, она имеет необходимый функционал для использования в самостоятельном приложении, она, однако, имеет ряд преимуществ, из-за которых и была выбрана в качестве основного средства разработки:

* работа внутри среды разработки Visual Studio, одной из наиболее популярных, а значит, хорошо знакомой многим разработчикам, и имеющей удобный инструментарий как для загрузки дополнительных пакетов инструментов и библиотек, так и для непосредственного процесса разработки и отладки, внутренний инструмент для работы с популярными системами контроля версий (Git, Team Foundation Server), а также возможность устанавливать сторонние модификации для повышения удобства работы, такие как разные версии расширения Решарпер (Resharper) от компании Джетбрэйнс (JetBrains);
* возможность сборки приложений одновременно для основных мобильных платформ (Android, iOS, Windows);
* возможность разработки мобильных приложений любой сложности,

Также отдельным преимуществом при выборе Xamarin было то, что люди, участвующие в разработке, имеют наибольший опыт в работе с языком C# и платформе .NET, на которой и работает библиотека Xamarin Forms. Дополнительным преимуществом является высокая популярность языка C# в Ижевске среди как разработчиков, так и студентов, благодаря чему поиск разработчиков для развития и поддержки приложения значительно упрощается.

Сравнение языков мобильной разработки представлено в таблице 1.2.

Таблица 1.2

Сравнение средств разработки мобильных приложений

|  |  |  |  |
| --- | --- | --- | --- |
| Язык | Основная среда разработки | Платформы | Основные категории приложений |
| Java | Android Studio | Android, Android Wear | Универсальные |
| Swift | XCode | iOS, macOS, tvOS | Универсальные |
| HTML5/JavaScript | Eclipse, Notepad++ | iOS, Android, Tizen | Расширения для веб-страниц |
| Xamarin forms/ C# | Vishal Studio | iOS, Android, Windows 10 | Универсальные |
| Objective-C | XCode | iOS, macOS, tvOS | Универсальные |

### Описание системы программного обеспечения

Для реализации и функционирования проекта необходимо наличие: смартфона под управлением операционной системы Android 5.0.0, iOS 6 или старше с доступом к сети Интернет через сеть Wi-Fi или через оператора мобильной связи.

Для разработки данной системы были использованы:

1. Microsoft Visual Studio 2017;
2. #ReSharper;
3. инструменты для разработки мобильных приложений Xamarin Forms[2],
4. Microsoft Android Emulator.

## 2. РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ЛИЧНОГО КАБИНЕТА СТУДЕНТА

### 2.1. Описание постановки задачи

### 2.1.1 Характеристика задачи

Задача – разработать мобильное приложение личного кабинета студента ИжГТУ.

Приложение разрабатывается с целью повышения удобства доступа студента к функционалу сайта ИжГТУ. Для унификации было решено разработать единое приложение для студентов и преподавателей с разделением функционала в зависимости от роли пользователя (студент или преподаватель).

При разработке поставлены следующие задачи:

* разработать алгоритм единой авторизации, который должен определить роль пользователя и открыть соответствующий роли функционал;
* разработать функционал отображения информационных разделов студента;
* разработать внутри приложения чат с преподавателями, добавив возможность просмотра информации по отдельному преподавателю.

Переходы между экранными формами должны быть интуитивно понятными для пользователя. Диаграмме состояний, описывающая все возможные переходы между экранными формами в части приложения, описывающей функционал студента, представлена на рис. 2.1.

Диаграмма состояний

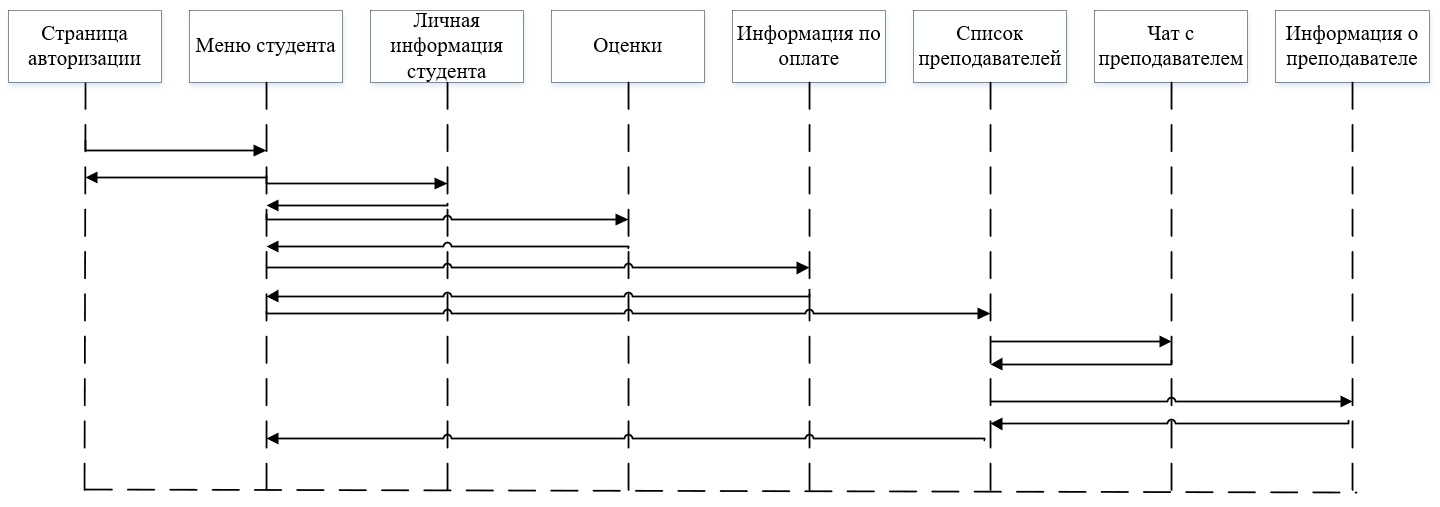


Рис. 2.1

Как показывает рисунок, из страницы авторизации открывается меню студента, откуда открыт доступ к просмотру личной информации, оценкам и информации об оплате студента. После идет страница со списком преподавателей студента, откуда можно перейти к диалогу с преподавателем или просмотреть информацию о преподавателе, которая может потребоваться студенту. Из страниц, представляющих собой пункты меню, можно вернуться обратно в меню, а из самого меню при необходимости можно вернуться на страницу авторизации. Данный функционал обеспечивается классом Navigation, представляющим из себя стек из страниц, которые последовательно кладутся в коллекцию операцией Navigation.Push(Page) и забираются из стека операцией Navigation.Pop(). Для пользователя всегда отображается страница, находящаяся в “вершине” данного стека.

Во избежание случайного выхода из меню на страницу авторизации из-за нажатия кнопки “назад” принято решение не использовать класс Navigation при переходе в меню из страницы авторизации, а переопределять отображаемую страницу MainPage напрямую при помощи присваивания. Для того, чтобы оставить возможность выйти из учетной записи, в главном меню добавлена кнопка Logout, которая выводит пользователю страницу авторизации и выходит из учетной записи.

### 2.1.2 Входная информация

Перечень данных, получаемых от пользователя во время работы:

* логин пользователя на сайте ИжГТУ;
* пароль;
* сообщения, отправляемые преподавателям в чатах;
* выбор соответствующих разделов приложения пользователем.

### 2.1.3 Выходная информация

Перечень информации, получаемая студентом при использовании приложения:

* основная личная информация пользователя;
* сгруппированная по учебному году и семестру информация о результатах прохождения сессии;
* информация о задолженностях об оплате и состоянии счета для студентов, обучающихся на платной основе;
* объявления, оставляемые преподавателем через версию приложения для преподавателя;
* контактные данные преподавателей;
* личные сообщения от преподавателей.

### 2.2. Алгоритм авторизации и определения роли пользователя

### 2.2.1. Назначение и характеристики

Алгоритм должен обеспечивать единую авторизацию для преподавателей и студентов, при этом определяя роль пользователя и функционал, соответствующий роли.

### 2.2.2. Входная информация

Для начала работы алгоритма авторизации от пользователя требуются следующие данные:

* логин на сайте ИжГТУ,
* пароль.

### 2.2.3. Выходная информация

В случае успешной авторизации приложение определяет роль пользователя (студент или преподаватель) и открывает соответствующий функционал

### 2.2.4. Алгоритм решения

Получая на вход от пользователя логин и пароль, алгоритм отправляет HTTP запрос на API сайта с логином м паролем в качестве параметров. В случае неудачи аутентификации сервер возвратит сообщение с кодом ошибки. В случае успешной аутентификации сервер возвращает данные авторизации, содержащие токен, используемый в целях безопасности как заголовок запросов на получение защищенной информации отдельного пользователя. Используя полученный токен, алгоритм отправляет серверу запрос на получение информации о пользователе. В ответ сервер отправляет информацию о пользователе, в том числе логическое поле IsPPS, определяющий, является ли данный пользователь преподавателем. Данная информация записывается в глобальную переменную CurrentUser. В зависимости от признака IsPPS приложение определяет роль пользователя и далее открывает функционал, соответствующий роли пользователя.

Список использованных в рамках реализации задачи функций:

* Authorize (email, password) – метод, отправляющий на сервер запрос об аутентификации, включающий в себя логин и пароль пользователя. В зависимости от успеха аутентификации возвращает сгенерированный на сервере токен и идентификатор пользователя или сообщение об ошибке.
* GetUser() – метод, отправляющий запрос на получение информации о пользователе, отправляющий на сервер соответствующий HTTP запрос с полученным в методе Authorize и идентификатором пользователя и получающий в ответ информацию о пользователе, позволяющую определить роль пользователя.

Логика алгоритма представлена в виде схемы графически на рис. 2.2

Схема алгоритма авторизации



Рис. 2.2

### 2.2.5. Описание контрольного примера

### 2.2.5.1. Назначение

Система призвана получить данные от пользователя и открыть функционал, соответственно его роли.

### 2.2.5.2. Исходные данные

Будут указаны 3 примера: некорректные логин и пароль, авторизация студента и авторизация тестовой учетной записи преподавателя.

### 2.2.5.3. Результаты

Входные данные и результаты работы алгоритма представлены на рис. 2. 3 – 2.7

Ввод неверных данных авторизации

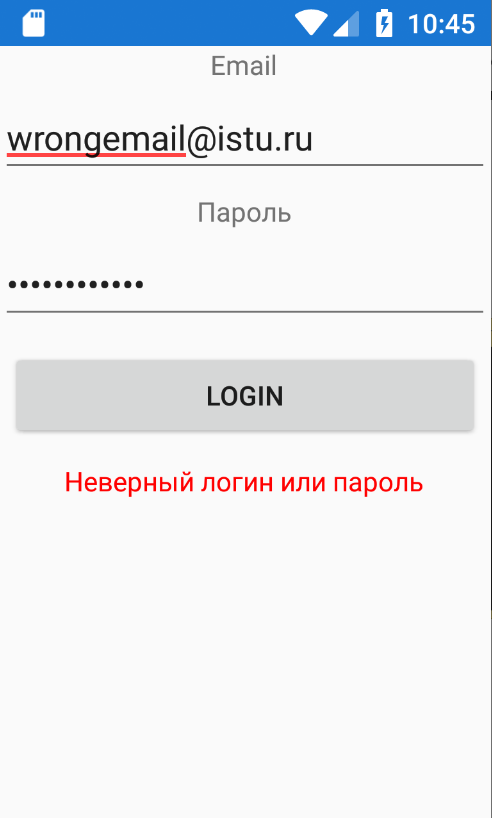


Рис. 2.3

Ввод данных студента Меню студента

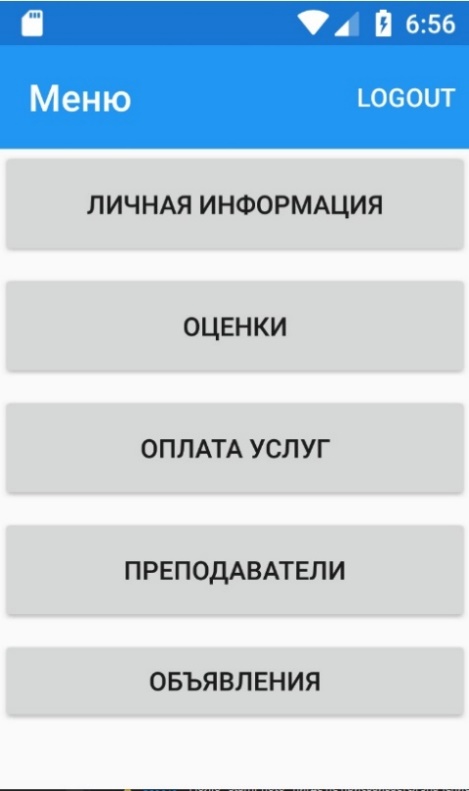
 

Рис. 2.4 Рис.2.5

Ввод тестовых

данных преподавателя Меню преподавателя

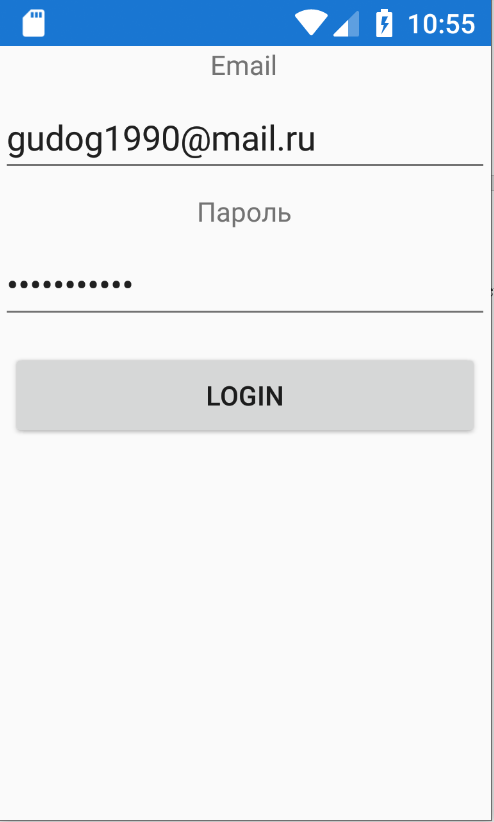
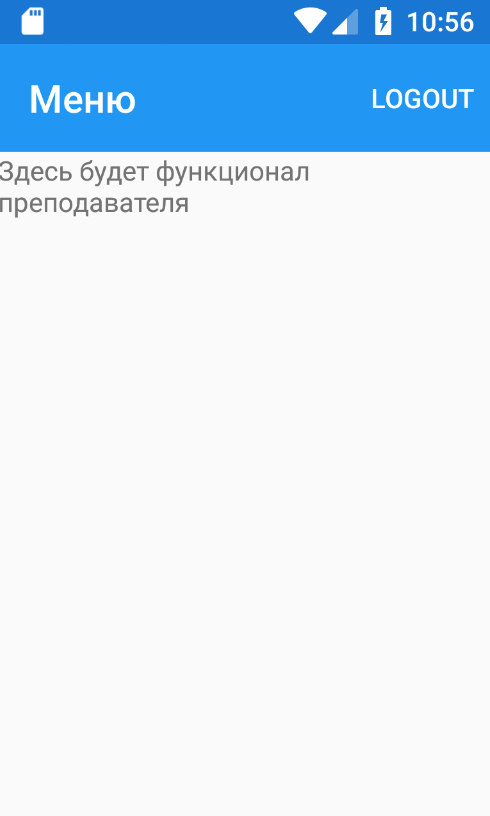
 

Рис. 2.6 Рис. 2.7

### 2.2.5.4 Результаты испытания алгоритма единой авторизации.

Как видно из скриншотов выше, алгоритм авторизации выполняет свою задачу и верно определяет роль пользователя и предоставляет соответствующий функционал.

### 2.3 Разработка функционала отображения информационных разделов студента

### 2.3.1 Характеристика задачи

Задача – разработать способ получения данных по студенту и формы отображения получаемой информации. В случаях, когда информация представлена в виде списка, логически сгруппировать ее.

### 2.3.2. Входная информация

Авторизованный пользователь должен выбрать нужный ему раздел и нажать соответствующую кнопку.

Информация, отображаемая на страницах, запрашивается приложением у сервера с помощью HTTP запросов, для каждого вида информации имеется свой запрос.

Ответ от сервера приходит в виде строки в формате JSON, описывающей модель отображаемых данных, после чего приложение проводит десериализацию и записывает в оперативную память как экземпляр соответствующего класса.

При формировании запроса к серверу приложение использует полученные на этапе авторизации токен и идентификатор пользователя. В зависимости от запроса к URL запроса добавляется строка, вызывающая контроллер.

В разделе оценок студенту также необходимо выбрать интересующие его учебный год и семестр, чтобы отобразить список предметов и оценки по ним.

### 2.3.3. Выходная информация

В зависимости от выбранной страницы открывается соответствующая страница с нужной информацией, дублирующей информацию с сайта ИжГТУ.

В разделе оценок информация сгруппирована по учебному году и семестру и отображается сессия, соответствующая году и семестру.

После выбора пользователем нужного учебного года и семестра список заполняется оценками и предметами в соответствии с выбранными годом и семестром.

### 2.3.4. Результаты решения

Каждая страница содержит уникальную для приложения разметку с соответствующими полями и экземпляр класса, описывающий те данные, которые приходят от сервера на соответствующий запрос. После получения ответа от сервера и инициализации данных происходит заполнение соответствующих элементов разметки, после чего пользователь может видеть всю нужную информацию.

Список использованных функций:

* GetUser() – получить информацию о пользователе;
* GetSession() – получение информации об оценках пользователя-студента посредством HTTP запроса с идентификатором студента в качестве параметра;
* GetDebts() – получение информации по задолженностям по оплате за обучение посредством HTTP запроса с идентификатором пользователя в качестве параметра;
* GetTeachers() – получение списка преподавателей посредством HTTP запроса с идентификатором студента в качестве параметра.

Логика алгоритмов представлена в виде схем на рис. 2.8 – 2.11.

Алгоритм загрузки личной Алгоритм загрузки

информации информации об оплате

Рис. 2.8 Рис. 2.9

Алгоритм загрузки информации Алгоритм загрузки

о результатах сессий информации о преподавателях

Рис. 2.10 Рис. 2.11

### 2.3.5. Описание контрольного примера

### 2.3.5.1. Назначение

Система призвана получить и отобразить данные, выбранные пользователем.

### 2.3.5.2. Исходные данные

Будут указаны примеры заполненных страниц, выбранных пользователем.

### 2.3.5.3. Результаты

Выходные данные представлены в виде снимков экрана рис. 2.12 – 2.14

Личная информация Оценки

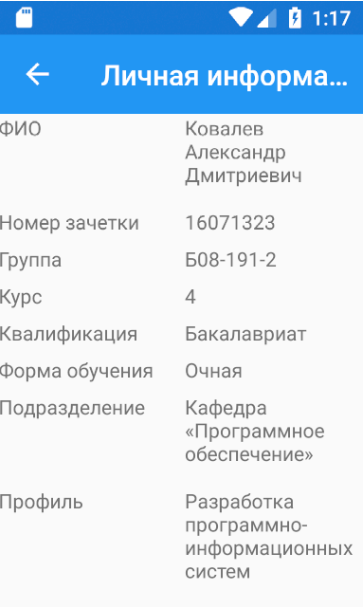
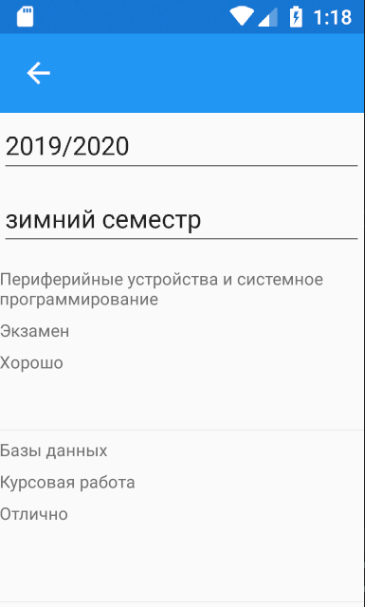
 

Рис. 2.12 Рис. 2.13

Оплата за обучение

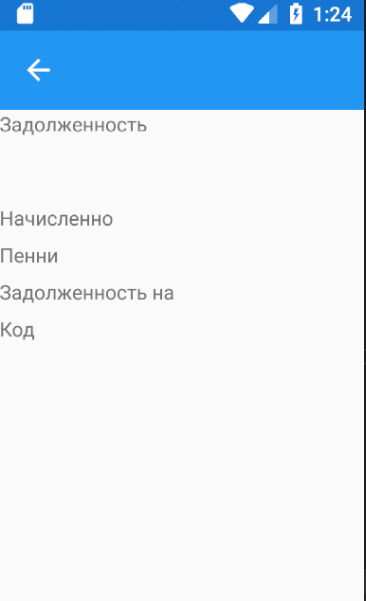


Рис. 2.14

### 2.3.5.4 Результаты испытания отображения информационных разделов студента

Как видно из скриншотов, пользователь студент может просматривать нужную информацию, хранимую о нем на сайте. Архитектура приложения же позволяет быстро добавить новый информационный раздел или модифицировать существующий.

### 2.4 Разработка чата с преподавателем

### 2.4.1 Характеристика задачи

Задача – разработка функционала связи с преподавателем с помощью чата внутри приложения. Приложение должно предоставлять пользователю список преподавателей, которые ведут у него занятие в данный момент или выставляли оценку студентам его группы ранее. Это сделано для того, чтобы студенты, имеющие академические задолженности также могли связаться с преподавателем и закрыть долг.

Так же требуется добавить возможность отображения дополнительной информации о преподавателе, такой как должность, кафедру и контактные данные.

### 2.4.2 Входная информация

Для выбора списка преподавателей пользователю требуется нажать на соответствующую кнопку, после чего выбрать нужного в данный момент преподавателя, нажав на его имя в списке. В процессе общения текст сообщения вводится посредством стандартной клавиатуры смартфона.

Для просмотра информации по отдельному преподавателю от пользователя требуется долговременно удержать строку, соответствующую выбранному преподавателю.

2.4.3 Выходная информация

После выбора раздела приложение предоставляет список преподавателей и количество непрочитанных сообщений в каждом из диалогов.

После выбора нужного чата открывается история сообщений с преподавателем и открывается возможность для ввода и отправки сообщения.

При продолжительном удержании кнопки преподавателя в списке открывается информация о преподавателе, иногда нужная студенту, такая как кафедра, должность, контактные данные.

2.4.4 Описание решения

При открытии страницы с преподавателями приложение делает HTTP запрос на получение списка преподавателей, после чего формирует таблицу-список, отображаемые для пользователя. Также каждая строка дополняется числом непрочитанных студентом сообщений от преподавателя и последним сообщением в чате, при наличии.

При выборе преподавателя приложение получает посредством HTTP запроса историю сообщений между студентом и преподавателем. После отображения сообщений на сервер отправляется сигнал о том, что непрочитанные сообщения теперь прочитаны.

Ввод сообщений осуществляется в текстовое поле внизу экрана. При нажатии на кнопку отправить приложение отправляет HTTP запрос на запись сообщения.

При долгом нажатии на имя преподавателя в списке, приложение отправляет запрос на получение информации о преподавателе, после чего заполняет форму информации и открывает ее пользователю.

При возврате к странице списка преподавателей информация о диалогах обновляется с помощью нового HTTP запроса к серверу, после чего пользователь может выбрать другой диалог, просмотреть информацию о преподавателях или выйти в главное меню.

Список методов, используемых в решении:

* GetTeachers() – метод, использующий записанный в объекте CurrentUser идентификатор студента и получающий от сервера список преподавателей,
* GetDialog(int teacherId) – метод, принимающий идентификатор преподавателя и получающий от сервера историю диалога между преподавателем и студентом,
* GetTeacherInfo(int teacherId) – метод, принимающий идентификатор преподавателя и получающий информацию о преподавателе, открытую для студентов.

Логика открытия страниц раздела представлена на рис. 2.15

Логика открытия страниц раздела



Рис. 2.15

### 2.4.5 Описание контрольного примера

### 2.4.5.1. Назначение

Система должна предоставить пользователю список преподавателей, отобразить информацию по отдельным преподавателям и обеспечить общение студента и преподавателя посредством чата

### 2.3.5.2. Исходные данные

В качестве исходных данных рассматриваются следующие сценарии:

* открытие списка преподавателей, ставивших оценки группе студента
* открытие диалога с преподавателем, отправка и получение сообщений,
* открытие информации о преподавателе для студента в объеме, требуемом для обучения.

### 2.4.5.3. Результаты

В качестве контрольного примера используются все варианты работы пользователя с данным разделом приложения:

1. просмотр списка преподавателей, выставлявших оценки или ведущих занятия у студента,
2. просмотр истории диалога с преподавателем после нажатия на соответствующую выбранному преподавателю строку, отправка и получение сообщений,
3. просмотр информации по отдельному преподавателю после продолжительного нажатия на соответствующую строку.

Результаты работы решения представлены на рис. 2.18-2.20.

Список диалогов Диалог с преподавателем

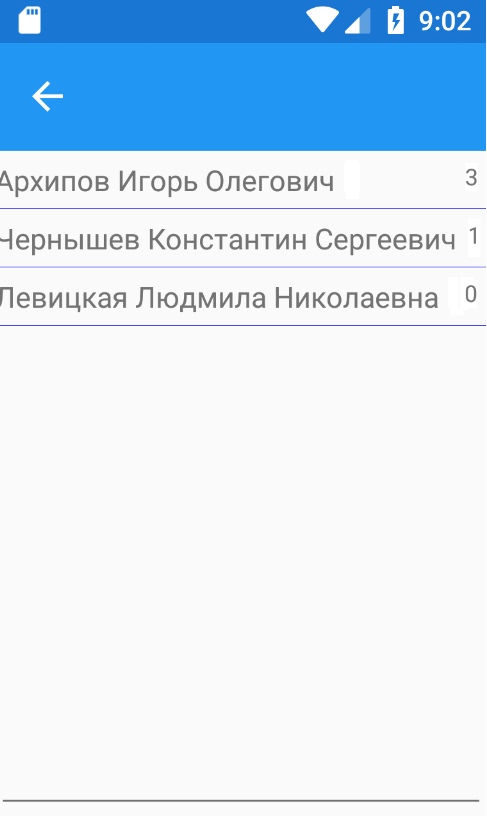
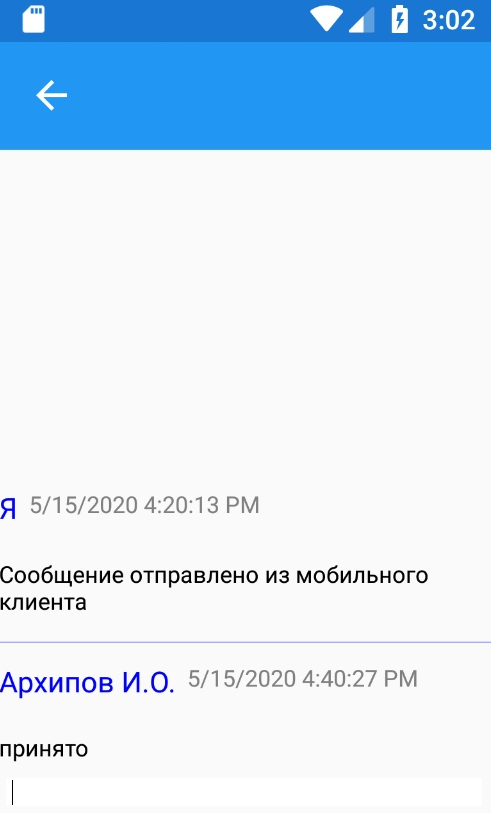
 

Рис. 2.18 Рис. 2.19

Прототип формы информации о преподавателе

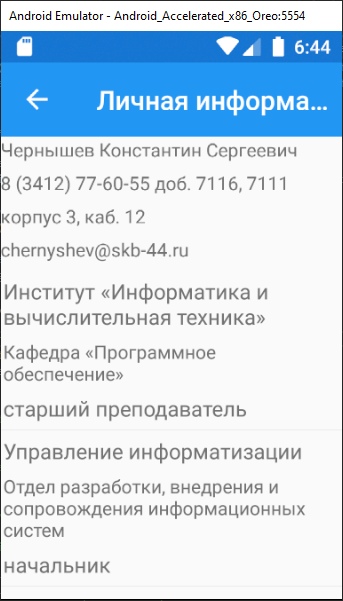


Рис. 2.20

### 2.4.5.4 Результаты испытания функционала связи с преподавателем

Как видно из представленных выше скриншотов, весь функционал связи с преподавателем работает как ожидается. Логика открытия страниц действует так, как от нее требуется, есть возможность просматривать и отправлять сообщения, просматривать информацию о преподавателях.

## ЗАКЛЮЧЕНИЕ

В ходе работы было создана часть приложения для работы в личном кабинете студента ИжГТУ.

Итогом работы является приложение, которое обеспечивает удобный доступ к основному функционалу сайта, добавлен новый функционал объявлений как на серверной части, так и на клиентском приложении. На данном этапе разработки приложение позволяет повысить удобство пользования сайтом ИжГТУ в следующих вариантах использования:

* просмотр информации, необходимой во время учебы;
* просмотр объявлений преподавателей о консультациях, переносах занятий и пр.;
* связь с преподавателем через внутренний чат.

Архитектура приложения позволяет наращивать функционал по мере добавления требований, а язык C#, достаточно популярный среди программистов, позволяет быстро найти разработчиков для выполнения этих доработок даже среди студентов.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Отчет App Annie о рынке мобильных услуг за 2019 год: [Электронный ресурс]. https://qmobi.agency/blog (Дата обращения 30.05.2020),
2. Xamarin Forms руководство: [Электронный ресурс]. https://metanit.com/sharp/xamarin (Дата обращения: 09.04.2020),
3. Разработка под iOS и Android: рейтинг языков программирования: [электронный ресурс] <https://apptractor.ru/razrabotka-pod-ios-i-android-reyting-yazyikov-programmirovaniya> (Дата обращения 05.05.2020),
4. ГОСТ 19.701 – 90 (ИСО 5807 – 85) Схемы алгоритмов, программ, данных и систем. – М.: Издательство стандартов, 1991
5. ГОСТ 19.505 – 79 Единая система программной документации. Руководство оператора. Требования к содержанию и оформлению. – М.: Издательство стандартов, 1988
6. Методические указания по оформлению курсовых и выпускных квалификационных работ для студентов, обучающихся на кафедре «Программное обеспечение». – Ижевск: Издательство ИжГТУ, 2018

ПРИЛОЖЕНИЕ 1

## ТЕКСТЫ ПРОГРАММЫ

AuthorizationPage.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:widget="clr-namespace:Android.Widget;assembly=Mono.Android"

xmlns:controls="clr-namespace:XLabs.Forms.Controls;assembly=XLabs.Forms"

x:Class="TryXamarin.Views.CommonViews.AuthorizationPage" Title="Авторизация">

<ContentPage.ToolbarItems>

<ToolbarItem>

</ToolbarItem>

</ContentPage.ToolbarItems>

<ContentPage.Content>

<StackLayout>

<Label x:Name="EmailLabel"

Text="Email"

VerticalOptions="Center"

HorizontalOptions="CenterAndExpand"/>

<Entry x:Name="EmailEntry"

TextChanged="EmailEntry\_OnTextChanged"/>

<Label x:Name="PasswordLabel" Text="Пароль"

VerticalOptions="Center"

HorizontalOptions="CenterAndExpand" />

<Entry x:Name="PasswordEntry"

TextChanged="PasswordEntry\_OnTextChanged"

IsPassword ="True"/>

<Button Margin="5" Text="Login" Clicked="LoginButton\_OnClicked"></Button>

<Label x:Name="ErrorLabel"

VerticalOptions="Center"

HorizontalOptions="CenterAndExpand" />

<controls:CheckBox DefaultText="Сохранить пароль"></controls:CheckBox>

</StackLayout>

</ContentPage.Content>

</ContentPage>

AuthorizationPage.xaml.cs

using System;

using TryXamarin.DataHelpers;

using TryXamarin.Models.Common;

using TryXamarin.Views.TeacherViews.MainMenu;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace TryXamarin.Views.CommonViews

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class AuthorizationPage : ContentPage

{

private UserAuthorization \_userAuthorization;

public string Email = "16071323@edu.istu.ru";

public string Password = "02.03.1998";

public AuthorizationPage ()

{

\_userAuthorization = GlobalVariables.UserAuthorization;

InitializeComponent ();

}

private void EmailEntry\_OnTextChanged(object sender, TextChangedEventArgs e)

{

Email = EmailEntry.Text;

}

private void PasswordEntry\_OnTextChanged(object sender, TextChangedEventArgs e)

{

Password = PasswordEntry.Text;

}

#region LogIn Buttons

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

{

var email = "16071323@edu.istu.ru";

var password = "02.03.1998";

//var email = "gudog1990@gmail.com";

//var password = "12345678";

LoginAction(email, password);

}

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

{

var email = "gudog1990@gmail.com";

var password = "12345678";

LoginAction(email, password);

}

private void LoginAction(string email, string password)

{

try

{

GlobalMethods.Authorize(email, password);

GlobalVariables.CurrentUser = GlobalMethods.GetUser();

if (!GlobalVariables.CurrentUser.IsPPS)

{

GlobalVariables.CurrentUser.AuthorizationType = AuthorizationType.Student;

App.Current.MainPage = new NavigationPage(new MainMenuPage());

return;

}

else

{

GlobalVariables.CurrentUser.AuthorizationType = AuthorizationType.Teacher;

App.Current.MainPage = new NavigationPage(new TeacherViews.MainMenu.MainMenuPage());

}

throw new ArgumentException();

}

catch (Exception exception)

{

ErrorLabel.TextColor = Color.Red;

ErrorLabel.Text = "Неверный логин или пароль";

}

}

#endregion

}

}

ChatsPage.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.CommonViews.ChatsPage">

<StackLayout>

<ListView x:Name="ChatsListView"

ItemsSource="{Binding Items}"

ItemTapped="Handle\_ItemTapped"

CachingStrategy="RecycleElement">

</ListView>

<Entry></Entry>

</StackLayout>

</ContentPage>

ChatsPage.xaml.cs

using System.Collections.ObjectModel;

using TryXamarin.DataHelpers;

using TryXamarin.Models.Common;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace TryXamarin.Views.CommonViews

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class ChatsPage : ContentPage

{

public ObservableCollection<ChatModel> Abonents { get; set; }

public ChatsPage()

{

InitializeComponent();

FillInfo();

}

public void FillInfo()

{

var a = ChatHelper.GetChatCollection();

Abonents = GlobalVariables.CurrentUser.AuthorizationType == AuthorizationType.Student ? new ObservableCollection<ChatModel>(a.Students) : new ObservableCollection<ChatModel>(a.Teachers);

ChatsListView.ItemsSource = Abonents;

ChatsListView.SeparatorColor = Color.Blue;

ChatsListView.ItemTemplate = new DataTemplate(() =>

{

Label nameLabel = new Label { FontSize = 15 };

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

nameLabel.HorizontalTextAlignment = TextAlignment.Start;

Label messagesLabel = new Label() { FontSize = 12 };

messagesLabel.SetBinding(Label.TextProperty, "UnreadMessagesCount");

messagesLabel.HorizontalTextAlignment = TextAlignment.End;

messagesLabel.HorizontalOptions = LayoutOptions.EndAndExpand;

Label lastMessageLabel = new Label() { FontSize = 12 };

lastMessageLabel.SetBinding(Label.TextProperty, "LatestMessageText");

var viewCell = new ViewCell

{

View = new StackLayout

{

Padding = new Thickness(0, 5),

Orientation = StackOrientation.Vertical,

Children =

{

new StackLayout(){Children={nameLabel, messagesLabel}, Orientation = StackOrientation.Horizontal}, lastMessageLabel

}

}

};

lastMessageLabel.BackgroundColor =

string.IsNullOrEmpty(lastMessageLabel.Text) ? Color.White : Color.AliceBlue;

return viewCell;

});

ChatsListView.RowHeight = 70;

}

async void Handle\_ItemTapped(object sender, ItemTappedEventArgs e)

{

if (e.Item == null)

return;

var chat = (ChatModel) e.Item;

var a = ChatHelper.GetChatMessages(chat.Id);

Navigation.PushAsync(new ChatPage(chat.Id));

//Deselect Item

//((ListView)sender).SelectedItem = null;

}

}

}

ChatPage.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.CommonViews.ChatPage">

<StackLayout>

<ListView x:Name="MessagesListView"

ItemTapped="Handle\_ItemTapped"

VerticalOptions="EndAndExpand"

SeparatorColor="Blue"

CachingStrategy="RecycleElement">

</ListView>

<Entry Height="30"></Entry>

</StackLayout>

</ContentPage>

ChatPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Linq;

using System.Threading.Tasks;

using Android.Views;

using TryXamarin.DataHelpers;

using TryXamarin.Models.Common;

using TryXamarin.Models.Student;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using TextAlignment = Xamarin.Forms.TextAlignment;

namespace TryXamarin.Views.CommonViews

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class ChatPage : ContentPage

{

public ObservableCollection<ChatMessage> Messages { get; set; }

public ChatPage(int chatId)

{

InitializeComponent();

Messages = new ObservableCollection<ChatMessage>(ChatHelper.GetChatMessages(chatId));ChatHelper.GetChatMessages(chatId);

FillInfo();

}

public void FillInfo()

{

//Messages = new ObservableCollection<MessageBasic>(a);

MessagesListView.ItemsSource = Messages;

MessagesListView.SeparatorColor = Color.Blue;

MessagesListView.ItemTemplate = new DataTemplate(() =>

{

Label nameLabel = new Label { FontSize = 15 };

nameLabel.SetBinding(Label.TextProperty, "AuthorName");

nameLabel.HorizontalTextAlignment = TextAlignment.Start;

nameLabel.TextColor = Color.Blue;

Label dateLabel = new Label() { FontSize = 12 };

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

dateLabel.TextColor = Color.Gray;

Label textLabel = new Label() { FontSize = 12 };

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

textLabel.TextColor = Color.Black;

var header = new StackLayout

{

Padding = new Thickness(0, 5),

Orientation = StackOrientation.Horizontal,

};

header.Children.Add(nameLabel);

header.Children.Add(dateLabel);

var body = new StackLayout

{

Padding = new Thickness(0, 5),

Orientation = StackOrientation.Horizontal,

};

body.Children.Add(textLabel);

var viewCell = new ViewCell

{

View = new StackLayout

{

Padding = new Thickness(0, 5),

Orientation = StackOrientation.Vertical,

Children = { header, body }

}

};

return viewCell;

});

MessagesListView.RowHeight = 90;

}

async void Handle\_ItemTapped(object sender, ItemTappedEventArgs e)

{

if (e.Item == null)

return;

await DisplayAlert("Item Tapped", "An item was tapped.", "OK");

//Deselect Item

((ListView)sender).SelectedItem = null;

}

}

}

StudentDebtsPage.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.StudentViews.StudentDebts.StudentDebtsPage">

<ContentPage.Content>

<StackLayout>

<Grid x:Name="UserGrid">

<Grid.RowDefinitions>

<RowDefinition Height="60"/>

<RowDefinition Height="20"/>

<RowDefinition Height="20"/>

<RowDefinition Height="20"/>

<RowDefinition Height="20"/>

<RowDefinition Height="80"/>

<RowDefinition Height="60"/>

<RowDefinition Height="80"/>

<RowDefinition Height="45"/>

<RowDefinition Height="30"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"></ColumnDefinition>

<ColumnDefinition Width="\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Label Grid.Row="0" Grid.Column="0" Text="Задолженность"></Label>

<Label x:Name="AmountLabel" Grid.Row="0" Grid.Column="1"></Label>

<Label Grid.Row="1" Grid.Column="0" Text="Начисленно"></Label>

<Label x:Name="DebtLabel" Grid.Row="1" Grid.Column="1"></Label>

<Label Grid.Row="2" Grid.Column="0" Text="Пенни"></Label>

<Label x:Name="PenaltyLabel" Grid.Row="2" Grid.Column="1"></Label>

<Label Grid.Row="3" Grid.Column="0" Text="Задолженность на \nконец года"></Label>

<Label x:Name="AmountYearLabel" Grid.Row="3" Grid.Column="1"></Label>

<Label Grid.Row="4" Grid.Column="0" Text="Код"></Label>

<Label x:Name="Codelabel" Grid.Row="4" Grid.Column="1"></Label>

<!--<Label Text="Оплатить задолженность" Grid.Row="5"></Label>

<Image x:Name="AmountQR" Grid.Row="6" Grid.Column="0" Grid.ColumnSpan="2"></Image>

<Label Text="Оплатить пенни" Grid.Row="7"></Label>

<Image x:Name="PennyQR" Grid.Row="8" Grid.Column="0" Grid.ColumnSpan="2"></Image>-->

</Grid>

</StackLayout>

</ContentPage.Content>

</ContentPage>

StudentDebts.xaml.cs

using TryXamarin.DataHelpers;

using TryXamarin.Models.Student;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace TryXamarin.Views.StudentViews.StudentDebts

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class StudentDebtsPage : ContentPage

{

private StudentDebt DebtModel;

public StudentDebtsPage()

{

DebtModel = GlobalMethods.GetDebts();

InitializeComponent();

if (DebtModel != null)

FillIndo();

}

private void FillIndo()

{

DebtLabel.Text = DebtModel.Debt.ToString();

AmountLabel.Text = DebtModel.Amount.ToString();

AmountYearLabel.Text = DebtModel.AmountYear.ToString();

PenaltyLabel.Text = DebtModel.Penalty.ToString();

Codelabel.Text = DebtModel.Code;

}

}

}

StudentsMarksPage.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.StudentViews.StudentsMarks.StudentsMarksPage">

<ContentPage.Content>

<StackLayout>

<Picker x:Name="YearPicker" SelectedIndexChanged="YearPicker\_OnSelectedIndexChanged"></Picker>

<Picker x:Name="SemesterPicker" SelectedIndexChanged="SemesterPicker\_OnSelectedIndexChanged"></Picker>

<ListView x:Name="MarksList"></ListView>

</StackLayout>

</ContentPage.Content>

</ContentPage>

StudentsMarksPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using TryXamarin.DataHelpers;

using TryXamarin.Models.Student;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace TryXamarin.Views.StudentViews.StudentsMarks

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class StudentsMarksPage : ContentPage

{

private List<StudentsMark> \_studentMarks;

public StudentsMarksPage ()

{

InitializeComponent ();

\_studentMarks = GlobalMethods.GetSession();

FillInfo();

}

private void FillInfo()

{

YearPicker.ItemsSource = \_studentMarks.OrderBy(x => x.Year).Select(x => x.Year).Distinct().ToList();

YearPicker.SelectedIndex = YearPicker.ItemsSource.Count - 1;

SemesterPicker.ItemsSource = \_studentMarks.OrderBy(x => x.Year).Select(x => x.Term.ToLower()).Distinct().ToList();

SemesterPicker.SelectedIndex = SemesterPicker.ItemsSource.Count - 1;

MarksList.ItemsSource = \_studentMarks.Where(x => x.Year == (string)(YearPicker.SelectedItem));

}

private void YearPicker\_OnSelectedIndexChanged(object sender, EventArgs e)

{

var picker = (Picker) sender;

var marksListItemsSource = \_studentMarks

.Where(x => x.Year == (string)picker.SelectedItem &&

(SemesterPicker.SelectedItem == null || x.Term.ToLower() == SemesterPicker.SelectedItem.ToString().ToLower())).ToList();

MarksList.ItemsSource = marksListItemsSource;

MarksList.ItemTemplate = new DataTemplate(() =>

{

Label subjectLabel = new Label { FontSize = 12 };

subjectLabel.SetBinding(Label.TextProperty, "SubjectName");

Label formControlLabel = new Label() { FontSize = 12 };

formControlLabel.SetBinding(Label.TextProperty, "FormControl");

Label markLabel = new Label() { FontSize = 12 };

markLabel.SetBinding(Label.TextProperty, "Mark");

return new ViewCell

{

View = new StackLayout

{

Padding = new Thickness(0, 5),

Orientation = StackOrientation.Vertical,

Children = { subjectLabel, formControlLabel, markLabel }

}

};

});

MarksList.RowHeight = 120;

}

private void SemesterPicker\_OnSelectedIndexChanged(object sender, EventArgs e)

{

var picker = (Picker)sender;

var marksListItemsSource = \_studentMarks

.Where(sm => sm.Year == YearPicker.SelectedItem.ToString() &&

sm.Term.ToLower() == picker.SelectedItem.ToString()).ToList();

MarksList.ItemsSource = marksListItemsSource;

MarksList.ItemTemplate = new DataTemplate(() =>

{

// привязка к свойству Name

Label subjectLabel = new Label { FontSize = 12 };

subjectLabel.SetBinding(Label.TextProperty, "SubjectName");

// привязка к свойству Company

Label formControlLabel = new Label() { FontSize = 12 };

formControlLabel.SetBinding(Label.TextProperty, "FormControl");

// привязка к свойству Price

Label markLabel = new Label() { FontSize = 12 };

markLabel.SetBinding(Label.TextProperty, "Mark");

// создаем объект ViewCell.

return new ViewCell

{

View = new StackLayout

{

Padding = new Thickness(0, 5),

Orientation = StackOrientation.Vertical,

Children = { subjectLabel, formControlLabel, markLabel }

}

};

});

MarksList.RowHeight = 120;

}

}

}

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"

x:Class="TryXamarin.Views.StudentViews.StudentUser.UserPage">

<ContentPage.Content>

<ScrollView>

<StackLayout>

<Grid x:Name="UserGrid">

<Grid.RowDefinitions>

<RowDefinition Height="60"/>

<RowDefinition Height="20"/>

<RowDefinition Height="20"/>

<RowDefinition Height="20"/>

<RowDefinition Height="20"/>

<RowDefinition Height="20"/>

<RowDefinition Height="60"/>

<RowDefinition Height="90"/>

<RowDefinition Height="45"/>

<RowDefinition Height="30"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"></ColumnDefinition>

<ColumnDefinition Width="\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Label Grid.Row="0" Grid.Column="0" Text="ФИО"></Label>

<Label x:Name="FioLabel" Grid.Row="0" Grid.Column="1" Text="{Binding Path=UserModel.Fio}"></Label>

<Label Grid.Row="1" Grid.Column="0" Text="Номер зачетки"></Label>

<Label x:Name="ScoreBookLabel" Grid.Row="1" Grid.Column="1"></Label>

<Label Grid.Row="2" Grid.Column="0" Text="Группа"></Label>

<Label x:Name="GroupLabel" Grid.Row="2" Grid.Column="1"></Label>

<Label Grid.Row="3" Grid.Column="0" Text="Курс"></Label>

<Label x:Name="CourseLabel" Grid.Row="3" Grid.Column="1"></Label>

<Label Grid.Row="4" Grid.Column="0" Text="Квалификация"></Label>

<Label x:Name="Qualificationabel" Grid.Row="4" Grid.Column="1"></Label>

<Label Grid.Row="5" Grid.Column="0" Text="Форма обучения"></Label>

<Label x:Name="EducationFormLabel" Grid.Row="5" Grid.Column="1"></Label>

<Label Grid.Row="6" Grid.Column="0" Text="Подразделение"></Label>

<Label x:Name="DepartmentLabel" Grid.Row="6" Grid.Column="1"></Label>

<Label Grid.Row="7" Grid.Column="0" Text="Профиль"></Label>

<Label x:Name="ProfileLabel" Grid.Row="7" Grid.Column="1"></Label>

<Label Grid.Row="8" Grid.Column="0" Text="Специальность"></Label>

<Label x:Name="SpecialityLabel" Grid.Row="8" Grid.Column="1"></Label>

<Label Grid.Row="9" Grid.Column="0" Text="Период обучения"></Label>

<Label x:Name="PeriodLabel" Grid.Row="9" Grid.Column="1"></Label>

</Grid>

</StackLayout>

</ScrollView>

</ContentPage.Content>

</ContentPage>

UserPage.xaml.cs

using System.Linq;

using TryXamarin.DataHelpers;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace TryXamarin.Views.StudentViews.StudentUser

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class UserPage : ContentPage

{

public Models.Common.User UserModel;

public UserPage ()

{

Title = "Личная информация";

UserModel = GlobalMethods.GetUser();

InitializeComponent ();

FillInfo();

}

private void FillInfo()

{

FioLabel.Text = UserModel.Fio;

CourseLabel.Text = UserModel.Students.First().Course.ToString();

EducationFormLabel.Text = UserModel.Students.First().EducationForm;

DepartmentLabel.Text = UserModel.Students.First().Department;

GroupLabel.Text = UserModel.Students.First().Group;

SpecialityLabel.Text = UserModel.Students.First().Speciality;

ProfileLabel.Text = UserModel.Students.First().Profile;

ScoreBookLabel.Text = UserModel.Email.Split('@').First();

Qualificationabel.Text = UserModel.Students.First().Qualification;

}

}

}

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.CommonViews.MainMenuPage">

<ContentPage.ToolbarItems>

<ToolbarItem Clicked="MenuItem\_OnClicked" Text="LogOut">

</ToolbarItem>

</ContentPage.ToolbarItems>

<ContentPage.Content>

<StackLayout>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height ="60"/>

<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="Личная информация" Clicked="UserButton\_OnClicked"></Button>

<Button Grid.Row="1" x:Name="SessionButton" Text="Оценки" Clicked="SessionButton\_OnClicked"></Button>

<Button Grid.Row="2" x:Name="DebtsButton" Text="Оплата услуг" Clicked="DebtsButton\_OnClicked"></Button>

<Button Grid.Row="3" x:Name="TeachersButton" Text="Преподаватели" Clicked="TeachersButton\_OnClicked"></Button>

<Button Grid.Row="4" x:Name="AnnouncementButton" Text="Объявления" Clicked="AnnouncementButton\_OnClicked"></Button>

<Button Grid.Row="5" x:Name="MessagesButton" Text="Студентики" Clicked="MessagesButton\_OnClicked"></Button>

</Grid>

</StackLayout>

</ContentPage.Content>

</ContentPage>

MainMenuPage.xaml.cs

using System;

using TryXamarin.DataHelpers;

using TryXamarin.Views.StudentViews.StudentDebts;

using TryXamarin.Views.StudentViews.StudentsMarks;

using TryXamarin.Views.StudentViews.StudentUser;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace TryXamarin.Views.CommonViews

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class MainMenuPage : ContentPage

{

public MainMenuPage ()

{

Title = "Меню";

InitializeComponent ();

}

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

{

Navigation.PushAsync(new UserPage());

}

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

{

GlobalVariables.UserAuthorization = null;

App.Current.MainPage = new AuthorizationPage();

}

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

{

Navigation.PushAsync(new StudentsMarksPage());

}

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

{

Navigation.PushAsync(new StudentDebtsPage());

}

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

{

Navigation.PushAsync(new ChatsPage());

}

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

{

throw new **NotImplementedException**();

}

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

{

throw new **NotImplementedException**();

}

}

}

ChatHelper.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Text;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

using TryXamarin.Models.Common;

using TryXamarin.Models.Student;

namespace TryXamarin.DataHelpers

{

public static class ChatHelper

{

public static ChatCollection GetChatCollection()

{

var httpClient = App.HttpClient;

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

var request = new HttpRequestMessage()

{

Method = HttpMethod.Get,

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

};

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["chats"].ToObject<ChatCollection>(new JsonSerializer() { NullValueHandling = NullValueHandling.Ignore });

return debt;

}

return null;

}

public static List<ChatMessage> GetChatMessages(int chatId)

{

var httpClient = App.HttpClient;

var baseAdress = $"https://istu.ru/api/mobile/chats/{chatId.ToString()}";

var request = new HttpRequestMessage()

{

Method = HttpMethod.Get,

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

};

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 messages = jObject["messages"].ToObject<List<ChatMessage>>(new JsonSerializer(){NullValueHandling = NullValueHandling.Ignore});

return messages;

}

return null;

}

}

}

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 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<Annotation> GetAnnotations()

{

return new List<Annotation>();

}

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)

{

}

public static void AddOrUpdateConsultation(Consultation consultation)

{

}

public static void AddAnnotation(Annotation annotation)

{

}

#endregion

}

}

GlobalVariables.cs

using TryXamarin.Models.Common;

namespace TryXamarin.DataHelpers

{

public static class GlobalVariables

{

public static UserAuthorization UserAuthorization;

public static User CurrentUser;

}

}

Annotation.cs

using System;

using System.Collections.Generic;

namespace TryXamarin.Models.Common

{

public class Annotation

{

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"); }

}

}

}

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 System.Collections.Generic;

using System.Text;

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}";

}

}

}

}

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;

}

}

Student.cs

using Newtonsoft.Json;

namespace TryXamarin.Models.Student

{

public class Student

{

public string Id;

public string Group;

public int Course;

public string Qualification;

public string Profile;

public string Speciality;

[JsonProperty("Education\_form")]

public string EducationForm;

public string Department;

}

}

StudentDebt.cs

using System;

using Newtonsoft.Json;

using TryXamarin.Annotations;

namespace TryXamarin.Models.Student

{

public class StudentDebt

{

public string Id { get; set; }

[JsonProperty("user\_id")]

public string UserId { get; set; }

public double? Amount { get; set; }

public double? Penalty { get; set; }

public double? Debt { get; set; }

[JsonProperty("amount\_year")]

public double? AmountYear { get; set; }

[JsonProperty("created\_at")]

public DateTime? CreatedAt { get; set; }

[JsonProperty("qr\_amount")]

public string QRAmount { get; set; }

[JsonProperty("qr\_penalty")]

public string QRPenalty { get; set; }

[JsonProperty("updated\_at")]

public DateTime? UpdatedAt { get; set; }

[CanBeNull] public string Code { get; set; }

}

}

StudentsMark.cs

using System;

using Newtonsoft.Json;

namespace TryXamarin.Models.Student

{

public class StudentsMark

{

public string Id { get; set; }

[JsonProperty("Subject")]

public string SubjectName { get; set; }

[JsonProperty("form\_control")]

public string FormControl { get; set; }

public string Term { get; set; }

public string Year { get; set; }

public string Staff { get; set; }

public string Mark { get; set; }

public DateTime Date { get; set; }

}

}

Teacher.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace TryXamarin.Models.Student

{

public class Teacher

{

public int Id { 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}";

}

}

}

}

ПРИЛОЖЕНИЕ 2

## РЕЗУЛЬТЬТАТЫ РАБОТЫ ПРОГРАММЫ

При запуске программы открывается окно авторизации. Для дальнейшего пользования приложением требуется ввести логин и пароль.

Авторизация пользователя

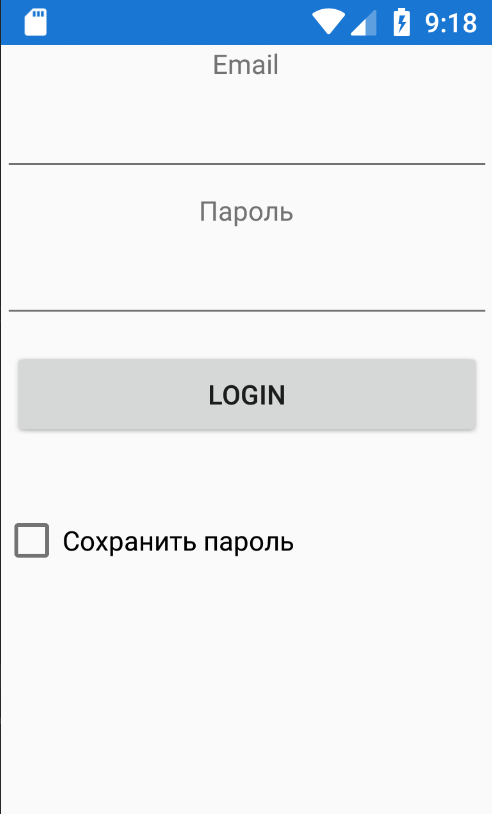


Рис. П.2.1

В случае неудачной авторизации об этом будет сообщено пользователю

Сообщение об ошибке авторизации

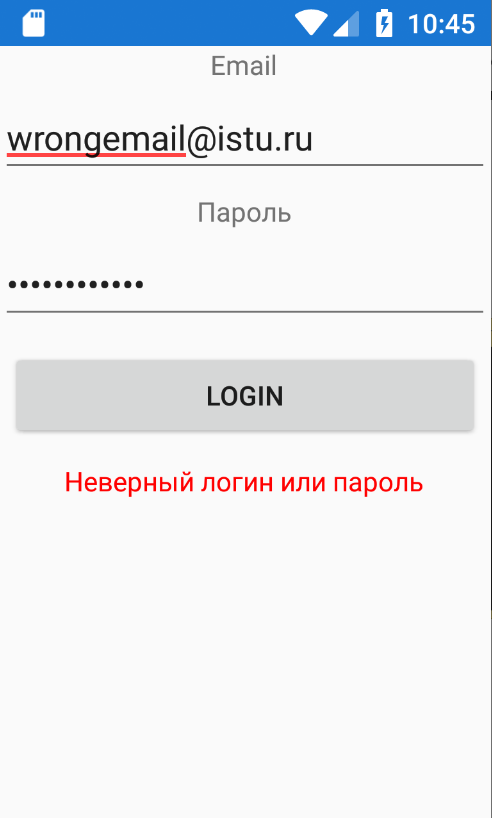


Рис. П.2.2

В случае успешной авторизации и определения роли студента, откроется главное меню студента.

Главное меню

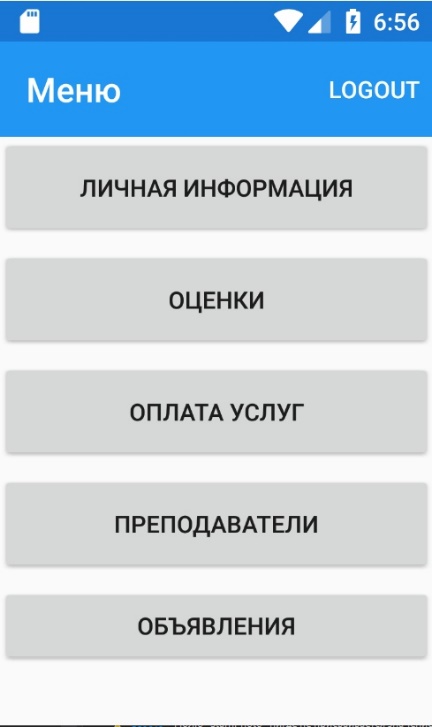


Рис. П.2.3

По нажатии на кнопку «Личная информация» откроется личная информация студента.

Личная информация пользователя

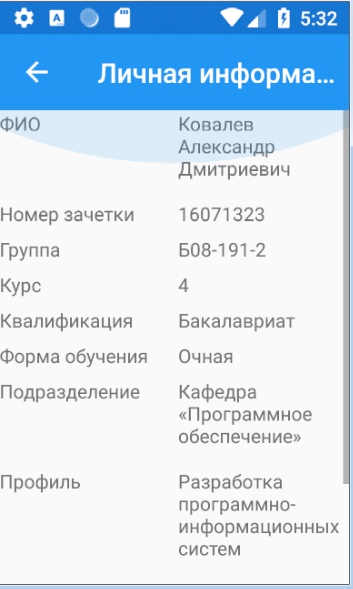


Рис. П.2.4

По нажатии на кнопку «оценки» откроется личная информация студента. Пользователь может выбрать интересующий его учебный год и семестр

Оценки

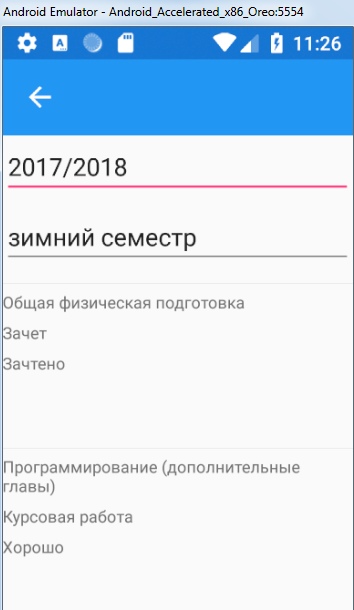


Рис. П.2.5

При нажатии на кнопку «преподаватели» откроется список чатов с преподавателями. Пользователю открывается список преподавателей, которые работали с ним за все время учебы в университете. Так же в списке показано количество непрочитанных сообщений в чате и последнее сообщение

Преподаватели

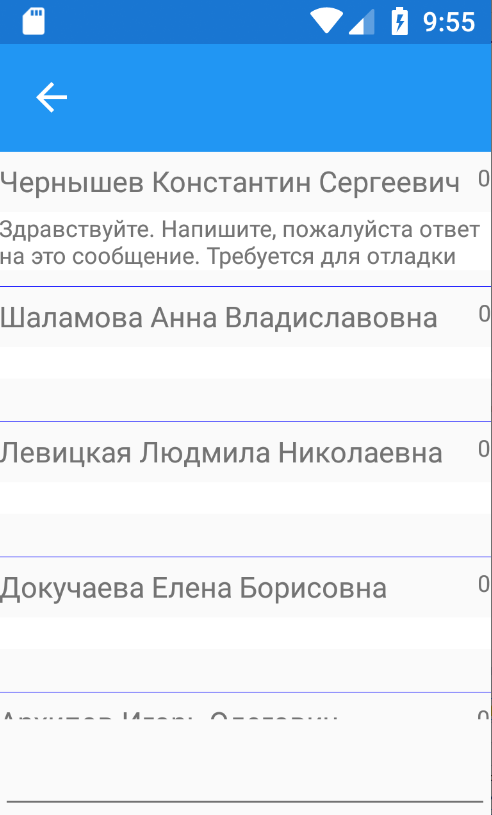


Рис. П.2.6

При нажатии на элемент списка откроется чат с выбраным преподавателем

Чат с преподавателем

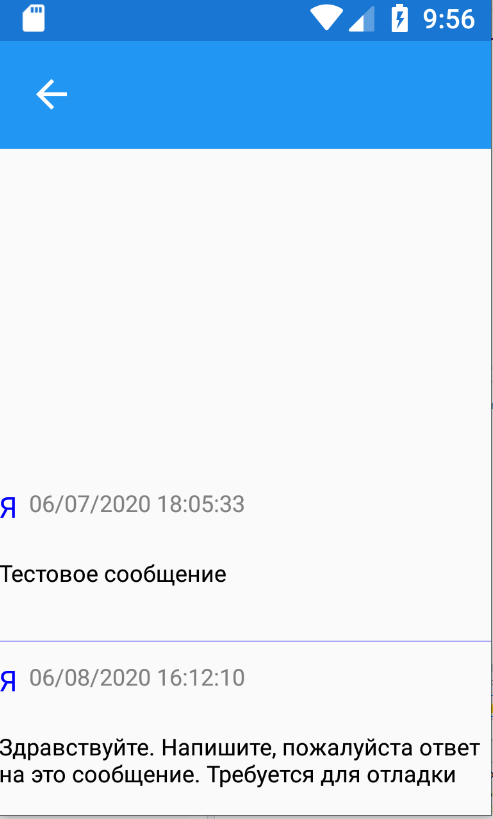


Рис. П.2.7

При длительном удержании элемента списка откроется информация о выбранном преподавателе

Информация о преподавателе

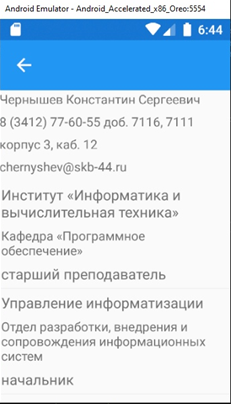


Рис. П.2.8

ПРИЛОЖЕНИЕ 3

## РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПРИЛОЖЕНИЯ

«Личный кабинет студента»

### П. 3.1 Назначение программы

Часть приложения, реализующая функционал студента, служит для отображения основной информации, нужной студенту в период обучения и хранящейся на серверах университета, а также для связи между пользователем студентом и преподавателями посредством мобильной версии стандартного чата из desktop -версии сайта университета.

Главное меню состоит из следующих пунктов:

* личная информация,
* оценки,
* оплата услуг,
* преподаватели,
* объявления.

### П.3.2 Условия выполнения программы

Требования к техническому обеспечению устройства пользователя приложения:

* Сенсорный экран
* оперативная память не менее 256 МБ;
* не менее 32 МБ свободного места на внутреннем накопителе;
* установленное приложение клавиатуры (Google или Apple);

Информационная система предъявляет следующие требования к программным средствам:

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

### П. 3.3 Пуск программы

Для запуска программы нужно выбрать приложение в проводнике смартфона путем нажатия соответствующей иконки.

### П. 3.4 Команды оператора

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

Страница авторизации

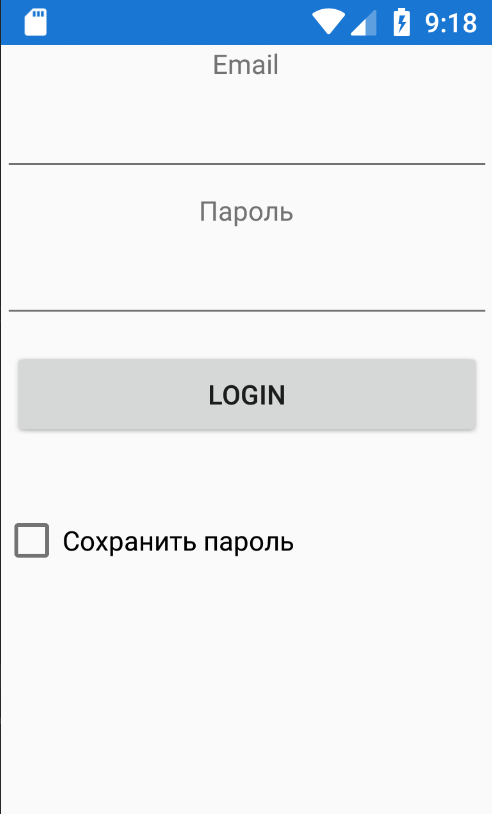


Рис. П.3.1

Также можно поставить галочку для сохранения введенных данных авторизации до следующего пуска. После авторизации пользователю станет доступно главное меню. Главное меню представлено на рис. П.3.2

Страница главного меню

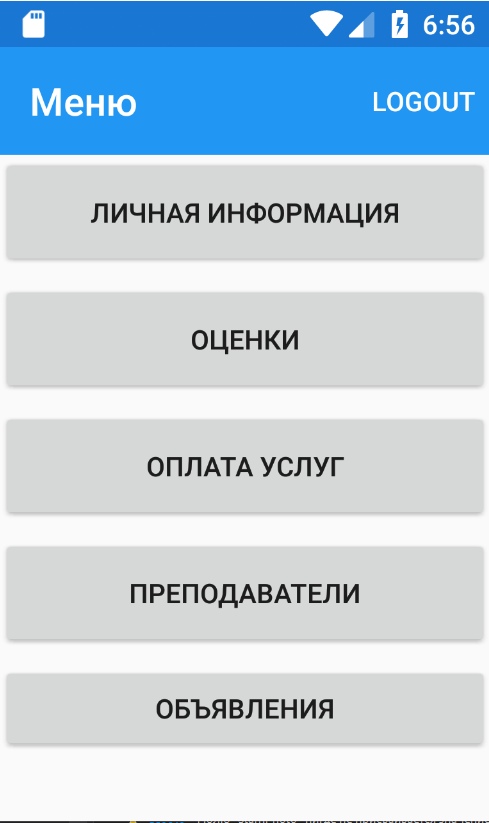


Рис. П.3.2

Для перехода к требуемому разделу, нужно нажать пальцем на соответствующую кнопку в главном меню.

Таблица П.3.1

|  |  |
| --- | --- |
| Пункт меню | описание |
| Личная информация | Просмотр личной информации пользователя |
| Оценки | Просмотр результатов сессий, сгруппированных по учебному году и семестру |
| Оплата услуг | Просмотр информации для совершения платежей для оплаты образовательных услуг |
| Преподаватели | Просмотр списка преподавателей студента, открытой информации по преподавателю, чат с преподавателем |
| Объявления | Просмотр объявлений от преподавателей |

### П. 3.5 Сообщения оператору

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

Таблица П.3.2

|  |  |
| --- | --- |
| Сообщения оператору | Действия оператора |
| Введен неверный логин или пароль | Проверить корректность ввода логина и пароля |