**Федеральное государственное образовательное бюджетное учреждение  
высшего образования**

**«Финансовый университет при Правительстве Российской Федерации»**

**КОЛЛЕДЖ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ**

**ПМ.03 Участие в интеграции программных модулей**

**Группа: 4ПКС-115**

**УТВЕРЖДАЮ**

**Председатель цикловой комиссии**

**программирования и баз данных**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Пестов А.И./**

**\_\_\_\_.\_\_\_\_. 2018**

**ПРОЕКТ КУРСОВОЙ**

**На тему: Программное приложение по рассылке расписания занятий студентам по SMS на языке С#**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**Руководитель курсового проекта**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Морозова М.В./**

**Исполнитель курсового проекта**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Деменчук Г.М./**

**Оценка за проект:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_.\_\_\_\_.2018**

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc523600377)

[1 ОБЩИЙ РАЗДЕЛ 4](#_Toc523600378)

[1.1 Системные требования 4](#_Toc523600379)

[1.2 Характеристика системы программирования 4](#_Toc523600380)

[2 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 5](#_Toc523600381)

[2.1 Предпроектное исследование предметной области 5](#_Toc523600382)

[2.2 Анализ требований и определение спецификация программного обеспечения 5](#_Toc523600383)

[2.3 Проектирование программного обеспечения 5](#_Toc523600384)

[2.4 Разработка пользовательских интерфейсов программного обеспечения 5](#_Toc523600385)

[2.5 Тестирование и отладка программного обеспечения 5](#_Toc523600386)

[3 РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ 6](#_Toc523600387)

[3.1 Руководство программиста 6](#_Toc523600388)

[3.2 Руководство пользователя 6](#_Toc523600389)

[ЗАКЛЮЧЕНИЕ 7](#_Toc523600390)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 8](#_Toc523600391)

[ПРИЛОЖЕНИЕ А 9](#_Toc523600392)

[ПРИЛОЖЕНИЕ Б 10](#_Toc523600393)

# ВВЕДЕНИЕ

На сегодняшний день оповещения являются неотъемлемой частью нашей жизни. Все оповещения по достаточно просто разделить по приоритету:

* Телефонные вызовы;
* SMS-оповещения;
* Оповещения от различных мессенджеров;

Как мы видим, телефонные вызовы являются наиболее приоритетными, но они требуют немедленного реагирования от пользователя и заставляют прерываться от его деятельности. SMS-оповещения не требуют немедленной реакции, при этом его получат болеее широкое количество пользователей по сравнению с оповещениями мессенджеров. По состоянию на 2018 год по данным Pew Research Center 59% опрошенных взрослых в мире являются владельцами смартфонов, а 31% пользуются простыми мобильными телефонами, единственный недостаток заключается в том, что данный вид оповещений является платным. При рассмотрений оповещений от мессенджеров мы игнорируем людей, у которых простые (кнопочные) мобильные телефоны, но при этом данный вид оповещений является бесплатным.

Также все чаще мы можем наблюдать, что технологии распознавания текста и компьютерного зрения все сильнее входят в нашу повседневную жизнь: машины с автопилотом, промышленные роботы, системы видеонаблюдения, некоторые флагманские модели смартфонов.

В данной работе мы как раз и будем использовать совокупность всех технологий выше: управление SMS-оповещениями с помощью клиента C# и мессенджера Telegram, анализ изображения с матрицей пар, распознавание текста.

# 1 ОБЩИЙ РАЗДЕЛ

## 1.1 Системные требования

Рекомендованные системные требования:

Выпуск Windows: Windows 7, Корпорация Майкрософт (Microsoft

Corporation), 2008г. Либо более новая версия.

Требования к системе:

* Процессор: Intel (R) Core (TM) i3-7100 (3.40GHz);
* Установленная память (ОЗУ): 6,00 ГБ;
* Тип системы: 64-разрядная операционная система, процессор х64;
* Клавиатура;
* USB-мышь либо наличие сенсорной панели (touchpad).

## 1.2 Характеристика системы программирования

Данная программа была написана на объектно-ориентированном языке программирования C# с помощью MS Visual Studio 13 Community с множеством различных встроенных библиотек и функций, удобным для пользователя интерфейсом и наличием отладчика кода.

Программирование выполнялось на языке C#, тип проекта – графическое окно.

C# — язык программирования, сочетающий объектно-ориентированные и контекстно-ориентированные концепции. Разработан в 1998—2001 годах группой инженеров под руководством Андерсa Хейлсбергa в компании Microsoft как основной язык разработки приложений для платформы Microsoft .NET. Компилятор с C# входит в стандартную установку самой .NET, поэтому программы на нём можно создавать и компилировать даже без инструментальных средств вроде Visual Studio.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет строгую статическую типизацию, поддерживает полиморфизм, перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения, комментарии в формате XML. Переняв многое от своих предшественников — языков C++, Delphi, Modula и Smalltalk — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем: так, C# не поддерживает множественное наследование классов (в отличие от C++) или вывода типов (в отличие от Haskell).

# 2 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

## 2.1 Предпроектное исследование предметной области

При разработке программного решения наиболее остро встает вопрос об источнике получения расписания учебного заведения и его последующего анализа. Далее будут рассмотрены источники расписания именно для Колледжа Информатики и Программирования при ФУ РФ. На данный момент существует три наиболее рациональных источника данных:

* Электронный журнал колледжа по адресу kip.eljur.ru
* Расписание в виде PDF на сайте kip.fa.ru
* Физическая распечатка PDF в колледже на стойке информатики

Сразу же мы можем исключить парсинг физической распечатки PDF по фотографии с телефона т.к. для подобного анализа изображений, требуется достаточно много условий:

* Большое количество операций по препроцессингу изображений: от банального размытия по гауссу, заканчивая вычислением пропорций между контурами ячеек с названием пары от ячеек с названием группы и выправления горизонта с помощью детектора границ Кэнни, определения прямых линий по преобразованию Хафа и вычисления результирующего угла между перпендикулярами, проходящими через центр изображения.
* Более тонкая настройка файла rus.traineddata для анализа кириллического шрифта с помощью open-source технологии Google Tesseract OCR;
* Возможная визуальная ограниченность для камеры телефона (лист бумаги находится за стеклом и отсвечивает, вспышка);
* При архитектуре standalone-приложения будет огромнейшая нагрузка на аппаратные средства мобильного устройства и как следствие, его быстрый разряд, при клиент-серверном решении требуется активное интернет-соединение;
* Большой процент риска человеческого фактора (лист бумаги может кто-то подменить в шутку, украсть и т д)

Электронный журнал колледжа, именно АИС «Электронный журнал» мы также не можем рассматривать в качестве достоверного источника информации о расписании т.к для оформления процедуры официального досупа к API системы требуется достаточно большое количество времени, а в противном случает мы нарушаем лицензию использования АИС и коэффициент рисков и времени разработки достаточно высок. В данном способе возможно три направления, расположенных в порядке «легальности»:

* Направление официального запроса к АИС с указанием целей, для которых мы планируем использовать доступ к API, а также информацией о нас, далее простой парсинг json по devkey на стороне клиента C#;
* Эмуляция браузера с поддержкой javascript и парсинг тегов HTML-документа с последующую их конвертацию в JSON, который будет отдаваться WEB API нашего сервера;
* Эмуляция клиента мобильного приложения IOS/Android и парсинг JSON/XML, который мы получили от backend-сервера АИС. Можно подменить headers непосредственно на клиенте C#, можно через прослойку в виде нашего backend-сервера, который может отвечать за прямое взаимодействие с сервером АИС.

Дать официальный запрос мы не можем т.к. аудитория возможных пользователей нашего сервиса слишком мала и время достаточно сильно ограничено.

Второй способ является сбалансированным, но самым нестабильным т.к. при изменении разметки страницы весь алгоритм парсинга перестает выполнять свою работу. Существенным плюсом является возможность узнавать расписание любых групп (нет ассоциации пользователь -> группа, в отличии от мобильного приложения), за анализ страницы может отвечать библиотека BeautifulSoup.

Третий способ представляет из себя подмену https-сертификата приложения для осуществления атаки по типу MITM (человек посередине) т.к. у большинства приложений отсутствует проверка на подмену сертификата, кроме банковских. Возможно получить только расписание своей группы т.к. есть четкая ассоциация токен -> группа. Токен работает ограниченный промежуток времени. За получение неправомерного доступа к компьютерной информации, а также создание компьютерных программ, заведомо предназначенных для осуществления несанкционированного доступа к информации, установлена ответственность в соответствии со статьями 272 и 273 Уголовного кодекса Российской Федерации. Способ не подходит.

Последним источником информации о расписании остается расписание занятий на сайте Колледжа Информатики и Программирования при ФУ РФ kip.fa.ru. Также есть свои преимущества и недостатки.

Преимущества:

* Достаточно прост процесс получения исходного файла для обработки: обыкновенный get-запрос по статической ссылке;
* Возможно применить гибридный парсинг исходного файла как растрового изображения в формате .png или .jpg с использованием технологии OpenCV и Tesseract OCR, так и векторного файла .PDF с символами, что в итоге многократно уменьшит риск каких-либо синтаксических ошибок в тексте расписания;
* Не требует глубокого препроцессинга исходного изображения и не нарушает какие-либо права, в отличии от вышеописанных способов.

Недостатки:

* В любой момент Финансовый Университет может отказаться от него, закрыть доступ и полностью перейти на АИС «Электронный журнал», что и произошло в процессе разработки данного решения;
* Иногда обновленное расписание может выкладываться с задержкой;
* Извлечение данных даже с векторного PDF является достаточно трудоемкой задачей;
* Возможный риск человеческого фактора.

В данном способе также существует несколько вариантов действий:

* Парсинг растрового изображения в формате .png или .jpg;
* Парсинг векторного файла .pdf с символами;
* Совокупный парсинг и векторного файла и растрового изображения.

Наиболее рациональным способом получения расписания является совокупность всех данных из всех трех вышеизложенных источников (сайт Колледжа Информатики и Программирования, АИС «Электронный журнал», физическая распечатка), но для банального уменьшения рисков и уменьшения времени работы мы остановимся на сайте колледжа.

В качестве анализа данных был выбран способ распознавания данных из растрового изображения .png т.к. был интерес в конечном качестве распознавания кириллических шрифтов и насколько сильно он отличался бы от векторного способа.

## 2.2 Анализ требований и определение спецификация программного обеспечения

Диаграмма вариантов использования представлена на рисунке 2.1, и состоит из действий, совершаемых пользователем в программе.

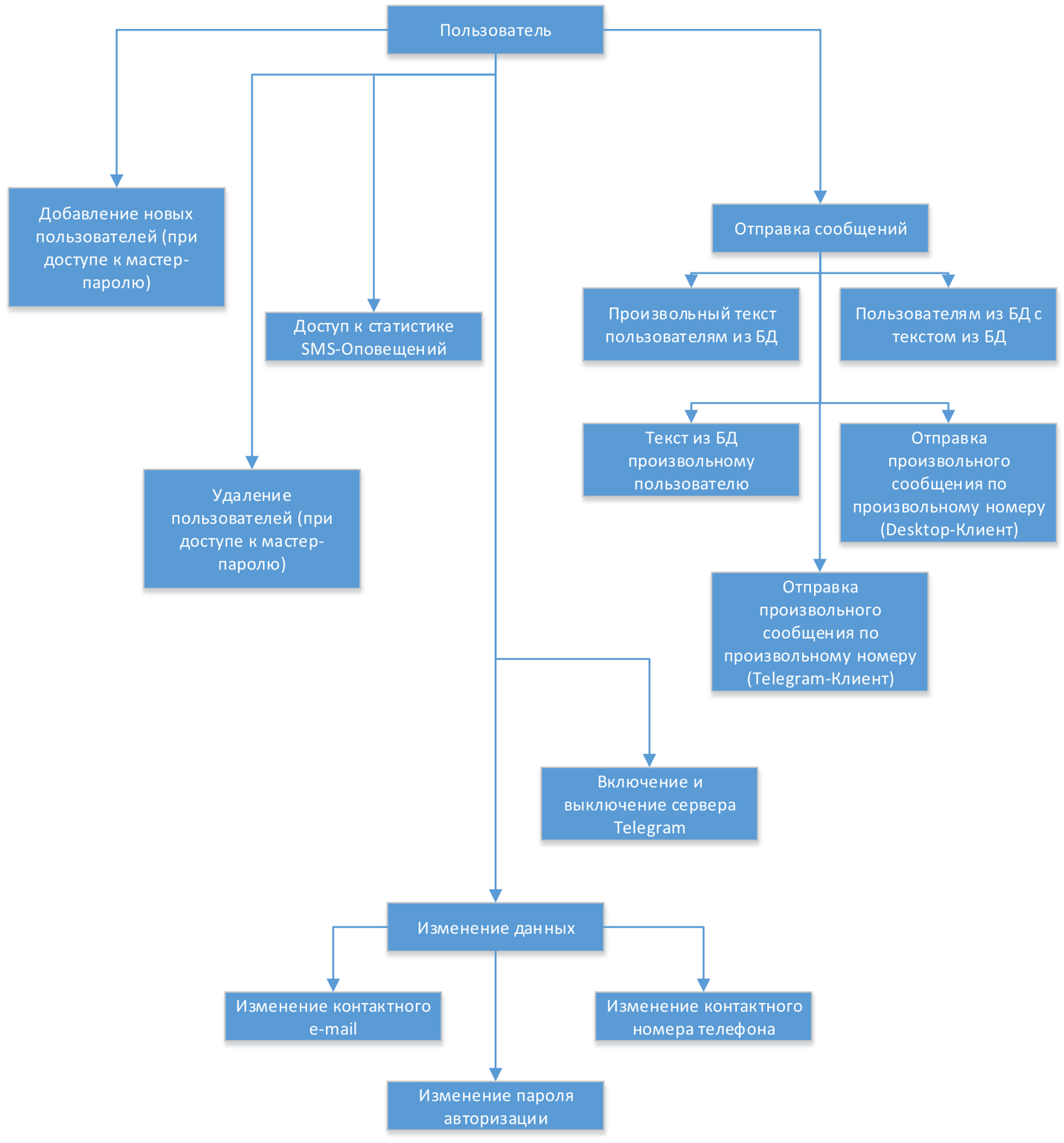


Рис. 2.1 – Диаграмма вариантов использования

В клиенте С# для рассылки расписания по SMS у пользователя есть множество возможностей, однако стоит отдельно рассмотреть ключевую и её параметр – отправку оповещений:

* Отправка SMS-оповещений пользователям из БД с расписанием из БД;
* Отправка произвольного текста пользователям из БД;
* Отправка текста из БД произвольному пользователю с выбором группы и последующем запоминанием его в БД (при надобности);
* Отправка произвольного сообщения по произвольному номеру телефона с использованием desktop-клиента;
* Отправка произвольного сообщения по произвольному номеру телефона с использованием бота в мессенджере Telegram (который хостится на клиенте C#)

Также немаловажной для пользователя является возможность доступа к статистике отправленных SMS-оповещений, где содержится следующая информация:

* Телефон, на который отправлялось сообщение в формате 7XXXXXXXXXX
* Статус сообщения (в очереди, доставлено, не доставлено, передано, ожидание статуса сообщения, сообщение отклонено, на модерации);
* Стоимость переданного сообщения (одно SMS на кириллице вмещает в себя 70 символов, если больше – учитывается как два и более сообщений соответственно);
* Текст исходного сообщения

У пользователя существует возможность изменения своих данных для авторизации в клиенте, а именно:

* Изменение своего контактного email, используемого в качестве логина пользователя
* Изменение своего пароля авторизации
* Изменение своего контактного номера телефона, который также может использоваться в качестве логина пользователя

При доступе к мастер-паролю приложения, у пользователя появляется возможность управления другими пользователями, а именно удаление и добавление новых (т.е. пользователь с мастер-паролем становится администратором приложения).

При добавлении нового пользователя администратору необходимо ввести следующие данные:

* Имя пользователя для отображении в главном меню программы;
* E-mail с последующим его подтверждением для использования в качестве логина системы;
* Номер телефона с последующим его подтверждением для использования в качестве альтернативного логина системы;
* Пароль создаваемого пользователя для авторизации в приложении;
* Мастер-пароль для подтверждения прав на осуществляемое действие.

При удалении пользователя администратору необходимо ввести следующие данные:

* Логин пользователя: может быть электронной почтой или номером телефона удаляемого пользователя;
* Мастер-пароль для подтверждения прав на осуществляемое действие.

Также существует возможность включения и отключения сервера Telegram. В главном меню приложения есть отображение:

* Баланса счета сервиса SMS-оповещений SMSAero
* Имени текущего пользователя

## 2.3 Проектирование программного обеспечения

В качестве основной архитектуры будем использовать клиент-серверное решение, как наиболее сбалансированное:

* Сервер парсинга и обработки pdf, который отдает json;
* C# клиент с возможностью развертки сервера Telegram;
* Автономный клиент Telegram;
* Автономный клиент VK.

Рассмотрим более подробно архитектуру каждого решения. На backend-сервере будет использоваться следующие технологии и библиотеки:

* Flask Web API - фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2;
* OpenCV - библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++;
* Tesseract OCR - свободная компьютерная программа для распознавания текстов;
* Pdf2image - модуль Python, обертка утилиты pdftoppm для преобразования PDF в объект PIL Image;
* DANTED SOCKS5 Proxy - состоит из сервера SOCKS и клиента SOCKS, реализующего RFC 1928 и связанные с ним стандарты;
* MySQL - свободная реляционная система управления базами данных;
* GNU Screen - свободная консольная утилита-мультиплексор, предоставляющая пользователю доступ к нескольким сессиям в рамках одной сессии;

Алгоритм парсинга:

1. Какой-либо клиент делает get-запрос с параметром pdf\_id по ссылке вида “/api/v1/parse\_json/” для запуска процесса обновления расписания с помощью flask API, фреймворк вызывает функцию parse\_json, которая отдает “True”, если удалось создать отдельный поток с вызовом функции ParseKIPTT, иначе возвращает “False”. Далее будут описываться все действия, которые будут выполняться в функции ParseKIPTT. В данной архитектуре flask запущен через GNU screen, хотя рациональнее использовать pm2.
2. Вызывается функция get\_document, где передается параметр pdf\_id, происходит скачивание и побайтовая конвертация .PDF в формат .png (.jpg не используем т.к. при использовании логики библиотеки openjpeg возможно получение артефактов компрессии, что плохо влияет на алгоритмы предварительной обработки изображения) и последующее его сохранение.
3. Начинается работа с OpenCV. Открываем сконвертированный файл, преобразуем его в черно-белый формат (для 8-битного одноканального изображения), делаем размытие по стандартному отклонению распределения Гаусса по sigmaX = 3 и sigmaY = 3 направлению, определяем границы с помощью детектора границ Кенни с порогом минимума 10 и максимума 250.
4. Находим все контуры на изображении, размещаем их в иерархии вложенных контуров c аппроксимацией по сжатию горизонтальных, вертикальных и диагональных сегментов и оставлением только их конечных точек.

FRONTEND:

Sqlite3

Будем использовать

## 2.4 Разработка пользовательских интерфейсов программного обеспечения

## 2.5 Тестирование и отладка программного обеспечения

# 3 РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ

## 3.1 Руководство программиста

## 3.2 Руководство пользователя

# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

# ПРИЛОЖЕНИЕ А

# ПРИЛОЖЕНИЕ Б