

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
И ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Направление подготовки: 09.03.04 – Программная инженерия
Профиль: Технологии разработки информационных систем

КУРСОВАЯ РАБОТА
РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ ДЛЯ ОТОБРАЖЕНИЯ
ПЕРСОНИФИЦИРОВАННОГО АКТУАЛЬНОГО РАСПИСАНИЯ ИТИС КФУ

Студент 3 курса

Группы 11-606

«___» _____ 2019г.

_____/ (Шакиров А.А.)

Научный руководитель

ассистент кафедры программной инженерии

«___» _____ 2019г.

_____/ (Шахова И.С.)

Казань - 2019 г.

СОДЕРЖАНИЕ

Введение.....	3
1 Анализ существующих аналогов.....	4
2 Архитектура системы.....	5
3 Описание работы серверной части приложения.....	7
3.1 Загрузка таблицы.....	7
3.2 Первичное выделение информации о расписании.....	7
3.3 Обновление информации в базе данных.....	7
3.4 Синтаксический анализ для выделения курсов по выбору и преподавателей иностранного языка.....	8
3.5 Подготовка ответов на запросы расписания.....	10
3.6 Оповещение клиентских приложений об изменениях.....	11
4 Описание работы клиентской части приложения.....	12
4.1 Используемые библиотеки.....	12
4.2 Загрузка расписания.....	12
4.3 Описание работы фильтрации курсов по выбору.....	12
4.4 Уведомления при изменении расписания.....	13
5 Пользовательский интерфейс.....	14
5.1 Экран выбора группы.....	14
5.2 Экран настройки отображения курсов по выбору.....	14
5.3 Экран с расписанием.....	15
5.4 Экран настроек.....	16
5.5 Уведомление.....	17
Заключение.....	19
Список использованных источников.....	20
Приложение А.....	21
Исходный код программы.....	21

ВВЕДЕНИЕ

На данный момент для студентов ИТИС КФУ существует несколько способов просмотра расписания через Интернет, но все они либо предоставляют неактуальные данные, либо отображают все расписание в единой таблице, без возможности персонализации на основе группы и курсов по выбору пользователя. К тому же для отслеживания изменений студенты вынуждены периодически самостоятельно проверять расписание.

Поэтому требуется мобильное приложение, которое позволяло бы просматривать актуальное персонализированное расписание и получать уведомления в случае его изменения. Разработка такого приложения для ОС Android, а также серверной части для него, и будет являться целью работы.

Итоговое клиент-серверное приложение должно включать в себя следующие функциональные решения:

1. Возможность просмотра расписания группы студента, полученного из Google таблицы [1] (наиболее актуальный источник).
2. Возможность отображать только выбранные студентом курсы по выбору.
3. Автоматические уведомления при изменении расписания, если изменение касается пользователя.

1 АНАЛИЗ СУЩЕСТВУЮЩИХ АНАЛОГОВ

На данный момент для студентов КФУ ИТИС существуют следующие способы просмотра расписания через Интернет:

1. Расписание в Google таблице [1]. Является наиболее актуальным расписанием. Содержит данные о перенесенных или отмененных парах. К сожалению, отсутствует возможность персонализации расписания, все группы представлены в общей таблице, скрыть курсы по выбору, неинтересные пользователю, невозможно.

2. Официальный сайт КФУ [2]. К сожалению, расписание на сайте КФУ, в отличие от расписания в Google таблице, не содержит информации о перенесенных или отмененных парах, и в некоторых случаях может быть устаревшим.

3. Мобильные приложения [3] [4], Telegram бот и некоторые другие сервисы, основывающиеся на информации с сайта КФУ. Основной недостаток тот же, что и у сайта КФУ — неактуальная информация.

Также у всех вышеперечисленных сервисов отсутствует возможность получать уведомления об изменении расписания группы.

2 АРХИТЕКТУРА СИСТЕМЫ

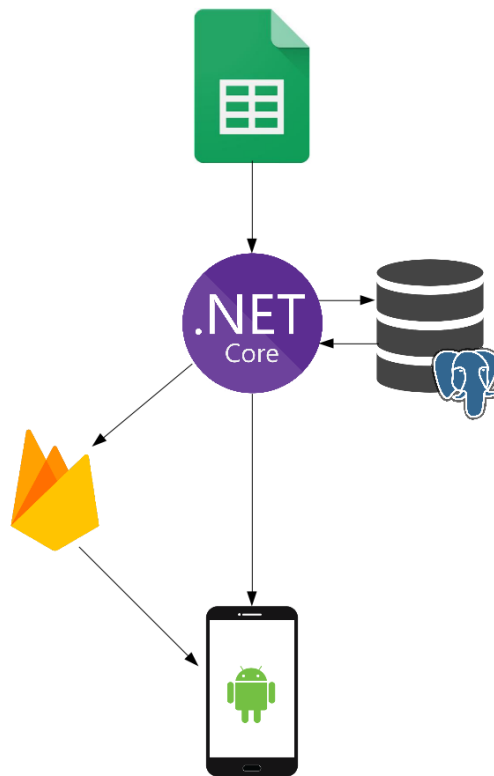


Рисунок 1 - Краткая схема работы приложения

Система состоит из следующих компонентов:

1. Google таблица, из которой происходит получение данных о расписании.

2. Серверная часть — ASP.NET Core Web API. Выполняет загрузку данных из Google таблицы, преобразование во внутренний формат данных, синтаксический анализ информации для выделения курсов по выбору, сохранение информации в базе данных, предоставление информации клиентскому приложению по протоколу http, а также оповещение клиентского приложения при изменениях расписания через Firebase Cloud Messaging.

3. PostgreSQL база данных, используемая для хранения последнего загруженного расписания и всех изменений расписания. Это требуется для избегания потери информации об изменениях расписания при перезапуске или временном отключении сервера, а также может в дальнейшем использоваться для реализации функции отображения истории изменений.

4. Мобильное приложение на ОС Android, выполняющее загрузку информации о имеющихся в таблице группах и расписание выбранной пользователем группы с сервера, отображение расписания и уведомлений об изменении расписания, с учетом выбранных пользователем курсов по выбору и других настроек.

5. Firebase Cloud Messaging, используемый для оповещения клиентского приложения об изменениях, даже если оно в данный момент не запущено. Использование FCM позволяет уменьшить расход заряда батареи, так как отсутствует необходимость постоянной работы приложения для отслеживания изменений [5]. Выбор FCM обусловлен тем, что Google Cloud Messaging объявлен устаревшим [6].

3 ОПИСАНИЕ РАБОТЫ СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ

3.1 Загрузка таблицы

Загрузка расписания происходит с использованием библиотеки Google.Apis.Sheets.v4 [7].

Загруженные данные также содержат сведения о цветах, шрифтах и прочую информацию, не используемую в дальнейшем. Поэтому полученная таблица приводится к внутреннему формату, хранящему лишь текстовое содержимое ячеек и данные об объединениях, что позволяет ускорить обработку на дальнейших шагах и использовать более лаконичный программный код.

3.2 Первичное выделение информации о расписании

На данном этапе из таблицы, представляющей собой набор ячеек с текстовой информацией извлекается информация о днях недели и времени пар, существующих группах и расписании каждой группы без дополнительного анализа курсов по выбору.

Для выделения данной информации используются хранящиеся в базе данных сведения о формате расписания. В случае изменения формата таблицы эти сведения могут быть оперативно обновлены посредством специального http запроса без перезапуска сервера.

3.3 Обновление информации в базе данных

Новое расписание сравнивается с хранимым в базе данных и, если присутствуют изменения, происходит обновление соответствующей информации в БД.

В БД хранится информация о днях недели и времени пар, а также все пары каждой группы, без дополнительной информации о курсах по выбору. В случае изменения формата расписания в Google таблице, изменять структуру БД не потребуется.

3.4 Синтаксический анализ для выделения курсов по выбору и преподавателей иностранного языка

Сначала происходит поиск во всем расписании пар, которые соответствуют курсам по выбору или иностранному языку. Это не представляет сложности, так как данные пары всегда начинаются с ключевых слов («Курс по выбору» или «Иностранный язык»).

Дальнейший анализ курсов по выбору и преподавателей иностранного языка происходит по-разному.

3.4.1 Преподаватели иностранного языка

Преподаватели иностранного языка в расписании разделены запятыми, это свойство и используется для синтаксического анализа.

Первым шагом является временное удаление всего текста, находящегося в скобках, так как там могут содержаться запятые.

Затем полученная строка разделяется по запятым. Для каждой части создается идентификатор, представляющий собой приведенные к верхнему регистру буквы, кроме находящихся в скобках. Полученные идентификаторы затем используются в клиентском приложении для идентификации преподавателей после внесения изменений в расписание.

Пример результата:

```
"rawLesson": "Иностранный язык: английский Варламова Е.В.1412  
(29.04 не будет) , Хасанова О.В. 1304, Алеева Г.Х. 1305 (с 15.04 по  
30.04 не будет перенос с 1.04 по 10.04 и 11.05) , Сиразова Л.С..1309,  
Гималетдинова Г.К.1405, Аюпова Р.А.1303, Халитова Л.К.1404",  
"lessonsGroup": {  
  "groupName": "Иностранный язык: английский",
```



```

"items": [
  {
    "raw": "Варламова Е.В.1412 (29.04 не будет)",
    "id": "ВАРЛАМОВАЕВ"
  },
  {
    "raw": "Хасанова О.В. 1304",
    "id": "ХАСАНОВАОВ"
  },
  {
    "raw": "Алеева Г.Х. 1305 (с 15.04 по 30.04 не  
будет перенос с 1.04 по 10.04 и 11.05)",
    "id": "АЛЕЕВАГХ"
  },
  {
    "raw": "Сиразова Л.С..1309",
    "id": "СИРАЗОВАЛС"
  },
  {
    "raw": "Гималетдинова Г.К.1405",
    "id": "ГИМАЛЕТДИНОВАГК"
  },
  {
    "raw": "Аюпова Р.А.1303",
    "id": "АЮПОВАРА"
  },
  {
    "raw": "Халитова Л.К.1404",
    "id": "ХАЛИТОВАЛК"
  }
]
}

```

3.4.2 Курсы по выбору

В большинстве случаев курсы по выбору в расписании разделены запятыми, но иногда — нет. При этом следует учесть, что в отличие от преподавателей иностранного языка, список курсов по выбору остается неизменным в течение всего семестра. Поэтому для анализа курсов по выбору используется заранее подготовленный список.

Для каждого курса из списка создается идентификатор, представляющий собой буквы в нижнем регистре.

Затем строка с курсами по выбору приводится к аналогичному виду — только буквы, приведенные к нижнему регистру, исключая то, что находится в скобках. При этом создается специальный массив чисел,

благодаря которому по индексу символа в в полученной строке можно найти его символ в исходной.

Следующим шагом является поиск вхождений идентификаторов из списка в полученной строке. Также можно в дальнейшем улучшить алгоритм, используя неточный поиск, что позволит анализировать текст с опечатками.

Используя информацию о вхождениях идентификаторов курсов и массив преобразования индексов, исходная строка разделяется на части, соответствующие найденным курсам по выбору. Для каждого найденного курса сохраняется его идентификатор, который затем используются в клиентском приложении для идентификации курсов по выбору после внесения изменений в расписание.

Пример:

```
"rawLesson": "Курс по выбору: Введение в робототехнику Магид  
Е.А. 1409(18.04 переносится на 26.04 пт в15.40 и 17.20 в  
ауд.1509),Методы оптимизации Фазылов В.Р.1404,Искусственный интеллект  
в компьютерных играх Кугуракова В.В.гр.№1 в 1408 (практика 9н.)",  
  "lessonsGroup": {  
    "groupName": "Курс по выбору:",  
    "items": [  
      {  
        "raw": "Введение в робототехнику Магид Е.А.  
1409(18.04 переносится на 26.04 пт в15.40 и 17.20 в ауд.1509)",  
        "id": "введениевробототехникумагид"  
      },  
      {  
        "raw": "Методы оптимизации Фазылов В.Р.1404",  
        "id": "методыоптимизациифазылов"  
      },  
      {  
        "raw": "Искусственный интеллект в компьютерных  
играх Кугуракова В.В.гр.№1 в 1408 (практика 9н.)",  
        "id":  
"искусственныйинтеллектвкомпьютерныхиграхкугуракова"  
      }  
    ]  
  }  
}
```

3.5 Подготовка ответов на запросы расписания

Полученная информация о расписании каждой группы хранится в

оперативной памяти вместе с дополнительной информацией (напр. минимальной версией клиентского приложения) в том виде, в котором должна быть отправлена клиентскому приложению при запросе расписания. Это позволяет обрабатывать большое количество запросов даже при низком быстродействии сервера, так как не требуется обращаться к базе данных или выполнять повторную обработку информации при каждом запросе.

3.6 Оповещение клиентских приложений об изменениях

Если расписание какой-либо группы было изменено (напр. изменилось время пары, была добавлена информация о переносе или отмене пары), то клиентские приложения, в которых пользователь выбрал данную группу получают об этом сообщение через Firebase Cloud Messaging. Для взаимодействия с Firebase на сервере используется библиотека FirebaseAdmin [8].

4 ОПИСАНИЕ РАБОТЫ КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ

Клиентская часть представляет собой мобильное приложение для ОС Android. Программный код написан на языке Kotlin.

4.1 Используемые библиотеки

В приложении используются RxJava [9] и RxAndroid [10] для работы с потоками. При работе Android приложения некоторые методы можно вызывать только из главного потока, поэтому реализация правильной работы с потоками без этих библиотек была бы слишком трудоемка.

Для получения информации с сервера используются библиотеки Retrofit, OkHttp, Gson.

Gson также используется для преобразования некоторой информации, которую требуется хранить на устройстве.

Приложение также использует множество других библиотек для решения различных задач.

4.2 Загрузка расписания

Расписание загружается с сервера по протоколу http. Затем оно сохраняется локально и позднее может быть отображено даже при отсутствии подключения к Интернету.

При запуске приложения сначала отображается расписание, сохраненное локально, а затем заменяется расписанием, загруженным с сервера. Это позволяет пользователю увидеть расписание сразу после запуска приложения даже при медленном подключении к Интернету.

4.3 Описание работы фильтрации курсов по выбору

После выбора пользователем отображаемых курсов по выбору в локальной базе данных сохраняются идентификаторы всех курсов, и настройки отображения каждого из них. Затем данная информация используется при отображении расписания.

Если в расписании появится курс с неизвестным ранее идентификатором, пользователю будет предложено выбрать, отображать новый курс или нет. Выбор будет сохранен в БД.

Аналогично работает фильтрация отображения преподавателей иностранного языка.

4.4 Уведомления при изменении расписания

Когда серверное приложение отправляет сообщение о том, что расписание было изменено, Firebase запускает `MyFirebaseMessagingService` (даже если приложение в данный момент закрыто). Затем происходит загрузка последней версии расписания и сравнение с расписанием, хранимым локально. При сравнении учитываются также курсы по выбору пользователя и преподаватели иностранного языка. Если изменение расписания значимо для пользователя, будет отображено соответствующее уведомление.

5 ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

5.1 Экран выбора группы

Данный экран предназначен для выбора группы и отображается при первом запуске приложения, а также может быть вызван из настроек.

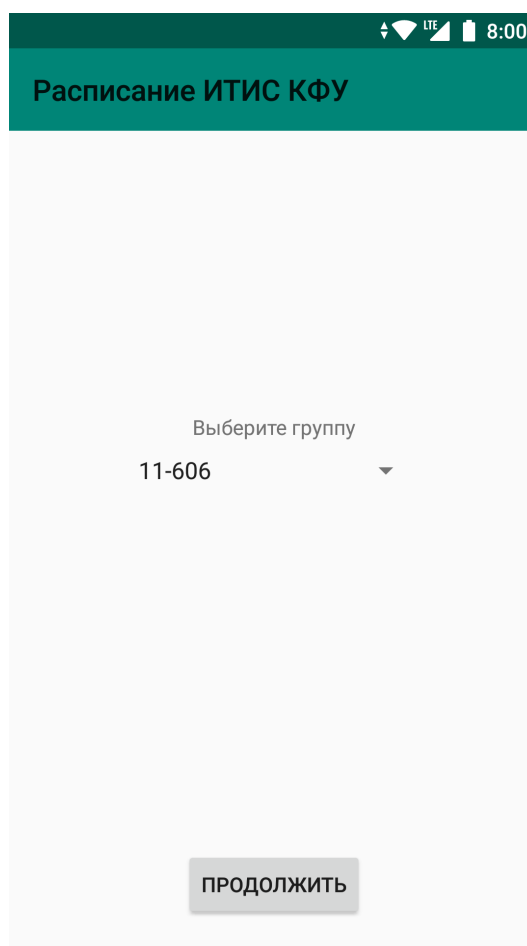


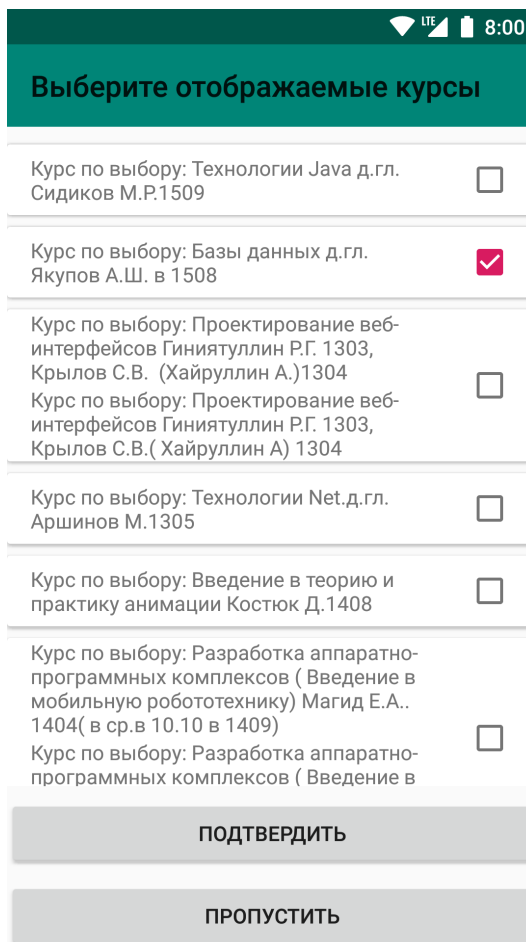
Рисунок 2 - Экран выбора группы

5.2 Экран настройки отображения курсов по выбору

Этот экран используется для выбора отображаемых курсов по выбору и преподавателей иностранного языка. Также можно пропустить выбор, тогда будут отображаться все курсы и преподавателя. Экран отображается

автоматически после выбора группы, а также может быть вызван из настроек для изменения ранее сделанного выбора.

Следует отметить, что если в расписании имеются курсы по выбору, с одинаковым идентификатором, но разным содержанием (напр. если пары проходят в разных кабинетах), то они отображаются в одной группе.



Выберите отображаемые курсы	
Курс по выбору: Технологии Java д.гл. Сидиков М.Р.1509	<input type="checkbox"/>
Курс по выбору: Базы данных д.гл. Якупов А.Ш. в 1508	<input checked="" type="checkbox"/>
Курс по выбору: Проектирование веб-интерфейсов Гиниятуллин Р.Г. 1303, Крылов С.В. (Хайруллин А.)1304	<input type="checkbox"/>
Курс по выбору: Проектирование веб-интерфейсов Гиниятуллин Р.Г. 1303, Крылов С.В.(Хайруллин А) 1304	<input type="checkbox"/>
Курс по выбору: Технологии Net.д.гл. Аршинов М.1305	<input type="checkbox"/>
Курс по выбору: Введение в теорию и практику анимации Костюк Д.1408	<input type="checkbox"/>
Курс по выбору: Разработка аппаратно-программных комплексов (Введение в мобильную робототехнику) Магид Е.А.. 1404(в ср.в 10.10 в 1409)	<input type="checkbox"/>
Курс по выбору: Разработка аппаратно-программных комплексов (Введение в	<input type="checkbox"/>
ПОДТВЕРДИТЬ	
ПРОПУСТИТЬ	

Рисунок 3 - Экран выбора отображаемых курсов

5.3 Экран с расписанием

На данном экране непосредственно отображается расписание. Он отображается при запуске приложения, если ранее была выбрана группа.

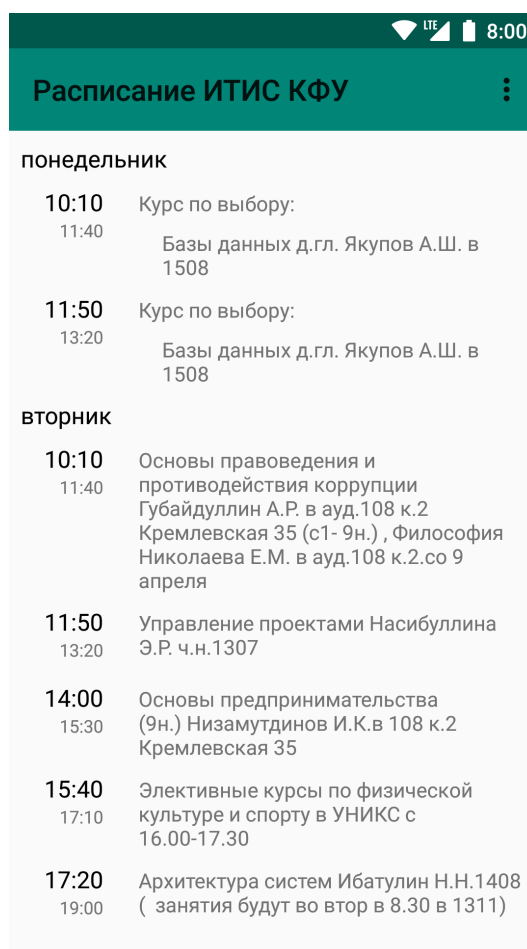


Рисунок 4 - Экран с расписанием

5.4 Экран настроек

На данном экране можно изменить настройки отображения расписания (напр. скрытие пустых пар), или перейти к изменению выбора группы или курсов по выбору.

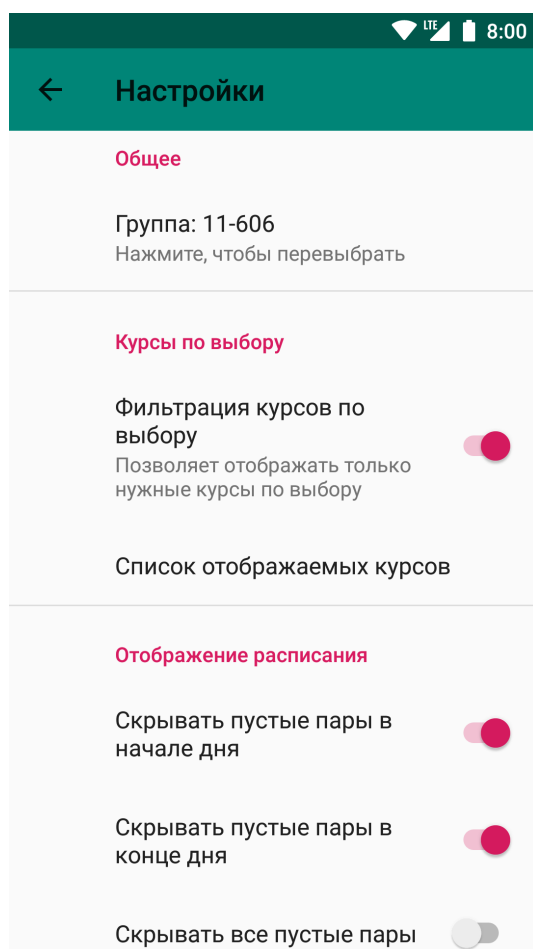
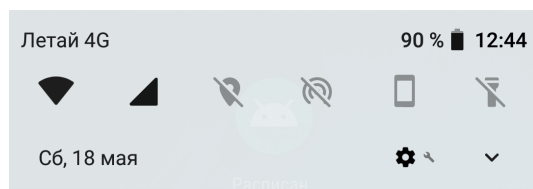


Рисунок 5 - Экран настроек

5.5 Уведомление

Уведомление, отображаемое при изменении расписания. Содержит день недели и время пары, которая была изменена.



Расписание ИТИС КФУ • сейчас

Расписание изменено
понедельник 10:10

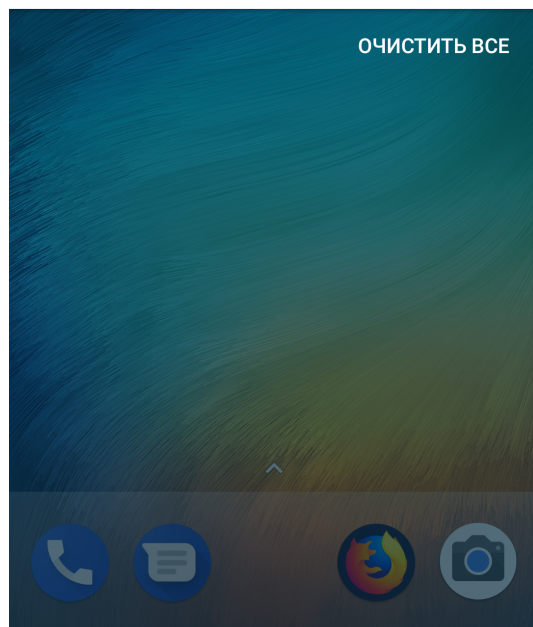


Рисунок 6 - Уведомление

ЗАКЛЮЧЕНИЕ

Результатом работы является клиент-серверное приложение, позволяющее пользователю просматривать актуальное расписание ИТИС КФУ, а также получать уведомления при его изменении.

Потенциальными пользователями приложения являются студенты ИТИС КФУ.

Приложение имеет возможность настроить отображение расписания в соответствии с персональными пожеланиями.

Был проведен опрос среди 10 студентов ИТИС. 7 опрошиваемых подтвердили, что использование приложения удобнее просмотра расписания в Google таблице с мобильного устройства.

В дальнейшем планируется расширить функционал приложения возможностью просмотра истории изменений расписания.

Также планируется улучшить синтаксический анализ курсов по выбору для уменьшения вероятности ошибки.

Исходный код приложения доступен в сети Интернет [11].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Google таблица с расписанием ИТИС КФУ [Электронный ресурс] - <https://docs.google.com/spreadsheets/d/16U6dbrd4zGhiQW-c3SR4OE2ueUNX4ts8MYjWiT7YqLw>
2. Расписание на сайте КФУ [Электронный ресурс] - <https://kpfu.ru/studentu/ucheba/raspisanie>
3. Android приложение «Расписание КФУ без авторизации» [Электронный ресурс] - <https://play.google.com/store/apps/details?id=burov.schedule.kpfu>
4. Android приложение «КФУ» [Электронный ресурс] - <https://play.google.com/store/apps/details?id=com.kfu.lantimat.kfustudent>
5. Официальная документация Firebase Cloud Messaging [Электронный ресурс] - <https://firebase.google.com/docs/cloud-messaging/>
6. Официальная документация Google Cloud Messaging [Электронный ресурс] - <https://developers.google.com/cloud-messaging/>
7. Документация по Google.Apis.Sheets.v4 [Электронный ресурс] - <https://developers.google.com/sheets/api/quickstart/dotnet>
8. Документация по FirebaseAdmin [Электронный ресурс] - <https://firebase.google.com/docs/reference/admin/dotnet>
9. Документация по RxJava [Электронный ресурс] - <https://github.com/ReactiveX/RxJava/wiki>
10. Документация по RxAndroid [Электронный ресурс] - <https://github.com/ReactiveX/RxAndroid>
11. Исходный код приложения [Электронный ресурс] - <https://github.com/Alm7z/timetable>

ПРИЛОЖЕНИЕ А

Исходный код программы

Синтаксический анализ для выделения курсов по выбору:

```
using System.Collections.Generic;
using System.IO;
using core.ClientModel;
using System;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;

namespace core.Parsing.LGParsing
{
    public class LGBByListParser
    {
        public static string[] startPatterns = readStartsFromFile();

        public static string[] readStartsFromFile()
        {
            string[] lines =
File.ReadAllLines("electives_starts.txt");
            lines = lines.Select(s => s.OnlyLetters().ToLower())
                .Where(s => s.Length > 0)
                .ToArray();

            if (lines.Distinct().Count() != lines.Length)
                Console.WriteLine("WARNING!!! lines.Distinct !=
lines");

            return lines;
        }

        public static List<ClientLessonsGroupItem> Split(string
lessonsPart)
        {
            List<(string key, int indexInString)> found = new
List<(string key, int index)>();

            var letters =
lessonsPart.ToLower().OnlyLettersSavingIndexes();
            foreach (string startPattern in startPatterns)
            {
                int index = letters.str.IndexOf(startPattern);
                if (index != -1)
                {
                    found.Add((startPattern, letters.map[index]));
                }
            }

            found = found.OrderBy(f => f.indexInString).ToList();
        }
    }
}
```

```

        var distinctFound = found.GroupBy(f => f.indexInString)
            .Select(g => g.First())
            .ToList();

        if (distinctFound.Count != found.Count)
            Console.WriteLine("WARNING! distinctFound != found
(using distinctFound)");

        found = distinctFound;

        List<string> parts =
lessonsPart.SplitByIndexes(found.Select(f => f.indexInString));

        parts = parts
            .Select(s => Regex.Replace(s, @"(^[ ,;]+)|([ ,;]+$)",
""))
            .ToList();

        List<ClientLessonsGroupItem> res = new
List<ClientLessonsGroupItem>();

        for (int i = 0; i < parts.Count; i++)
        {
            res.Add(new ClientLessonsGroupItem(parts[i],
found[i].key));
        }

        return res;
    }
}

```

Синтаксический анализ для выделения преподавателей иностранного

ЯЗЫКА:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using core.ClientModel;
using core.ServerModel;

namespace core.Parsing.LGParsing
{
    public static class LGByDividersParser
    {
        public static List<ClientLessonsGroupItem> Split(string
lessonsPart)
        {
            var lessons = SplitStrByDividers(lessonsPart)
                .Select(s => s.Trim())
                .Where(s => s.Length > 0);

            return lessons.Select(l => new ClientLessonsGroupItem(

```

```

        1,
        1.OutsideBrackets()
        .OnlyLetters()
        .ToUpper()
    ))
    .ToList();
}

private static List<char> dividers = new List<char>()
{
    ',',
    ';',
};

private static List<string> SplitStrByDividers(string source)
{
    Dictionary<(char start, char end), int> bracesCounts = new
Dictionary<(char start, char end), int>
    {
        { ('(', ')'), 0 },
        { ('[', ']'), 0 },
        { ('{', '}'), 0 }
    };

    List<int> divPositions = new List<int>();

    divPositions.Add(0);

    for (int i = 0; i < source.Length; i++)
    {
        foreach (var keyValuePair in bracesCounts)
        {
            if (source[i] == keyValuePair.Key.start)
            {
                bracesCounts[keyValuePair.Key]++;
                break;
            }
            if (source[i] == keyValuePair.Key.end)
            {
                bracesCounts[keyValuePair.Key]--;
                break;
            }
        }

        if (dividers.Contains(source[i]) && bracesCounts.All(p
=> p.Value == 0))
        {
            divPositions.Add(i);
        }
    }

    divPositions.Add(source.Length);

    List<string> res = new List<string>();
    for (int i = 0; i < divPositions.Count - 1; i++)
    {

```

```

        res.Add(source.Substring(divPositions[i],
divPositions[i + 1] - divPositions[i]));
    }

    res = res
        .Select((s, i) => i > 0 ? s.Substring(1) : s)
        .Select(s => s.Trim())
        .ToList();

    return res;
}
}
}

```

Сравнение старого и нового расписания перед отображением уведомления:

```

private fun compareTimetables(
    old: List<LessonInfo>,
    new: List<LessonInfo>,
    selected: List<CourseSelection>?
) {
    if (old.size != new.size) {
        showDefNotif()
        return
    }

    for (i in 0 until old.size) {
        if (old[i] != new[i]) {
            if (old[i].lessonsGroup == null || new[i].lessonsGroup
== null || selected == null) {
                // raw || courses filter disabled
                showTNotif(old[i], new[i])
            } else {
                // added new course id
                if (!new[i].lessonsGroup!!.items.all { lgi ->
                    selected.any { it.id == lgi.id }
                }) {
                    showTNotif(old[i], new[i])
                }

                if (new[i].lessonsGroup!!.items.any { lgi ->
                    val sel = selected.find { it.id ==
lgi.id } ?: return@any true
                    if (sel.selected) {
                        val oldLgi =
old[i].lessonsGroup!!.items.find { it.id == lgi.id } ?: return@any
true
                        if (oldLgi.raw != lgi.raw)
                            return@any true
                    }
                }
                false
            }
        }
    }
}

```



```
        )) {
            showTNotif(old[i], new[i])
        }
    }
}
}
```