МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Вычислительной Техники\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(полное название кафедры)

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Торопова Никиты Александровича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество студента – автора работы)

\_\_\_\_\_\_\_\_\_Разработка Android-клиента кабинетов студента и преподавателя НГТУ\_\_\_\_\_\_\_\_\_

(тема работы)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Факультет Автоматики и Вычислительной Техники\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(полное название факультета)

Направление подготовки \_\_\_\_\_\_\_09.03.04 «Программная инженерия»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(код и наименование направления подготовки бакалавра)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Автор выпускной квалификационной работы**

\_\_\_\_\_\_\_\_Торопов Н.А.\_\_\_\_\_\_

(фамилия, имя, отчество)

\_\_\_\_\_\_\_АВТФ АВТ-215\_\_\_\_\_\_

(факультет, группа)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

**Руководитель от НГТУ**

\_\_\_\_\_\_\_\_Романов Е.Л.\_\_\_\_\_\_\_

(фамилия, имя, отчество)

\_\_\_\_\_\_\_\_\_к.т.н. доцент\_\_\_\_\_\_\_\_

(ученая степень, ученое звание)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

Новосибирск 2016

Реферат

Оглавление

[Введение. 4](#_Toc452162107)

[Список терминов и сокращений. 5](#_Toc452162108)

[1. Постановка задачи и анализ предметной области. 7](#_Toc452162109)

[1.1 Постановка задачи. 7](#_Toc452162110)

[1.2 Обзор аналогов. 7](#_Toc452162111)

[1.2.1 Расписашка 7](#_Toc452162112)

[1.3 Архитектура Android. 8](#_Toc452162113)

[1.4 Методы авторизации в личном кабинете. 11](#_Toc452162114)

[1.5 Выбор средств разработки. 12](#_Toc452162115)

[1.5.1 Среда разработки. 12](#_Toc452162116)

[1.5.2 Библиотеки парсинга. 13](#_Toc452162117)

[1.5.3 Программа для записи запросов на сервер. 13](#_Toc452162118)

[1.6 Требования к программному продукту 13](#_Toc452162119)

[2. Структурная разработка 14](#_Toc452162120)

[2.1 Алгоритм получения доступа в личный кабинет. 14](#_Toc452162121)

[2.1.1 Авторизации при помощи *Android WebView* 14](#_Toc452162122)

[2.1.2 Авторизация при помощи запроса на сервер 16](#_Toc452162123)

[2.2 Алгоритм извлечения информации с *web*-сайта 18](#_Toc452162124)

[3. Разработка графического интерфейса 20](#_Toc452162125)

[3.1 Описание возможностей приложения 20](#_Toc452162126)

[3.1.1 Просмотр расписания занятий. 20](#_Toc452162127)

[3.1.2 Просмотр и редактирование учебного плана. 20](#_Toc452162128)

[3.1.3 Просмотр успеваемости (Студент). 21](#_Toc452162129)

[3.1.4 Просмотр списка групп. 21](#_Toc452162130)

[3.1.5 Отправка информационных сообщений студентам. 21](#_Toc452162131)

[3.1.6 Просмотр показателей деятельности 21](#_Toc452162132)

[3.2 Варианты использования 21](#_Toc452162133)

[3.2.1 Авторизация пользователя 21](#_Toc452162134)

[3.2.2 Просмотр расписания занятий 21](#_Toc452162135)

[3.2.3 Просмотр успеваемости (студент). 22](#_Toc452162136)

[3.2.4 Просмотр списка групп. 22](#_Toc452162137)

[3.2.5 Отправка информационных сообщений студентам. 23](#_Toc452162138)

[3.2.6 Просмотр учебного плана. 23](#_Toc452162139)

[3.2.7 Просмотр показателей деятельности 23](#_Toc452162140)

# Введение.

Пока не представляю, что здесь писать. Возможно, будет текст о мобильных платформах и сайте НГТУ.

# Список терминов и сокращений.

***API*** (интерфейс программирования приложений, интерфейс прикладного программирования) (англ. *application programming interface*) — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах. Используется программистами при написании всевозможных приложений.

**Личный кабинет** – *Web*-Сайт для преподавателей и студентов хранящий в себе необходимую для работы и учёбы информацию, доступ к которому, имеет индивидуально каждый студент и преподаватель.

Пример личного кабинета:

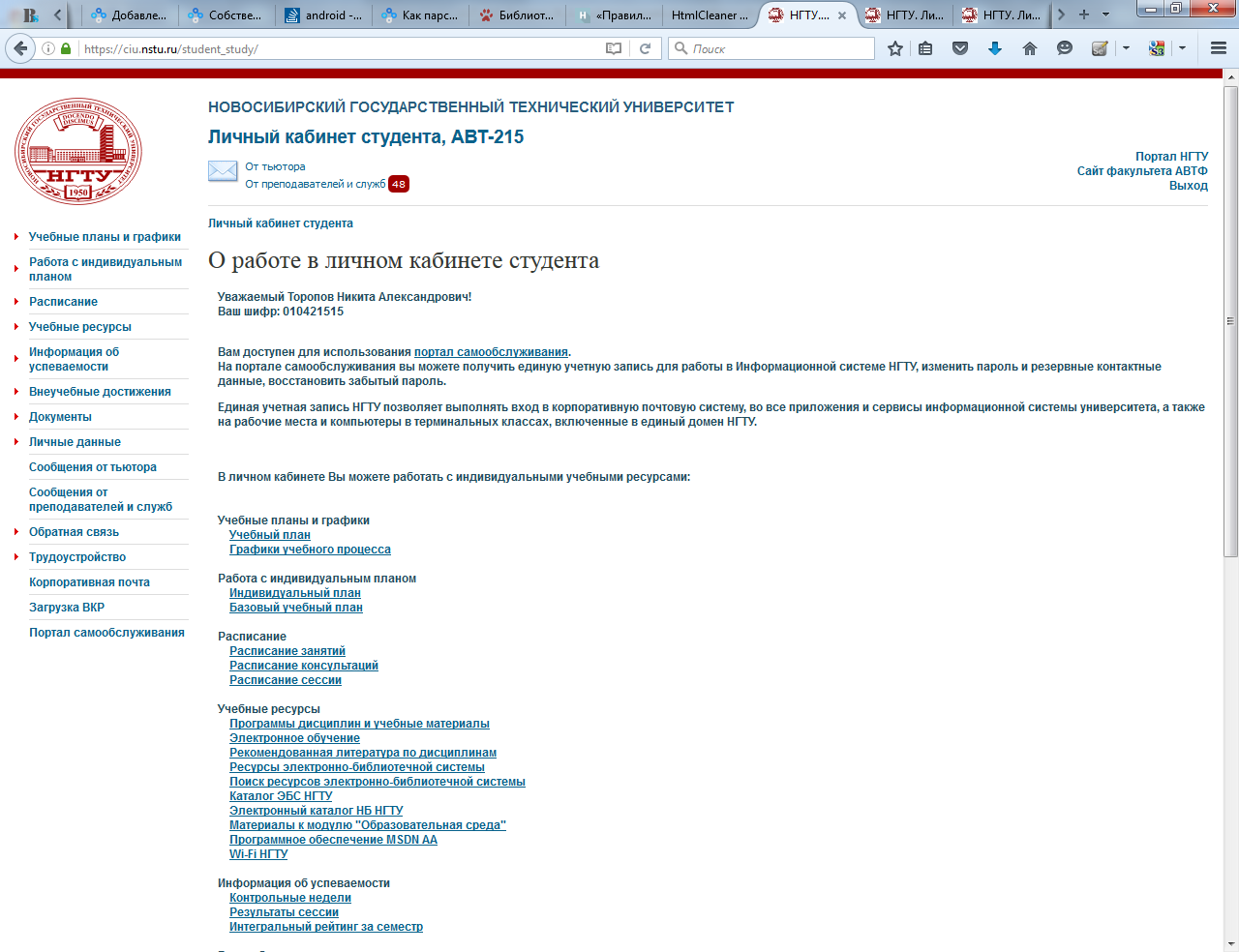


Рисунок 1. Личный кабинет студента.

**Синтаксический анализ «Парсинг»** - процесс сопоставления линейной последовательности лексем (слов, токенов) естественного или формального языка с его формальной грамматикой. Результатом обычно является дерево разбора.

**Токен -** идентификатор доступа выдаваемы после успешной авторизации. Хранится в *Cookie* браузера и передается в заголовке запроса к серверу.

***AJAX -*** *Asynchronous Javascript And Xml* – технология обращения к серверу без перезагрузки страницы;

***IDE***- Интегрированная среда разработки. Система программных средств, используемая программистами для разработки программного обеспечения.

# Постановка задачи и анализ предметной области.

## Постановка задачи.

Основную задачу можно сформулировать следующим образом: необходимо разработать программу обеспечивающую доступ в личный кабинет преподавателя или студента НГТУ с ограниченным функционалом. Такая формулировка включает в себя ряд подзадач:

* Разработка модуля авторизации
* Анализ функциональных возможностей личных кабинетов и выбор основных функций.
* Разработка модуля «личный кабинет» студента
* Разработка модуля «личный кабинет» преподавателя.

Для решения поставленной задачи необходимо провести анализ существующих решений представленных подзадач, выявить их достоинства и недостатки, и на основе результатов сравнения функциональных возможностей выбрать наиболее приемлемый вариант для собственной разработки.

## Обзор аналогов.

На сегодняшний день рынок мобильных приложений «пестрит» всевозможными приложениями любой функциональности. Следовательно, почти любая идея, выбранная для реализации, уже имеет свои аналоги и решения. Так как личный кабинет является частью *Web*-сайта НГТУ, мое приложение является неким клиентом для работы с данным сайтом. Аналогом можно представить приложение «Расписашка».

### Расписашка

Данное приложение является клиентом для работы с Web-сайтом НГТУ, а именно с расписанием занятий студентов. Приложение предоставляет возможность выбора факультета и группы, для которой необходимо вывести расписание.

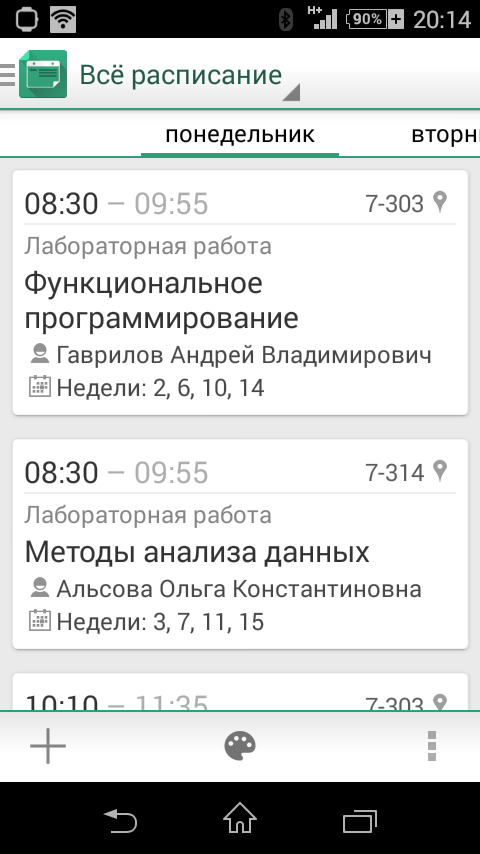


Рисунок 1.1 Внешний вид приложения «Расписашка».

Данное приложение предоставляет доступ к расписанию с Web-сайта НГТУ и не предоставляет доступ в «личный кабинет». Таким образом, можно предположить, что аналогов разрабатываемому приложению нет.

## Архитектура Android.

В настоящее время среди всех мобильных платформ наибольшей популярностью пользуются платформы *iOS* и *Android*. Однако, несмотря на то, что *iOS* более дружелюбна к пользователям, практична в вопросах функциональности и стабильна, нежели *Android*, в плане разработки она имеет больше минусов, чем плюсов. Средства разработки приложений под *iOS*, тестирование приложения, доступ к форумам разработчиков являются платными, разрабатывать приложение можно исключительно на компьютере с операционной системой *Mac* *OS*. Средства разработки *Android*, в свою очередь, находятся в свободном доступе и могут быть установлены на любой платформе, а при возникновении тупиковой ситуации всегда можно найти решение проблемы в сети или спросить у более опытных разработчиков. Язык разработки *iOS* — *Objective-C*, что тоже является минусом, поскольку я не имею достаточного для реализации задач опыта программирования на данном языке.



Рисунок 1.2 Сравнение *iOS* и *Android.*

Архитектура *Android* представляет собой программный стек для мобильных устройств, включающий в себя операционную систему, связующее программное обеспечение и ключевые приложения.



Рисунок 1.3 Основные компоненты архитектуры *Android*[3].

Как видно из рисунка 1.3, архитектуру операционной системы *Android* можно разделить на пять связанных уровней:

* Уровень приложений. *Android* поставляется с набором стандартных приложений.
* Уровень каркаса приложений (фреймворк). *Android* предоставляет разработчикам возможность разработки мощных функциональных приложений: они имеют доступ к тем же функциям, структурам и интерфейсам, которые используются при создании встроенных приложений.
* Уровень библиотек. *Android* включает в себя набор библиотек, реализованный на языке *С/С*++. Фреймворк позволяет разработчикам использовать эти библиотеки.
* Уровень среды исполнения. Платформа использует регистр-ориентированную виртуальную машину *Dalvik*. Каждое приложение запускается в собственном процессе и работает с уникальным экземпляром виртуальной машины.
* Уровень ядра *Linux*. Для работы с основными системными сервисами (управление процессами, памятью, обеспечение безопасности, работа с драйверами) используется *Linux* версии 2.6.

Самым большим плюсом ОС *Android* с точки зрения реализации поставленных задач является возможность доступа к методам, функциям и структурам, используемым при разработке стандартных приложений.

Таким образом, по таким критериям как возможности при разработке, доступность телефона с выбранной платформой и доступность средств разработки мною была выбрана платформа *Android*.

## Методы авторизации в личном кабинете.

Для получения доступа к личному кабинету необходимо иметь токен передаваемый в *Cookie*.. Для его получения можно использовать следующие варианты:

1. Загрузка страницы авторизации при помощи *Android WebView* прохождение авторизации, и извлечение *Cookie*.
2. Отправка запроса на сервер с логином и паролем пользователя и получение ответа содержащего токен.

В первом случае происходит загрузка всего контента страницы авторизации и это расходует трафик мобильного интернета, а так же загрузка происходит довольно долго, так как существует большое количество перенаправлений и «тяжелый контент». Так же страница авторизации не является адаптивной и не удобна в использовании на мобильных телефонах.

Во втором случае трафик мобильного телефона расходуется гораздо меньше, так как отсутствует какой-либо загружаемый контент. Вдобавок ко всему это ускоряет процесс авторизации, так как нет необходимости ждать загрузки страницы авторизации. Запрос отправляется напрямую на сервер и с него же получает ответ. Так же это позволяет создать более удобную форму для авторизации пользователя.

## Выбор средств разработки.

### Среда разработки.

Разработчик приложения под ОС *Android* имеет довольно большой выбор средств разработки: *IntelliJ IDEA, NetBeans, Eclipse*, а так же комплект *Android SDK* на основе *Eclipse IDE* и комплект *Android Studio* на основе *IntelliJ IDEA IDE*. Выбор среды разработки основывался наследующих требованиях:

* Наличие удобного редактора кода;
* Наличие эффективных средств отладки, позволяющих работать с кодом в пошаговом режиме;
* Наличие графического редактора *XML*-файлов;
* Наличие навыков и опыта использования выбранной среды разработки.

Результаты сравнения интегрированных сред представлены в таблице 1.

Таблица 1 — Сравнение интегрированных средств разработки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Редактор кода | Средства отладки | *XML*-редактор | Опыт использования |
| *IntelliJ IDEA* | + | + | + | + |
| *NetBeans* | + | + | - | + |
| *Eclipse* | + | + | + | - |

Мною был выбран комплект *Android Studio*, поскольку он обладает всеми перечисленными параметрами, а так же легок и удобен в настройке.

### Библиотеки парсинга.

К сожалению «личный кабинет» не имеет *API* для доступа к нему. В связи с этим для реализации приложения мне придется использовать метод «парсинга» *HTML* страниц для получения необходимой информации.

Вот небольшой перечень библиотек для парсинга *HTML:*

1. *HTML Parser*
2. *Java HTML Parser*
3. *HotSax*
4. *HtmlCleaner*
5. *Jsoup*

Я остановил свой выбор на библиотеке *HtmlCleaner* так как у неё присутствует довольно простое и понятное описание и она включает все необходимые для меня функции. Так же она занимает меньший объем памяти по сравнению с *HTML Parser* и *Java HTML Parser.*

### Программа для записи запросов на сервер.

Для отслеживания отправляемых запросов на сервер пришлось прибегнуть к программе для нагрузочного тестирования *Apach JMeter*, так как отследить запросы стандартными средствами браузера не оказалось возможным. Данная программа позволяет перехватывать отправляемые запросы на серверную сторону. Так же она позволяет просмотреть заголовки и тела запросов.

## Требования к программному продукту

Исходя из поставленной задачи, можно вывести некоторые требования к программному продукту, а именно:

* Корректность извлекаемой информации из личного кабинета.
* Сохранность личных данных пользователя
* Сокращение расхода мобильного трафика
* Упрощение работы с личным кабинетом

# Структурная разработка

## Алгоритм получения доступа в личный кабинет.

Для доступа к информации, хранящейся в личном кабинете, необходимо получить токен выдаваемый сервером при авторизации и хранящийся в заголовке запроса в графе *Cookie.* Для получения данного токена можно воспользоваться следующими алгоритмами:

* Авторизация при помощи *Android WebView*
* Авторизации при помощи запроса на сервер.

### Авторизации при помощи *Android WebView*

Для реализации данного алгоритма пришлось прибегнуть к сторонней библиотеке *Crosswalk Project.* Это свободно распространяемая библиотека, представляющая из себя улучшенный вариант *Android WebView*. Вот перечень предоставляемых улучшений:

* Использование всех возможностей доступных в современных *web* браузерах: HTML5, CSS3, JavaScript.
* Использование экспериментальных API, недоступных в большинстве основных web браузеров.
* Контролирование цикла обновления приложения путем распространения с собственной средой выполнения.

Для открытия *web* страницы на форму устанавливается элемент *XWalkView.* Пример *loginactivity.xml:*

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical" android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<org.xwalk.core.XWalkView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:id="@+id/web"/>

</LinearLayout>

Так же создается класс *Login\_By\_WebView* наследующий интерфейс *Activity*. Переопределяется метод onCreate() в котором данному классу в качестве запускаемой активности дается *loginactivity.xml.* Так же в данном методе создаются и определяются объекты для работы с *Cookie* и *XWalkView,* а именно xWalkCookieManager и xWalkView. Загрузка страницы выполняется вызовом у объекта xWalkView метода load с параметрами.

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.loginactivity);

XWalkView xWalkView= (XWalkView) findViewById(R.id.web);

XWalkCookieManager xWalkCookieManager=new XWalkCookieManager();

xWalkView.load(url, null);

}

Так же в данном классе создается объект intent для передачи параметров и запуска следующей активности.

Intent intent=new Intent(this,Programm.class);

Так же в методе onCreate() объекту xWalkView устанавливается ресурсный клиент для перехвата *Cookie* и запуска новой активности.

xWalkView.setResourceClient(new MyResourceClient(xWalkView));

В итоге при открытии данной активности открывается страница авторизации на сайте НГТУ. Пройдя авторизацию на ней можно извлечь необходимый токен и передать его далее в программу. Данный метод не является удобным, так как необходимо ожидать загрузки всех компонент находящихся на странице авторизации, что ведёт к расходу мобильного трафика. Так же данная страница не является адаптивной для мобильных устройств, что усложняет её использование. Данный метод был использован для проверки гипотезы о возможности извлечения и передачи токена.

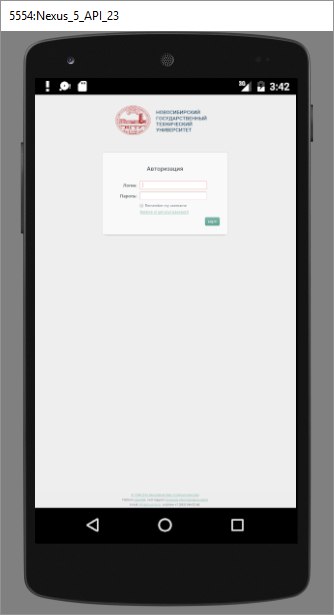


Рисунок 2.1.1.1 Открытие страницы авторизации через *WebView.*

### Авторизация при помощи запроса на сервер

При использовании программы Apach JMeter был обнаружен *POST* запрос отправляемый для авторизации. Самым важным в данном запросе является его тело. Это *JSON* объект, хранящий в себе логин и пароль пользователя, а так же хранящий авторизационный идентификатор необходимый для выдачи авторизационного токена.

Для отправки запроса необходимо запросить у пользователя логин и пароль, а у сервера авторизационный идентификатор, так как он имеет ограниченный срок жизни. Для реализации данного алгоритма был создан класс Login наследующий интерфейс *Activity*. Переопределяется метод onCreate() в котором данному классу в качестве запускаемой активности дается *login\_activity.xml.* В зависимости от выбранной роли на предыдущей активности устанавливается адрес отправки запроса. Так же в данном классе создается объект intent для передачи параметров и запуска следующей активности.

Intent intent=new Intent(this,Programm.class);

В версии Android 4.0 соединение с веб-серверами в основном потоке было запрещено. По этому, после ввода логина и пароля на форме и нажатии кнопки войти запускается асинхронный поток, в который передаются адрес отправки запросов, логин и пароль пользователя, для соединения с веб-сервером.

new MyAsyncTask(context)

.execute(url,login.getText().toString(),pass.getText().toString());

В данном потоке выполняется переопределенный метод doInBackground(). В нем создаются объекты класса HttpURLConnection для открытия соединения с сервером и отправки запросов. Первый объект класса HttpURLConnection отправляет на сервер пустой запрос методом *POST* и получает ответ, в котором хранится *JSON* объект. Из полученного объекта извлекается авторизационный идентификатор и встраивается в заранее сформированный запрос с логином и паролем пользователя.

Создание объекта класса HttpURLConnection и отправка пустого запроса:

HttpURLConnection connection = HttpURLConnection)url.openConnection();

connection.setRequestMethod("POST");

Получение ответа с сервера:

InputStream is = connection.getInputStream();

BufferedReader rd = new BufferedReader(new InputStreamReader(is));

String line;

StringBuffer response = new StringBuffer();

while((line = rd.readLine()) != null) {

response.append(line);

response.append('\r');

}

rd.close();

Извлечение *JSON* объекта из ответа сервера:

String jsonstring= response.toString();

JSONObject jsonObject = new JSONObject(jsonstring);

String authId=jsonObject.getString("authId");

После чего создается ещё один объект класса HttpURLConnection. Он отправляет запрос, в теле которого, хранится сформированный *JSON* объект и в ответе получает авторизационны токен, который возвращает после выполнения.

Отправка запроса:

DataOutputStream wr = new DataOutputStream(connection1.getOutputStream ());

wr.writeBytes (urlParameters);

wr.flush ();

wr.close ();

Далее вызывается переопределенный метод onPostExecute(String Myresult). На вход метод получает значение возвращаемое методом doInBackground(). Если значение не пустое значит, запускается следующая активность и в не передается данный результат. В противном случае метод сигнализирует об ошибке и оставляет пользователя на данной активности.

Открытие активности и передача параметра:

Intent start = new Intent(context,Programm.class);

start.putExtra("Role",Login.Role);

start.putExtra("cookie",Myresult);

startActivity(start);

Пример формы авторизации представлен в приложении А рисунок А2.

## Алгоритм извлечения информации с *web*-сайта

Для извлечения информации используется алгоритм синтаксического анализа «парсинга». *Web*-страница разбивается на лексемы (тэги). После чего, из полученных тэгов выбираются необходимые для дальнейшей работы, например таблицы или списки. Для каждой из таблиц создается объект класса Tabel, для списков объект класса Spisok.

Для начала «парсинга» необходимо запустить асинхронный поток и передать ему адрес документа для парсинга и токен доступа.

new MyAsync().execute(adress, geting\_intent.getStringExtra("cookie"));

В данном потоке выполняется переопределенный метод doInBackground(). В нем создаётся объект класса HtmlHelper в котором выполняется получение документа и разбор его на лексемы (тэги).

HtmlHelper hh = new HtmlHelper(params[0], params[1]);

List<TagNode> tabels = hh.getTabelByClass("tdall2");

Далее из полученных списков элементов создаются объекты по классам.

Tabel tabel = new Tabel();

Spisok spisok=new Spisok();

Все последующие взаимодействия объектов с активностью происходят в переопределенном методе onPostExecute().

# Разработка графического интерфейса

Основная цель данного приложения – обеспечение доступа к некоторым функциям личного кабинета НГТУ для преподавателя или для студента.

Основной функционал приложения представлен в схеме.

Авторизация

**Главный Экран**

+Просмотр расписания

+ Просмотр учебного плана

+ Просмотр успеваемости

**Главный Экран**

+ Просмотр показателей деятельности

+ Просмотр списка групп

+ Отправка информационных сообщений студентам

Пользователь

Студент

Преподаватель

Схема 1. Основной функционал приложения.

## Описание возможностей приложения

### Просмотр расписания занятий.

Каждый авторизованный пользователь имеет возможность просматривать свое расписание на текущую неделю.

### Просмотр и редактирование учебного плана.

В ходе процесса обучения студентам необходимо делать выбор предметов на следующий семестр (помимо обязательных предметов существуют предметы по выбору) в определенный срок. Программа предоставляет возможность выбора предмета и просмотр учебного плана для уточнения количества часов и выполняемых работ в течении семестра.

### Просмотр успеваемости (Студент).

Во время обучения у студентов появляется необходимость просмотреть свою успеваемость на наличие долгов или среднего балла.

### Просмотр списка групп.

Преподавателям необходимо иметь список группы студентов для отметки присутствующих на занятиях студентов.

### Отправка информационных сообщений студентам.

Существует необходимость оперативно информировать студентов на почту или в личный кабинет. Программа позволяет отправлять сообщения в личный кабинет студента, продублировав их на личную почту студента.

### Просмотр показателей деятельности

Для преподавателя существует возможность просмотреть показатели деятельности по университету.

## Варианты использования

### Авторизация пользователя

Для получения возможности использования приложения необходимо авторизоваться в системе.

Сценарий:

1. Ввод логина и пароля
2. Нажатие кнопки «Вход»

### Просмотр расписания занятий

В окне приложения отображается расписание на текущую неделю.

Сценарий:

1. Пользователь пролистывает расписание, на текущую неделю находя необходимый день недели
2. По нажатию на кнопку «Назад», пользователь возвращается на главный экран.

### Просмотр успеваемости (студент).

С помощью приложения должна быть возможность просмотреть успеваемость на текущий момент.

Сценарий:

1. Студент просматривает свою успеваемость.
2. По нажатию на кнопку «Назад», студент возвращается на главный экран.

### Просмотр списка групп.

С помощью данного приложения преподаватель может просмотреть список групп.

Сценарий:

1. Преподаватель выбирает факультет из выпадающего списка.
2. Из второго выпадающего списка преподаватель выбирает отделение, в котором хочет просмотреть список групп
3. Из третьего выпадающего списка преподаватель выбирает курс но котором находится искомая группа
4. Из четвертого списка преподаватель выбирает конкретную группу для просмотра её списка студентов
5. После выбора всех параметров преподаватель нажимает кнопку «Показать» и переходит на экран, на котором отображен список студентов выбранной группы.
6. По нажатию на кнопку «Назад», преподаватель возвращается на экран с выпадающими списками.
7. По нажатию на кнопку «Назад», на главный экран.

### Отправка информационных сообщений студентам.

С помощью данного приложения преподаватель может отправлять сообщения конкретному студенту или всей группе сразу.

Сценарий:

1. На главном экране нажав на кнопку «Сообщения» преподаватель переходит на экран с формой для отправки сообщений.
2. В поле для ввода преподаватель вводит полное наименование группы.
3. Нажав на выпадающий преподавателю предоставляется выбор «Отправить письмо всем» или отправить письмо, конкретному студенту кликнув на него в выпадающем списке.
4. Выбрав получателя(ей) преподаватель вводит тему письма и его текст.
5. По нажатию на кнопку «Отправить» письмо отправляется получателю(ям) и преподаватель возвращается на главный экран.

### Просмотр учебного плана.

С помощью данного приложения студент способен просматривать свой учебный план и выбирать предметы по выбору.

Сценарий:

1. По нажатию на кнопку «Учебный план» студент переходит на экран с таблицами из учебного плана.

По нажатию на кнопку «Назад», студент возвращается на главный экран.

### Просмотр показателей деятельности

С помощью данного приложения преподаватель может просмотреть показатели деятельности, как по всему вузу, так и по определенному факультету.

Сценарий:

1. По нажатию на кнопку «Показатели деятельности» преподаватель переходит на экран, содержащий выпадающий список с факультетами и с таблицей показателей.
2. При выборе факультета из выпадающего списка демонстрируется таблица по выбранному факультету.
3. При нажатии на кнопку «Назад» преподаватель возвращается на главный экран.

Изображения с видом графического интерфейса представлены в приложении А.