**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](#_Toc531274850)

[1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ 4](#_Toc531274851)

[1.1 Предпроектное исследование предметной области 4](#_Toc531274852)

[1.2 Среда и язык программирования 11](#_Toc531274853)

[1.3 Анализ требований и определение спецификация ПО 11](#_Toc531274854)

[2 СПЕЦИАЛЬНЫЙ РАЗДЕЛ 15](#_Toc531274855)

[2.1 Проектирование программного обеспечения 15](#_Toc531274856)

[2.2 Разработка пользовательских интерфейсов ПО 26](#_Toc531274857)

[3 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 60](#_Toc531274858)

[3.1 Технологии, применяемые при разработке ПО 60](#_Toc531274859)

[3.2 Тестирование и отладка программного обеспечения 60](#_Toc531274860)

[4 РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ 61](#_Toc531274861)

[4.1 Руководство программиста 61](#_Toc531274862)

[3.2 Руководство пользователя 67](#_Toc531274863)

[ЗАКЛЮЧЕНИЕ 68](#_Toc531274864)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 69](#_Toc531274865)

[ПРИЛОЖЕНИЕ А 70](#_Toc531274866)

[ПРИЛОЖЕНИЕ Б 71](#_Toc531274867)

# 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 с функцией распознавания лица.

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 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

## 1.1 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.

Последним шагом являлась задача выбора сервиса SMS-рассылок. На текущий момент наиболее популярными сервисами SMS-гейтов являются:

* twilio.com;
* plivo.com;
* nexmo.com;
* smsaero.ru;
* sms.ru;
* smsc.ru;

В числе выбранных оказались три популярных отечественных сервиса и три зарубежных. Среди них необходимо выбрать тот, который отвечает следующим критериям:

* Доступная, понятная, и подробная документация;
* Ключевые возможности сервиса, наличие SDK для быстрой интеграции в проекты и открытое и понятное API для взаимодействия;
* Поддержка клиента, обратная связь и скорость реагирования;

Первый и важный пункт: насколько удобно работать с сервисом и интегрировать его в любой проект.

Таблица 1. Сравнение документации сервисов

|  |  |  |
| --- | --- | --- |
| Документация | Подробная | Краткая |
| Twilio | + |  |
| Plivo | + |  |
| Nexmo | + |  |
| SMSAero | + |  |
| SMS.ru |  | + |
| SMSC | + |  |

Как мы можем заметить, у большинства сервисов присутствует подробная документация относительно своих функций. В некоторых документациях таких как, к примеру, Twilio и Nexmo присутствуют готовые реализации на нескольких языках программирования.

Вторым важным аспектом является «А что собственно может сервис?». Для многих мобильных проектов важно наличие SDK для интеграции, более быстрой разработки голосовых сервисов и подобных решений. В нашей архитектуре приложения мы будем использовать сервис только для рассылок и/или уведомлений.

Таблица 2. Сравнение возможностей сервисов

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Сервис | Модули для CMS: | SDK | Песочница | Звонки | Текст в речь | SIP | Обратный звонок | Рассылка |
| Twilio | + |  | + | + | + | + | + |  |
| Plivo | + | + | + | + | + | + | + |  |
| Nexmo |  |  | + | + | + | + | + |  |
| SMSAero | + | + |  |  |  |  |  | + |
| SMS.ru | + |  |  |  |  |  |  | + |
| SMSC | + |  |  |  |  |  |  | + |

Как видно из данной таблицы, российский рынок SMS-гейтов нацелен в первую очередь на массовые рассылки сообщений и уведомления пользователей. Тогда как западный рынок в первую очередь создает платформы для сервисного использования SMS. Такие вещи как мобильный SDK — редкость на данном рынке, присутствует только у Plivo и SMSAero.

Решающей является роль технической поддержки и её компетентность, скорость реагирования на запрос и оперативность устранения проблемы.

Таблица 3. Сравнение способов поддержки клиентов в сервисах

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Сервис | Чат | Форум | Почта | Скайп | Телефон |
| Twilio | + | + | + |  |  |
| Plivo |  | + | + |  |  |
| Nexmo |  | + | + |  | + |
| SMSAero | + | + | + | + | + |
| SMS.ru |  |  | + |  | + |
| SMSC | + | + | + | + | + |

Субъективно можно отметить, что сервисное ориентирование западных сервисов сказывается и здесь: российские реагируют гораздо быстрее и охотнее, нежели их западные конкуренты. Связано это в первую очередь с тем, что на российском рынке более активно построен процесс продаж этих услуг. Западные сервисы отвечают в основном по созданным тикетам, тогда как российским можно запросто написать или позвонить в скайп/в чат прямо на сайте. Наибольшее количество способов поддержки мы видим у SMSAero.

В результате анализа SMS-гейтов был выбран сервис SMSAero как наиболее рациональное соотношение между возможностями и уровнем поддержки, а также способностью быстрой интеграции в решение путем добавления класса с методами API в свое решение.

## 1.2 Среда и язык 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).

## 1.3 Анализ требований и определение спецификация ПО

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

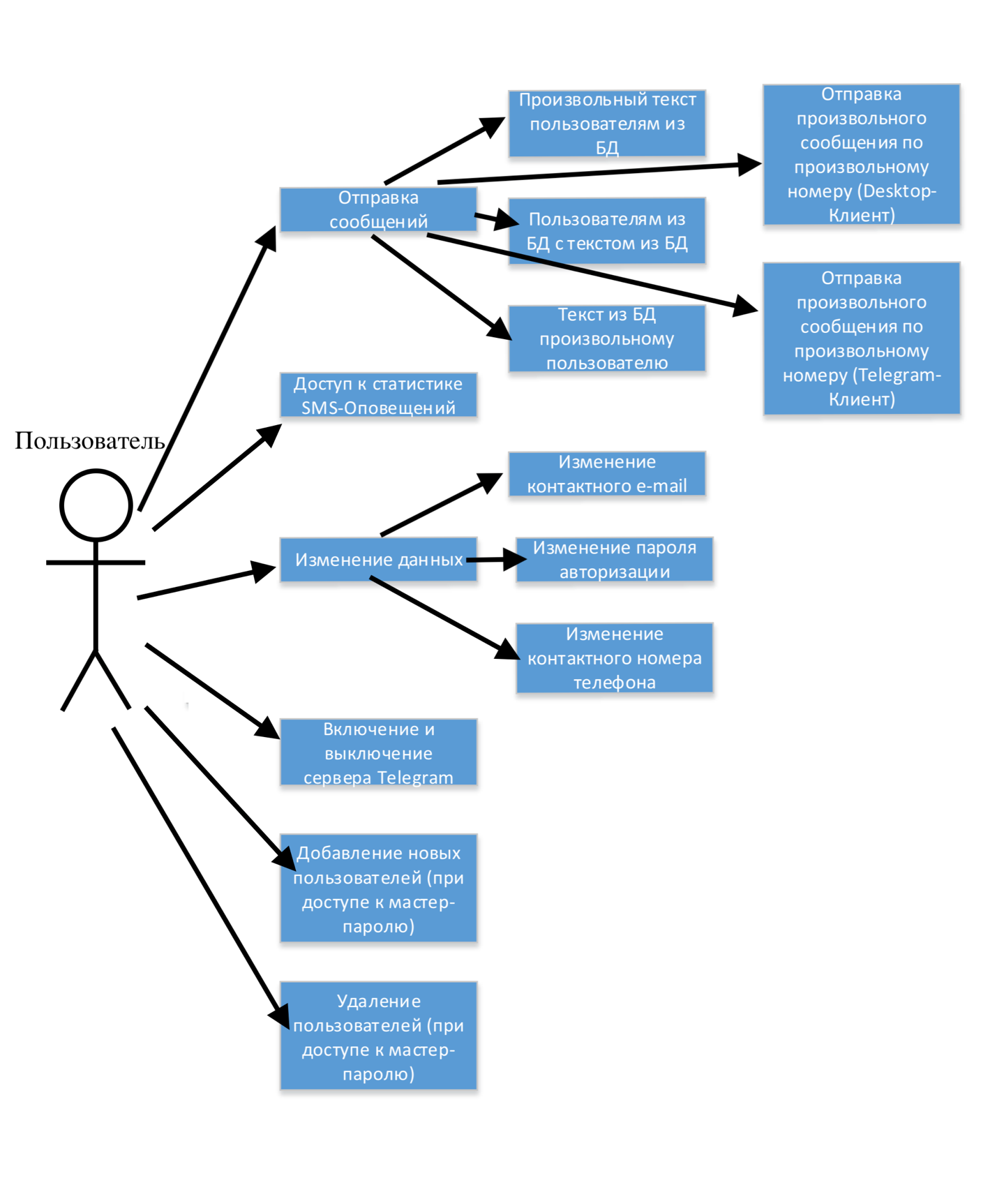


Рисунок 1 – Диаграмма вариантов использования

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

Контекстная диаграмма потоков данных представлена на рисунке 2, и представляет собой взаимодействие пользователя и приложения, связанных потоками данных.

На диаграмме можно увидеть, что от пользователя к программе поступают такие действия, как:

- заполнение форм;

- выбор пункта меню;

- выбор метода расчета оценки бизнеса.

Программа, в свою очередь, возвращает пользователю документ Word, содержащий готовый бизнес-план, формы для ввода, итоги расчетов и само меню.

Untitled Diagram2.png

Рисунок 2 – Контекстная диаграмма потоков данных

Детализированная диаграмма потоков данных является более подробной и представлена на рисунке 3.

На ней можно заметить, как при выводе меню в программу загружается документ Word, Access и фоновые изображения. Далее передаются введенные данные для вычислений, и применяется формула, в зависимости от выбранного пункта. Как итог, выводятся полученные расчеты либо график и документ Word .

Untitled Diagram3.png

Рисунок 3 – Детализированная диаграмма потоков данных

1.3.2 Функциональная диаграмма

Функциональная диаграмма - диаграмма, отражающая взаимосвязи функций разрабатываемого программного обеспечения

Контекстная функциональная диаграмма представлена на рисунке 4.

На ней можно увидеть, что на вход программе подаются:

- фоновые изображения;

- документ Word;

- два документа Access;

- введенные пользователем данные.

Программа опирается на:

- правила оформления бизнес-плана;

- формулы для анализа бизнеса;

Управляет программой пользователь.

А на выходе у программы:

- объект списка идей;

- стоимость бизнеса;

- срок окупаемости бизнеса;

- график нахождения точки безубыточности;

- минимальная прибыль для окупаемости предприятия;

- минимальный объем продаж для окупаемости предприятия;

- объект списка идей.

Untitled Diagram4(1).png

Рисунок 4 – Контекстная функциональная диаграмма

Детализированная функциональная диаграмма представлена на рисунке 5.

Данная диаграмма содержит основные функции программы:

- вывод меню;

- вывод выбранного пункта меню;

- обработка введенных данных;

- вывод одного из списков;

- расчет в зависимости от выбранного пункта.

На вход программе подается:

- документ Word;

- документ Access, содержащий список и описание идей для бизнеса;

- фоновые изображения;

На выходе пользователь получит:

- итоговые расчеты;

- график;

- документ Word, содержащий подробный бизнес-план;

- список книг;

- список идей для бизнеса.

Untitled Diagram5.pngРисунок 5 – Детализированная функциональная диаграмма

# 2 СПЕЦИАЛЬНЫЙ РАЗДЕЛ

## 2.1 Проектирование программного обеспечения

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

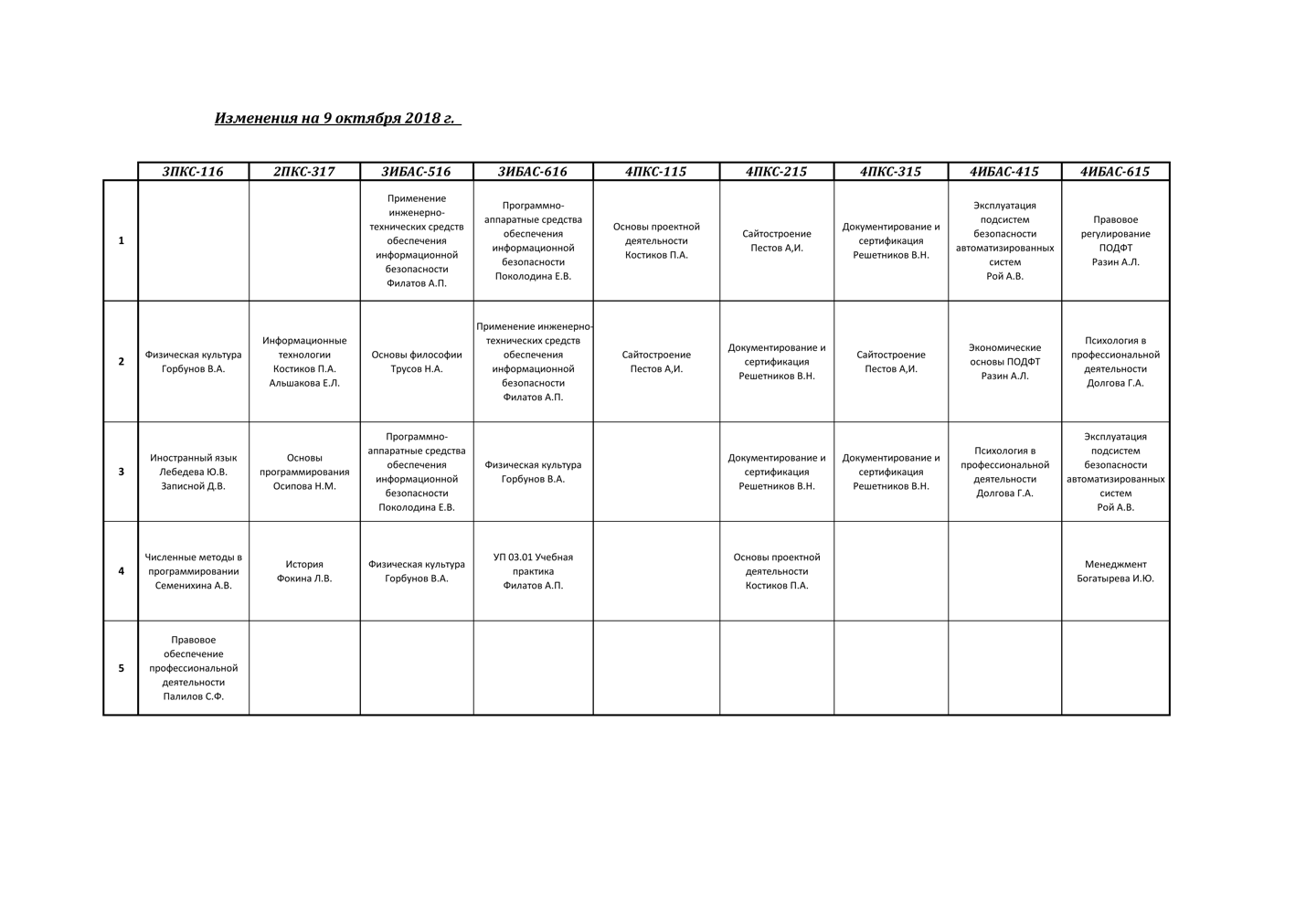
* Сервер парсинга и обработки pdf, который отдает json;
* C# клиент с возможностью развертки сервера Telegram;

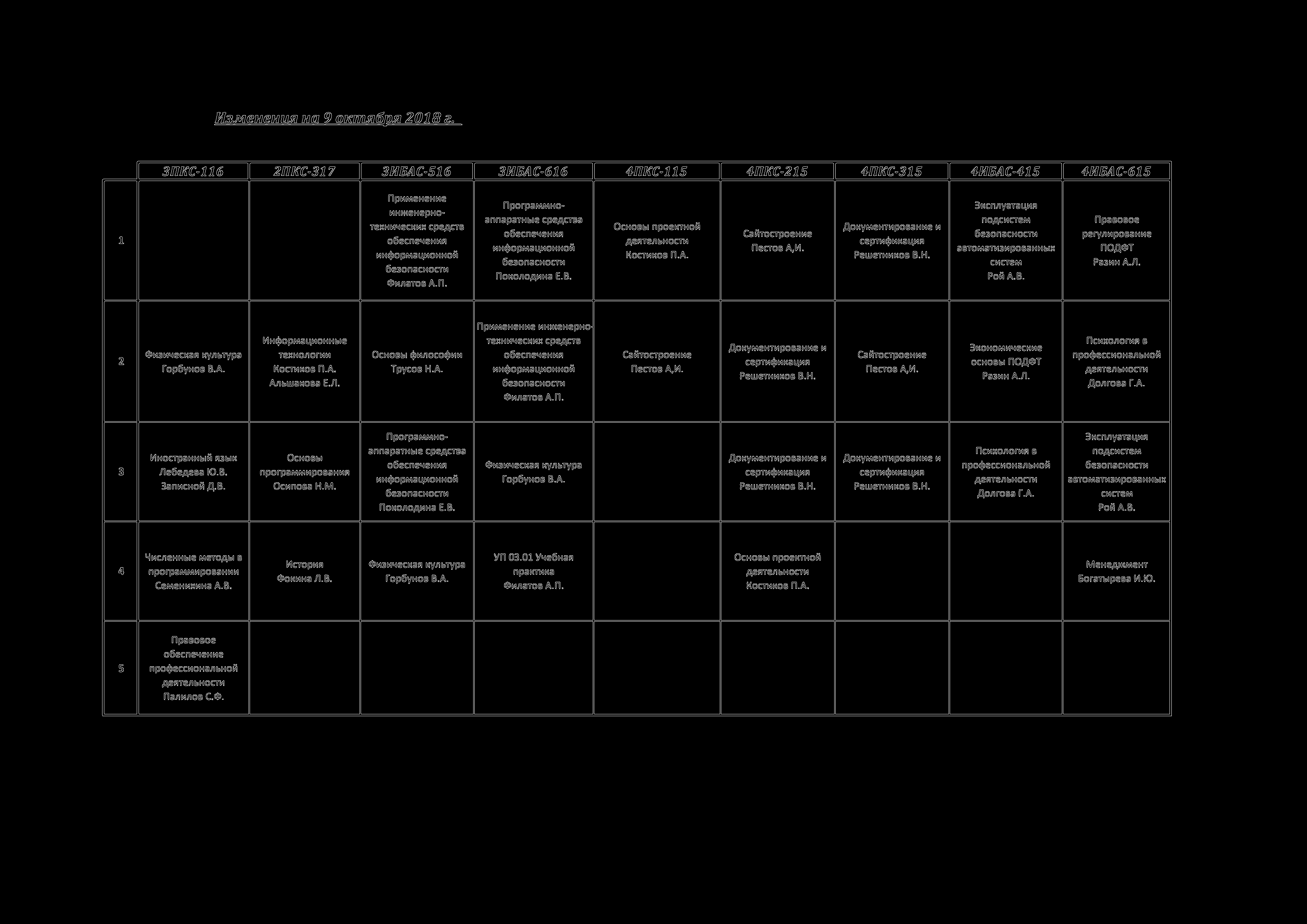
Рассмотрим более подробно архитектуру каждого решения. На backend-сервере будет использоваться следующие технологии и библиотеки:

* 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 на backend-сервере:

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 (рисунок 2.2), 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 (рисунок 2.3) с порогом минимума 10 и максимума 250.

 Рис. 2.2 – Исходное изображение

Рис. 2.3 – Определение границ с помощью детектора Кенни

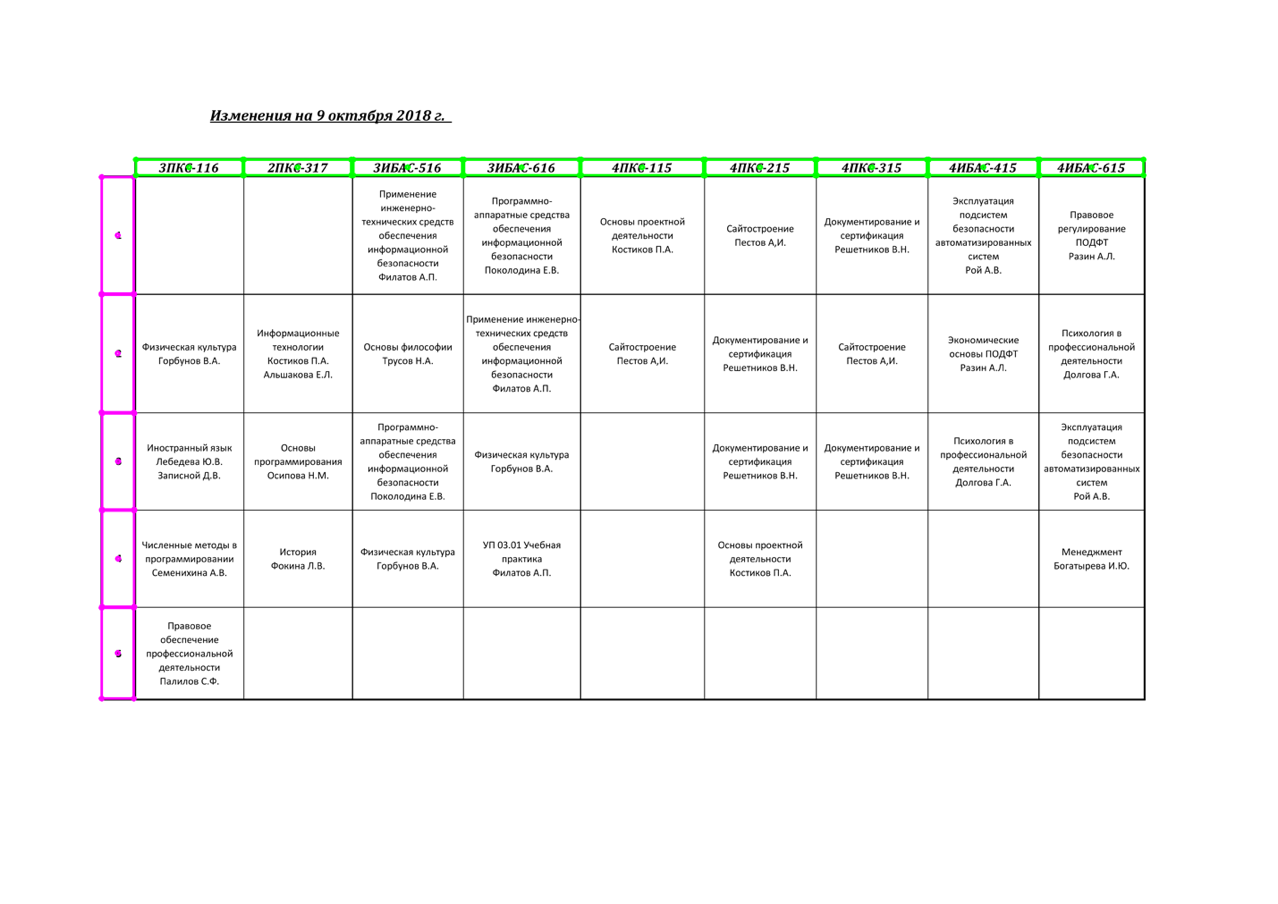
1. Находим все контуры на изображении, размещаем их в иерархии вложенных контуров c аппроксимацией по сжатию горизонтальных, вертикальных и диагональных сегментов и оставлением только их конечных точек.
2. Запускаем цикл по каждому найденному контуру и пытаемся вписать минимально возможный прямоугольник, которым можно обвести контур. Определяем координаты x и y каждого угла и вычисляем координаты центра.
3. Если ширина вписанного прямоугольника меньше 200 пикселей, но больше 30 и высота больше ширины, то такой прямоугольник является номером пары по счету (1-5) и мы отрисовываем вписанный прямоугольник, выделяем его углы и центр в виде кругов с радиусом 5 пикселей и добавляем координаты всех его углов в список leftnumber\_cell\_list.
4. Если высота вписанного прямоугольника меньше 100 пикселей, но больше 20, ширина больше высоты и абсолютная погрешность координат x, y между центром предыдущего отрисованного прямоугольника и нынешним больше 10 (функция centers\_checker), то такой прямоугольник является заголовком (названием) группы.
5. Если прямоугольник является заголовком группы, то обрезаем все изображение до координат вписанного прямоугольника с помощью функции cropimager, далее отдаем обрезанное изображение функции image\_to\_string библиотеки распознавания текста Tesseract OCR и получаем название группы в текстовом виде. Записываем в ассоциативный массив group\_text\_association центр прямоугольника в качестве ключа и распознанный текст в качестве значения. Отрисовываем вписанный прямоугольник, выделяем его углы и центр в виде кругов с радиусом 5 пикселей и добавляем координаты всех углов в список group\_cell\_list. Получаем рисунок 2.4.

Рис. 2.4 – Визуализация номеров пар и названий групп

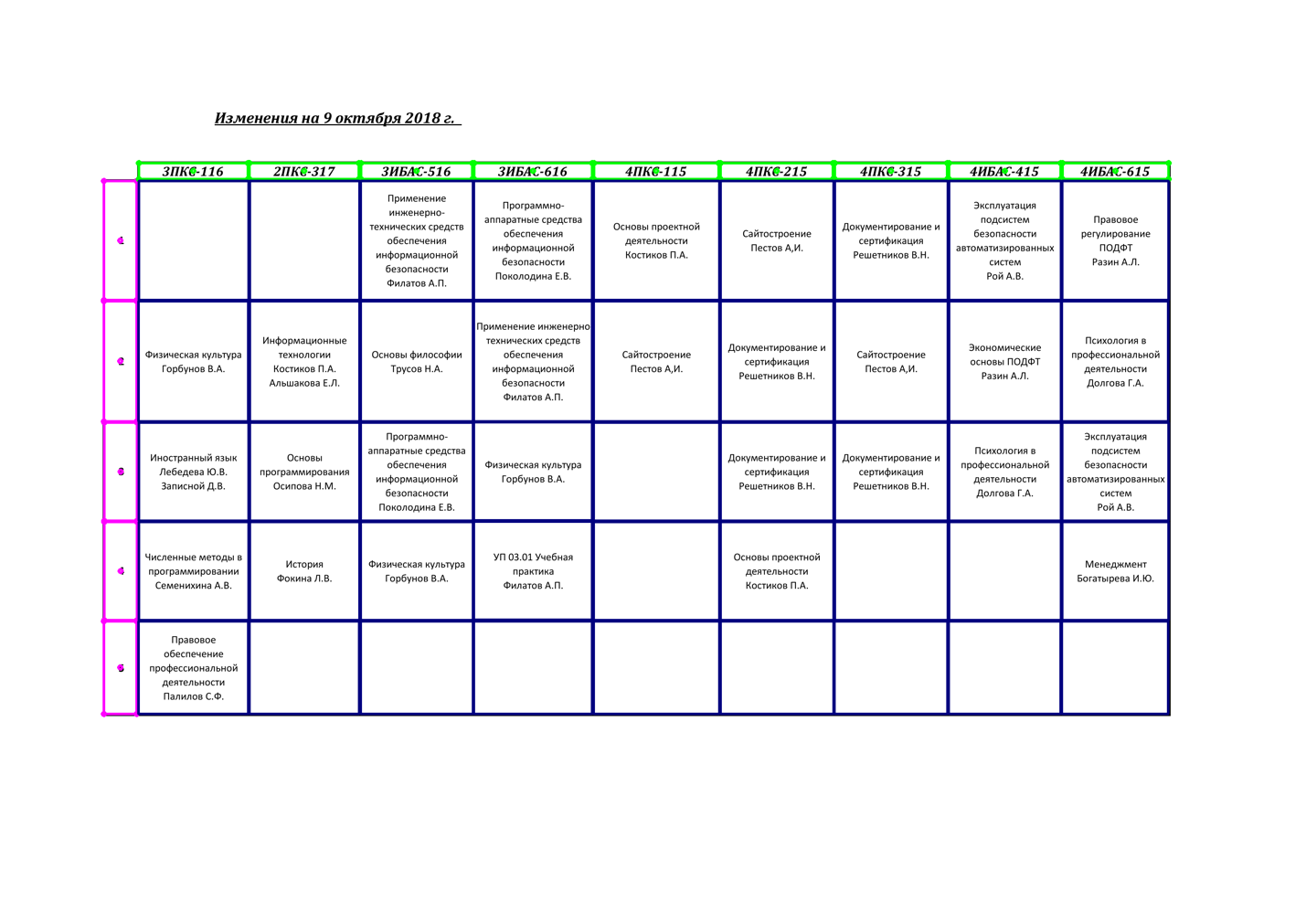
1. Выходим из цикла перебора по всем найденным контурам, сортируем ключи списка group\_text\_association по возрастанию оси абсцисс, обнуляем переменные и еще раз запускаем этот же цикл. Снова пытаемся вписать прямоугольник.
2. Если ширина и высота вписанного прямоугольника не больше 80% ширины и высоты предыдущего (функция check\_res), совокупная площадь больше 20000, но меньше 500000 пикселей, ширина и длинна больше 130 пикселей (функция AreaChecker), абсолютная погрешность координат x, y между центром предыдущего отрисованного прямоугольника и нынешним больше 10 (функция centers\_checker), контур не включает в себя контур названия групп (функция titlechecker) и контур номера пар (функция leftnumberchecker), то добавляем координаты центра в список circle\_store\_list, координаты углов в box\_store\_list, отрисовываем вписанный прямоугольник, обрезаем все изображение до координат вписанного прямоугольника с помощью функции cropimager, отдаем обрезанное изображение функции image\_to\_string Tesseract OCR, получаем название группы в текстовом виде, добавляем в ассоциативный массив center\_and\_text координаты осей x, y в качестве ключа и распознанный текст в виде значения. Добавляем координаты оси абцисс в список ColumnCheckerList и оси ординат в RowCheckerList для проверки подсчета количества объектов. Делаем инкремент глобального счетчика прямоугольников с названиями пар global\_counter.
3. После выхода из цикла получаем рисунок 2.5, считаем количество одинаковых повторений осей абцисс и ординат с помощью модуля collections в RowCheckerList и ColumnCheckerList соответственно.

Рис. 2.5 – Визуализация ячеек пар

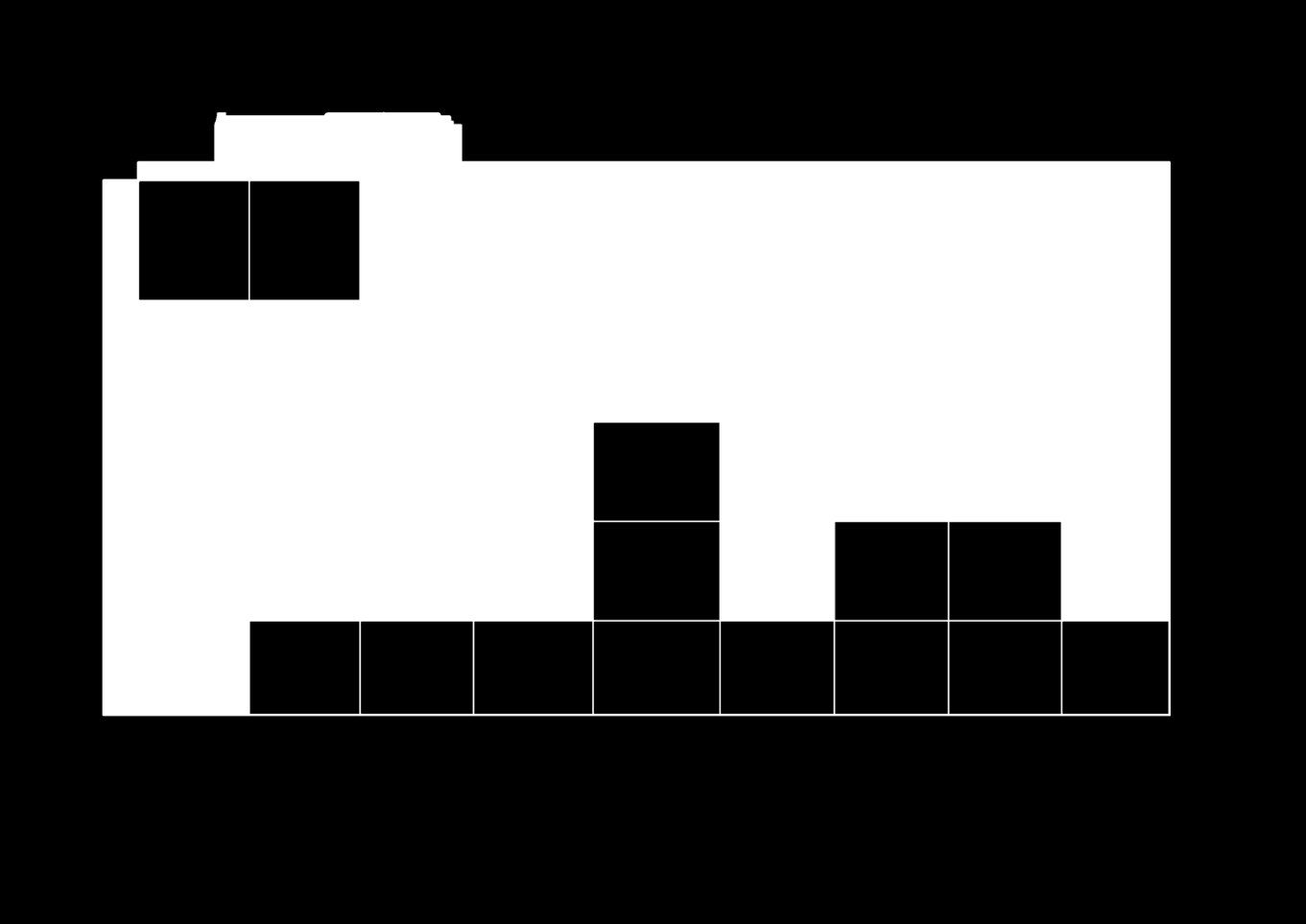
1. В структурированных объектах ищем максимальное количество повторений для определения размерности будущей матрицы.
2. Если размерность матрицы содержит такое же количество элементов, как и счетчик global\_counter, то продолжаем работу программы, в противном случае записываем результирующее изображение в директорию outputs с индексом BAD.
3. Формируем матрицу результатов с помощью функции matrix с аргументами в виде её размерности. В функции делаем обратную сортировку списка circle\_store\_list по оси y и по каждой колонке создаем вложенный список элемента matrix, в который записываем значения списка circle\_store\_list по индексу. Для каждого вложенного item (списка) в matrix делаем сортировку по оси x. Добавляем в finalmatrix элементы matrix в обратном порядке. Формирование матрицы завершено.
4. Определяем пустые ячейки (прямоугольники) с помощью функции get\_null\_values, в функции производим морфологическое преобразование изображения путем сильного расширения, а затем эрозии (рисунок 2.6). Снова находим все контуры на изображении, размещаем их в иерархии вложенных контуров c той же аппроксимацией, снова запускаем цикл по каждому контуру.

Рис. 2.6 – Изображение после морфологического преобразования

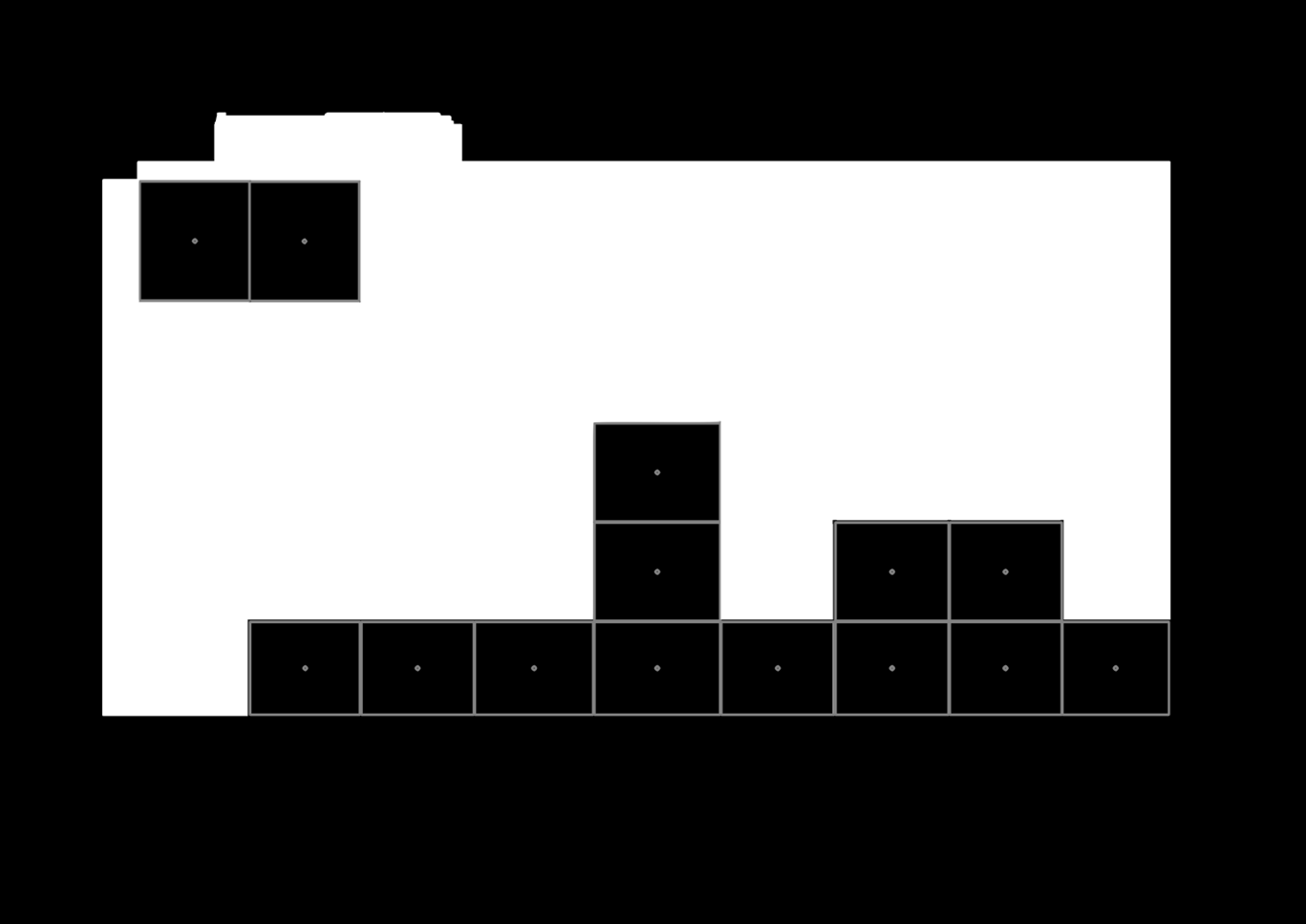
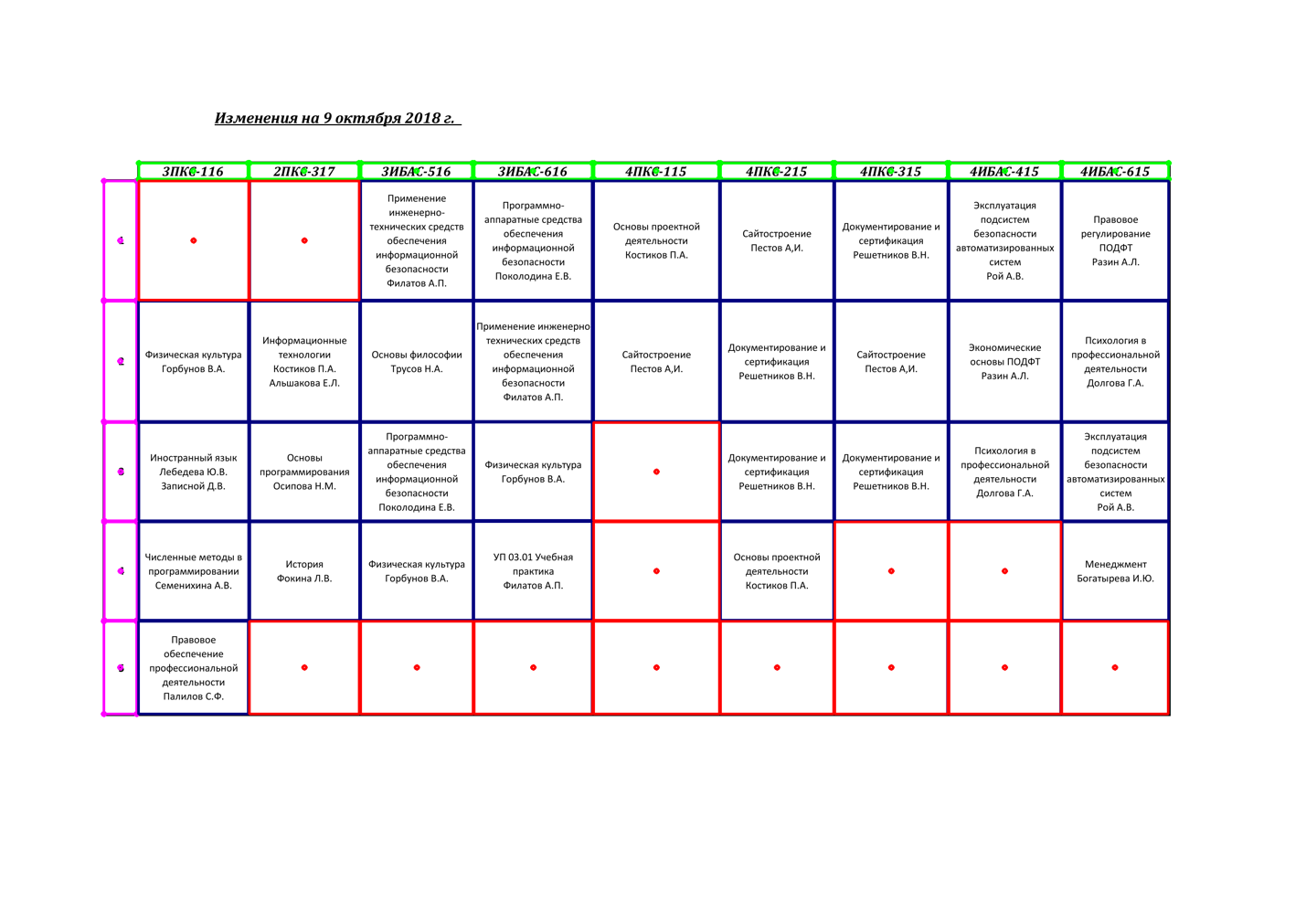
1. В цикле выполняются аналогичные условия пункта 10. Если условие сработало, то запускаем цикл по каждому элементу списка box\_store\_list, содержащего координаты углов всех отрисованных ячеек предметов. Если абсолютная погрешность между любыми доступными точками меньше 10 пикселей и всего таких точек 4 (каждый угол прямоугольника), то добавляем координаты центра вписанного прямоугольника в список outchecklist, отрисовываем вписанный прямоугольник и выделяем центр в виде круга с радиусом 5 пикселей (Рисунки 2.7 - 2.8). Выходим из цикла.

Рис. 2.7 – Визуализация вписанных прямоугольников

 Рис. 2.8 – Преобразованное изображение на выходе

1. Для каждой пары координат центра пустых пар в списке outchecklist запускаем цикл, в котором запускаем цикл по каждому значению координат в списке finalmatrix. Если абсолютная погрешность обоих осей меньше 10 пикселей, то считаем этот элемент пустой парой, иначе заносим распознанный текст пары с ассоциативного массива center\_and\_text. Работа с функцией get\_null\_values закончена.
2. Записываем результирующее изображение в директорию outputs с индексом GOOD. Вызываем функцию finalmatrix\_to\_json для преобразования списков groupcheck, finalmatrix и group\_text\_association в единый объект json.
3. Объект json записывается в базу данных MySQL с помощью функции MySQLWriter.

Перейдем к desktop-клиенту, реализованному на C#. Основные используемые модули:

* Telegram.Bot – .NET клиент для Telegram Bot API;
* SQLite – компактная встраиваемая СУБД;
* SQLitePCLRaw.core – портативная библиотека классов для низкоуровневого доступа к SQLite;
* Newtonsoft.Json – популярный JSON- фреймворк для .NET;
* MySQLConnector – асинхронный MySQL ADO.NET провайдер, поддерживающий MySQL Server, MariaDB, Percona Server, Amazon Aurora, Azure Database for MySQL и другие;
* HttpToSocks5Proxy – класс, который реализует интерфейс IWebProxy для HTTP(S)-прокси при соединении с сервером SOCKS5;
* HandyControl – Содержит некоторые наиболее часто используемые простые элементы управления WPF.

Алгоритм работы клиента С#:

1. Если у пользователя отсутствует активное подключение к интернету, то программа сообщит, что работа невозможна (Рис. 2.9).
2. Запуск Telegram-сервера, если в предыдущий сеанс программы он был активен (Рис. 2.10). Запрос авторизации в программе. Пока пользователь не авторизуется, он не перейдет в интерфейс управления. Если при предыдущем входе пользователь активировал checkbox “Запомнить меня”, то данные предыдущего входа подставятся автоматически (пароль в зашифрованном виде), в противном случае пользователю потребуется ввести свои учетные данные (Рис. 2.11 – 2.12).
3. После авторизации в программе пользователь попадает в основное меню (Рис. 2.13 – 2.14). В основном меню возможен выбор опций отправки сообщений (Рис. 2.15 – 2.24), переход к меню действий с пользователем/пользователями, переход в меню статистики SMS, управление инициализацией Telegram-сервера, кнопка выхода. Тут же расположена информация о текущей сессии пользователя и текущем балансе сервиса SMS-оповещений SMSAero.
4. При переходе в меню управления пользователями возникает окно выбора действия: изменение данных текущего пользователя, добавление и удаление пользователей (Рис 2.25).
5. При переходе на форму изменения данных текущего пользователя существует выбор между изменением электронной почты, паролем и контактным телефоном. (Рис 2.26).
6. При изменении e-mail требуется ввести старый и новый адреса электронной почты. (Рис 2.27) После ввода данных требуется подтверждение в виде кода, отправленного с e-mail. (Рис 2.28 – 2.32) Если код валидный, то происходит смена e-mail, иначе выводится соответствующее сообщение. (Рис 2.33 – 2.34)
7. При изменении пароля требуется ввести старый пароль пользователя с новым паролем и его корректным повторением (Рис 2.35 – 2.38)
8. При изменении телефона требуется ввести старый и новый номера телефонов, после чего подтвердить его кодом из SMS (Рис 2.39 – 2.46)
9. При добавлении нового пользователя требуется ввести: мастер-пароль для подтверждения прав добавления нового пользователя, имя пользователя для идентификации в системе, контактный e-mail с номером телефона, а также пароль для авторизации в приложении. После ввода данных требуется их подтверждение посредством ввода кодов, полученных по e-mail и номеру телефона (Рис 2.47 – 2.56).
10. При удалении пользователя требуется мастер-пароль для подтверждения прав удаления, а также логин удаляемого пользователя. В качестве логина может использоваться номер телефона или электронная почта пользователя. (Рис 2.57 – 2.61)
11. При переходе в меню статистики SMS-оповещений пользователю отображается телефон, статус сообщения, стоимость и текст исходного сообщения (Рис 2.62)
12. При нажатии на статус Telegram-сервера происходит его остановка или запуск в зависимости от его текущего состояния (Рис 2.63 – 2.65).

Алгоритм работы Telegram-сервера:

* При получении команды /sms с параметрами в виде номера телефона и исходного SMS-сообщения у пользователя запрашивается подтверждение перед отправкой сообщения (Рис 2.66). Если ответ утвердительный, то происходит отправка сообщения и информирование пользователя о текущем балансе SMS-сервиса (Рис 2.67). Если пользователь неверно передал параметры отправки сообщения, то будет выведен общий синтаксис команды (Рис 2.68);
* При получении команды /balance пользователю отправляется баланс аккаунта SMS-сервиса (Рис 2.69);
* При получении команды /history пользователю показываются последние пять отправленных сообщений сервиса (Рис 2.70);
* При получении команды /user с параметром id пользователю предоставляются возможность добавления или удаления пользователя для управления клиентом Telegram (Рис 2.71). Если параметры переданы неверно, то будет выведен общий синтаксис команды (Рис 2.72);
* Если введена какая-либо другая команда, то клиент предоставит пользователю список всех доступных команд (Рис 2.73).

Карта взаимосвязей ссылок элементов расположена на рисунке 2

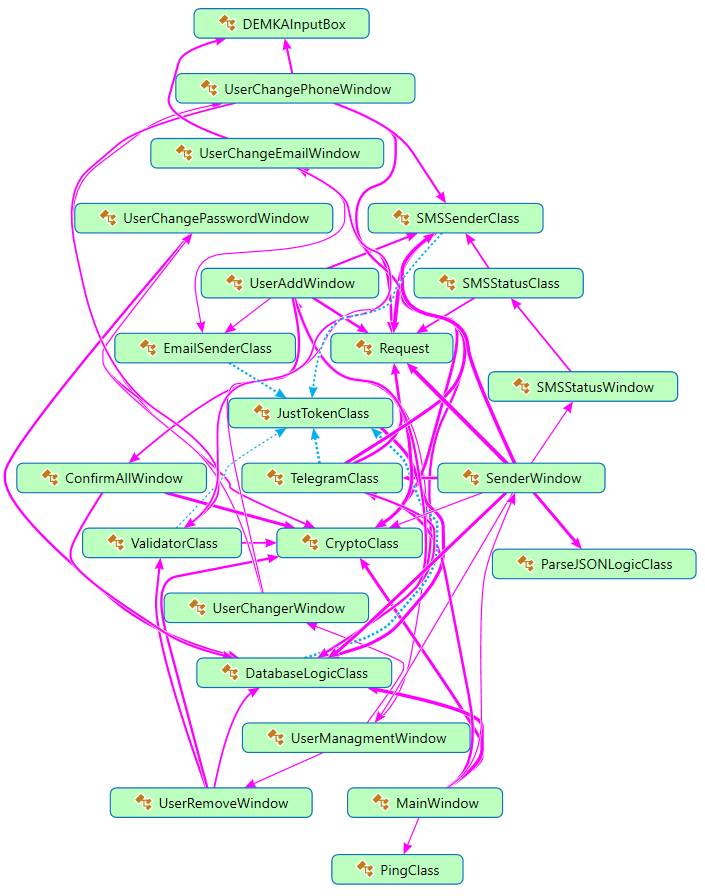
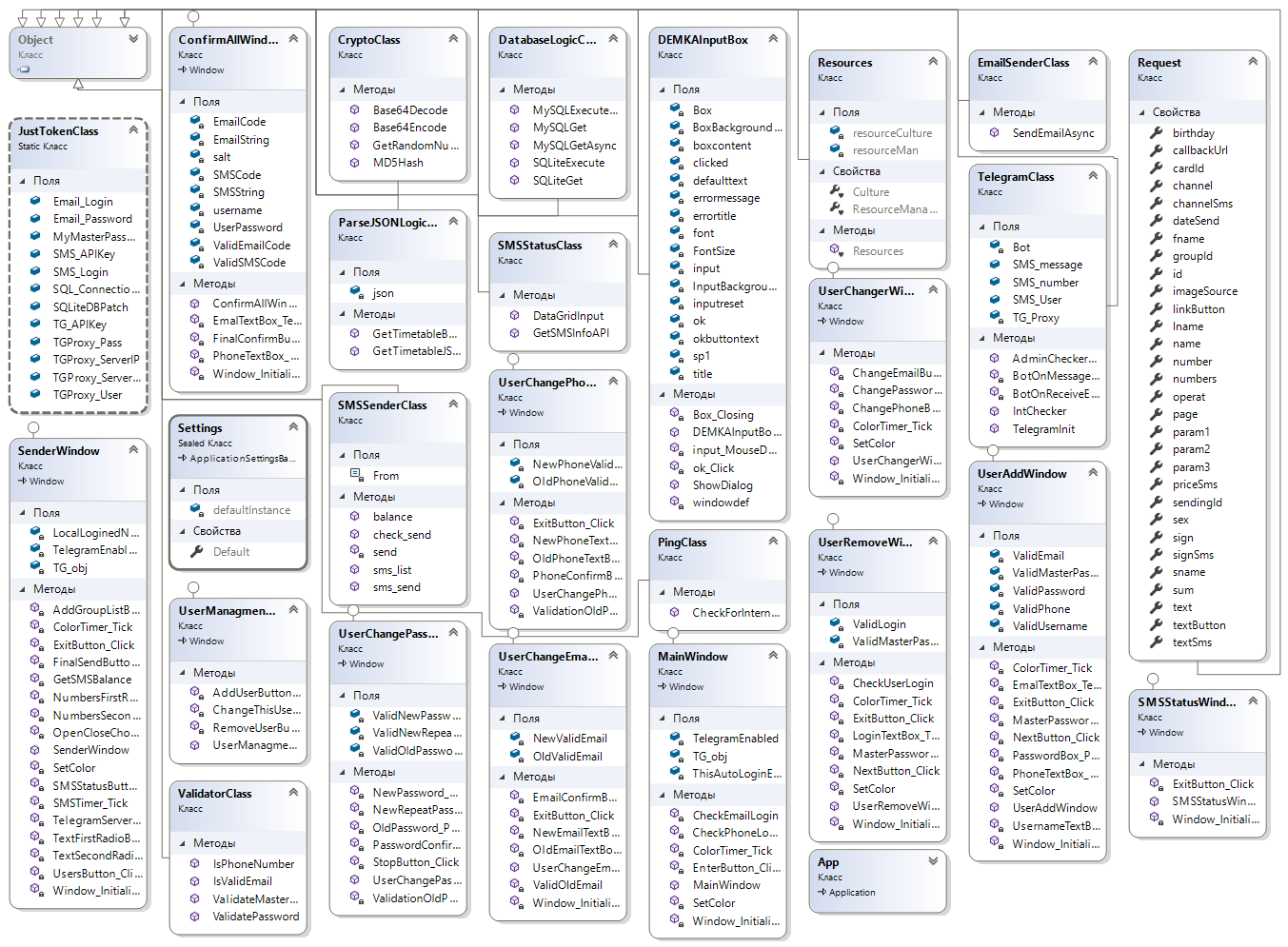


Рисунок 2 – Карта кода разрабатываемого решения

Диаграмма классов расположена на рисунке 3



## 2.2 Разработка пользовательских интерфейсов ПО

При разработке пользовательского интерфейса учитываются следующие правила:

* Интерфейс программы должен быть логичным и понятным Вашей целевой группе; любая операция, совершаемая пользователем, должна решаться минимальным числом действий;
* Уровень технической оснащенности интерфейса должен соответствовать уровню технической грамотности человека;
* Пользователь должен иметь возможность видеть результат своего действия.
* Разработка меню, панели инструментов, диалоговых окон и других элементов должна соответствовать целям и задачам проекта в целом.

Основной алгоритм взаимодействия пользователя описан в пункте 2.3. Интерфейс клиента показан на рисунках ниже.

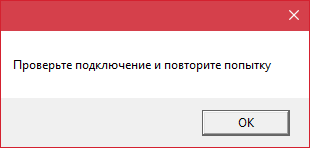


Рис. 2.9 – Сообщение об отсутствии активного интернет-соединения

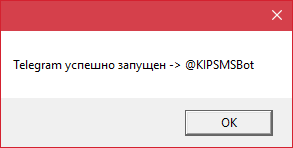


Рис. 2.10 – Сообщение об успешном запуске Telegram-сервера

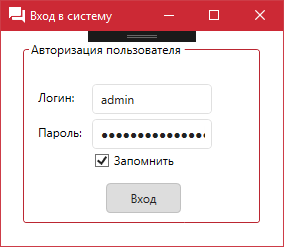


Рис. 2.11 – Автологин пользователя в системе

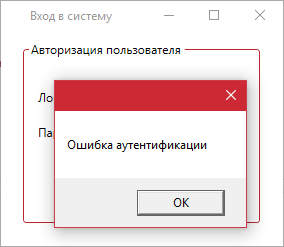


Рис. 2.12 – Сообщение о неправильной паре логина/пароля



Рис. 2.13 – Сообщение об успешной авторизации пользователя



Рис. 2.14 – Главное меню программы



Рис. 2.15 – Главное меню программы

 Рис. 2.16 – Выбор текста из БД с получателями из БД

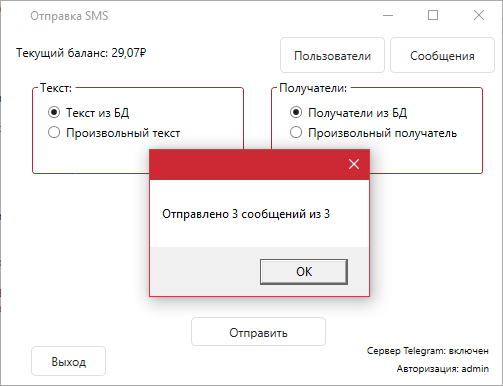


Рис. 2.17 – Статистика отправленных сообщений из БД

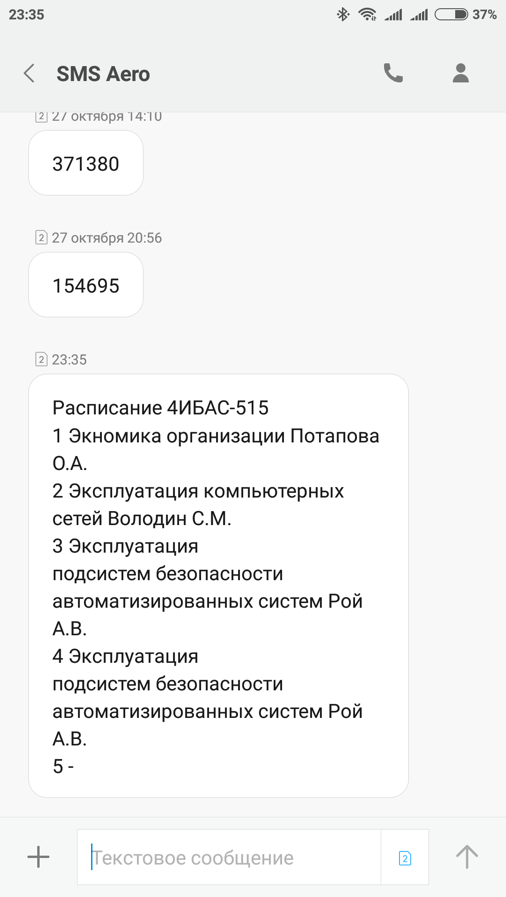


Рис. 2.18 – Полученный текст по SMS при выборе текста из БД с получателями из БД

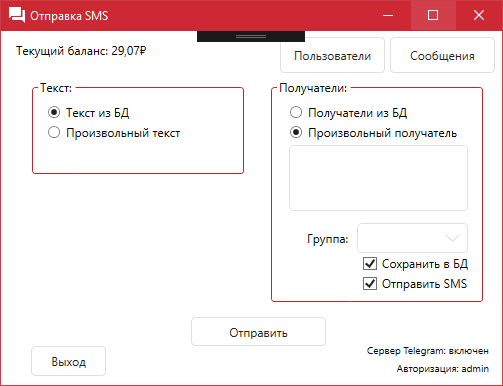


Рис. 2.19 – Выбор произвольного получателя с выбором текста из БД

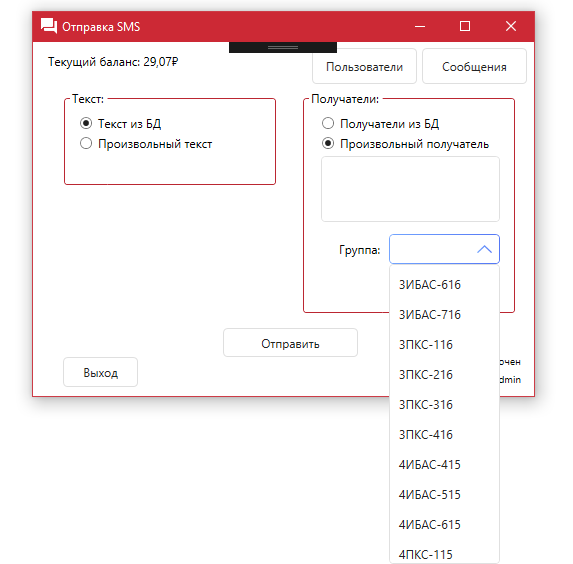


Рис. 2.20 – Выбор группы для записи ассоциации номера получателя и группы в БД

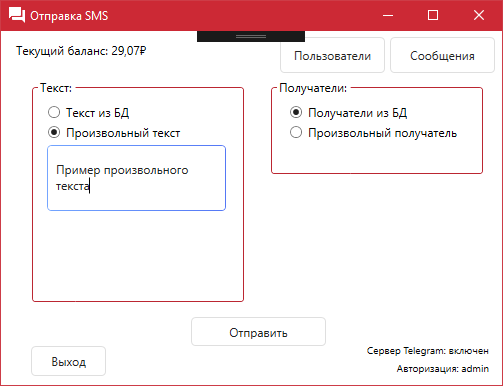


Рис. 2.21 – Пример выбора произвольного текста всем номерам-получателям из БД

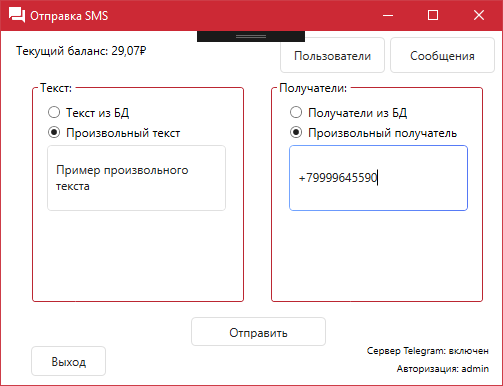


Рис. 2.22 – Пример отправки произвольного текста произвольному получателю

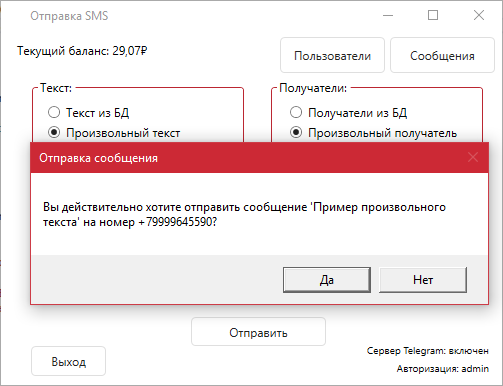


Рис. 2.23 – Пример выбора произвольного текста произвольному получателю

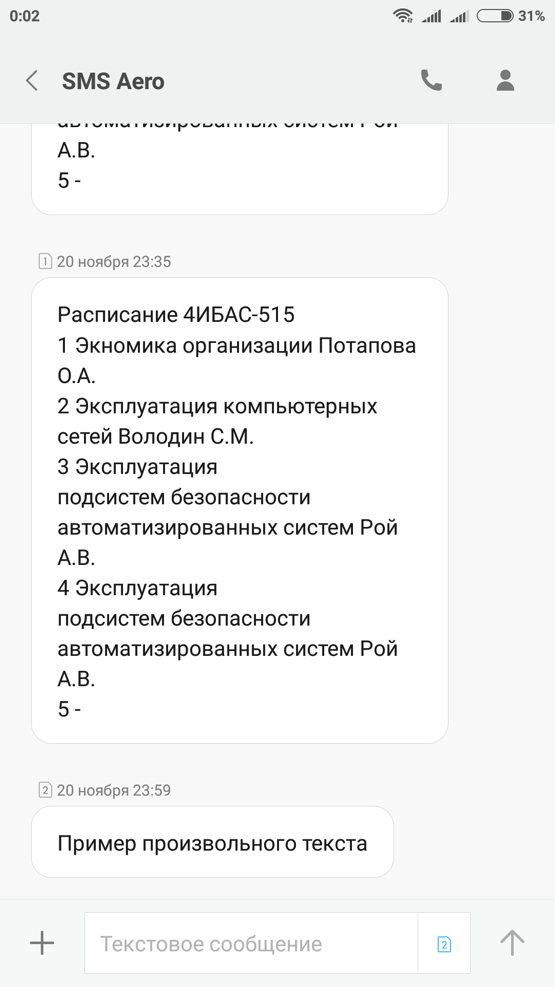


Рис. 2.24 – Полученный текст по SMS при выборе произвольного получателя и произвольного текста

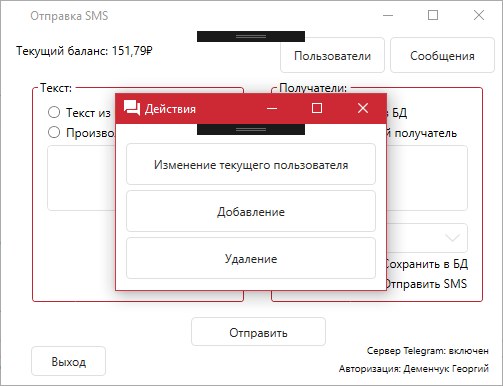


Рис. 2.25 – Форма меню управления пользователями

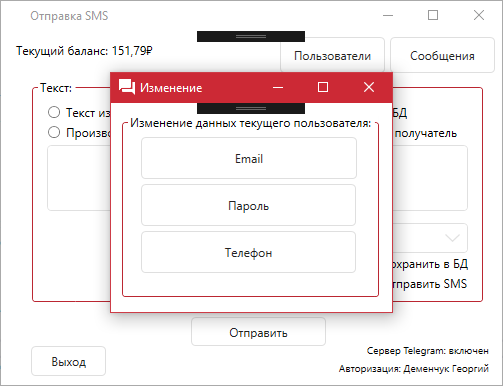


Рис. 2.26 – Форма изменения данных текущего пользователя



Рис. 2.27 – Форма изменения e-mail текущего пользователя

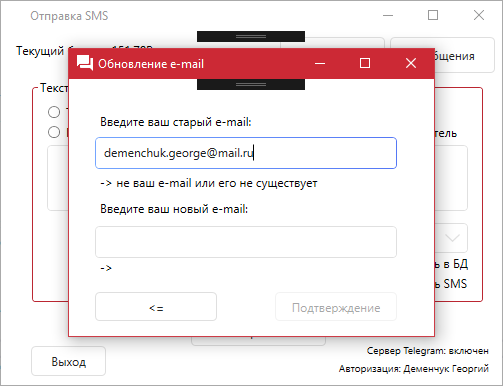


Рис. 2.28 – Пример разграничения доступа к изменению данных

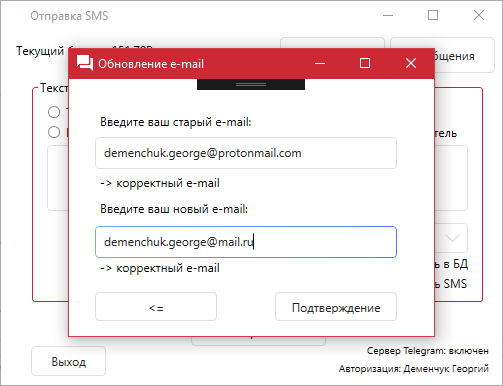


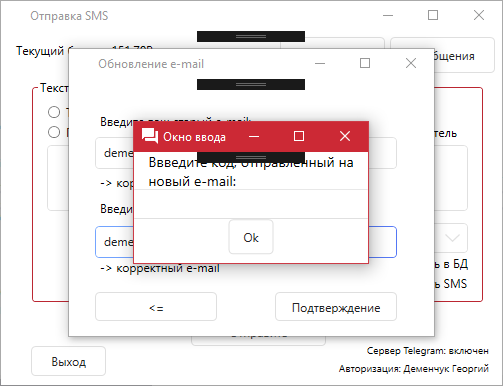
Рис. 2.29 – Пример корректного заполнения данных при изменении e-mail

Рис. 2.30 – Пример формы ввода кода подтверждения e-mail

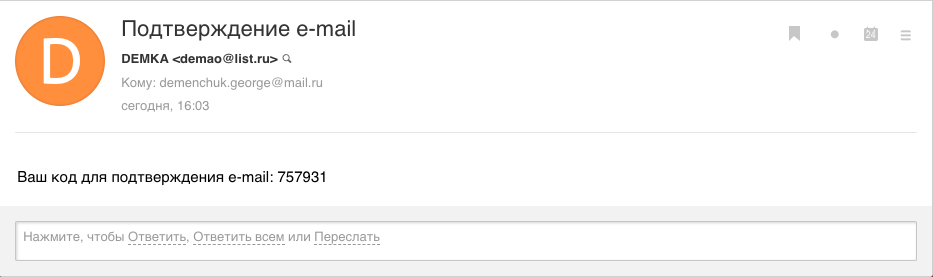
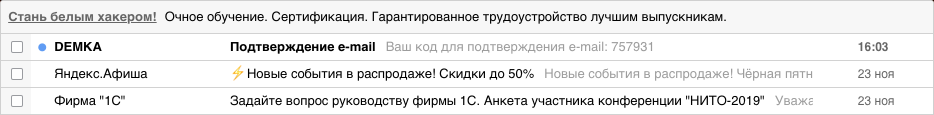


Рис. 2.31 – Пример полученного письма с кодом подтверждения

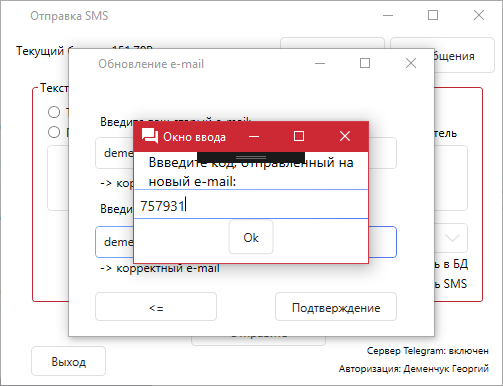


Рис. 2.32 – Пример ввода полученного кода подтверждения

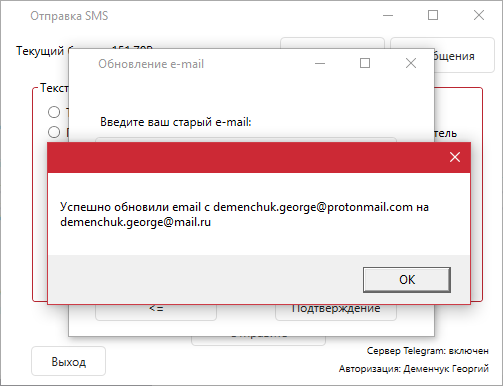


Рис. 2.33 – Сообщение об успешной смене e-mail

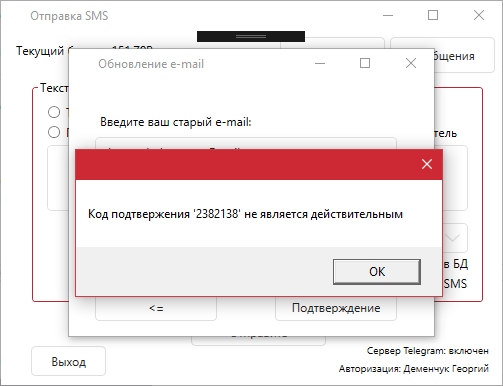


Рис. 2.34 – Сообщение о недействительном коде подтверждения e-mail

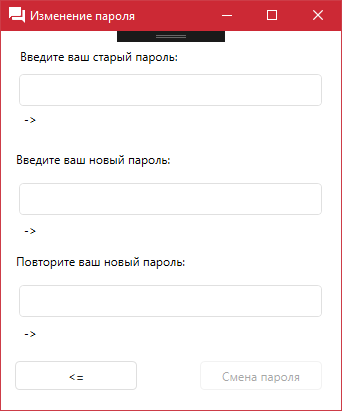


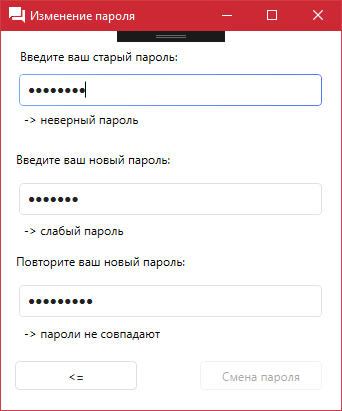
Рис. 2.35 – Форма изменения пароля текущего пользователя

Рис. 2.36 – Пример работы фильтрации при отрицательных результатах проверки

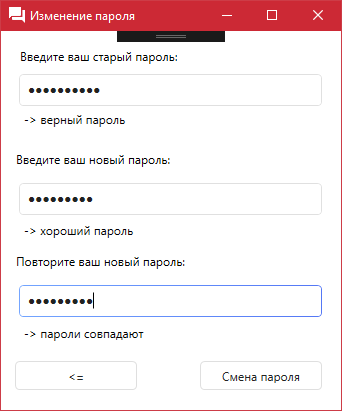


Рис. 2.37– Пример работы фильтрации при положительных результатах проверки

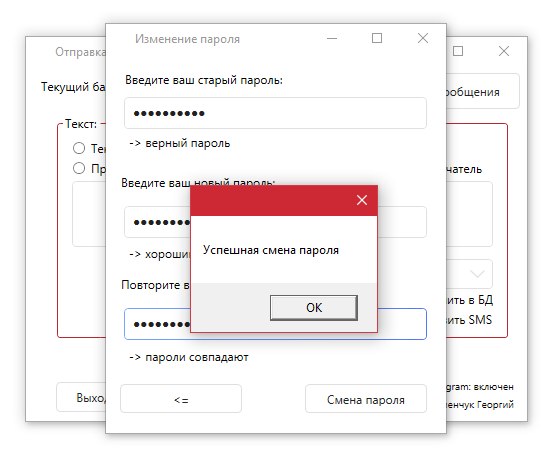


Рис. 2.38 – Сообщение об успешном изменении пароля

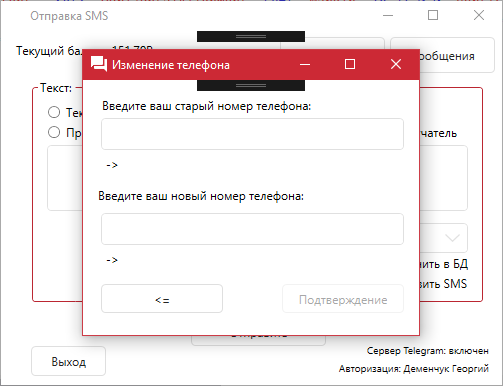


Рис. 2.39 – Форма изменения номера телефона текущего пользователя

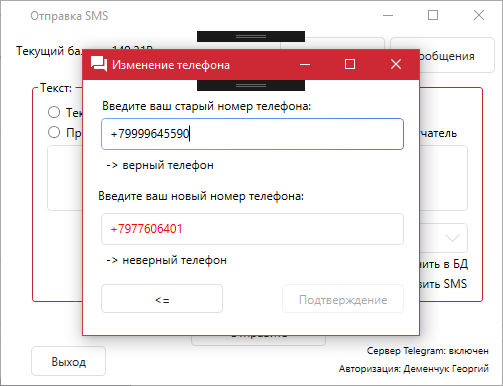


Рис. 2.40 – Пример работы фильтрации при отрицательных результатах проверки

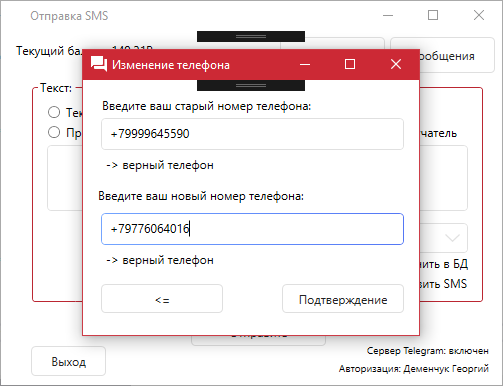


Рис. 2.41 – Пример работы фильтрации при положительных результатах проверки

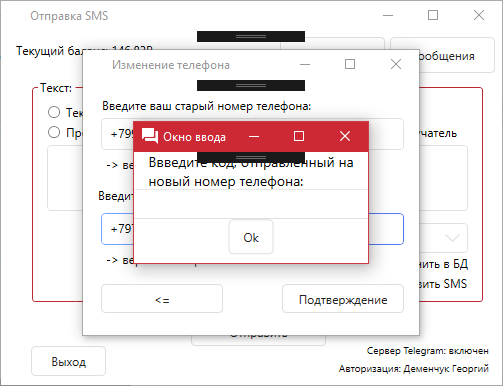


Рис. 2.42 – Пример формы ввода кода подтверждения мобильного телефона

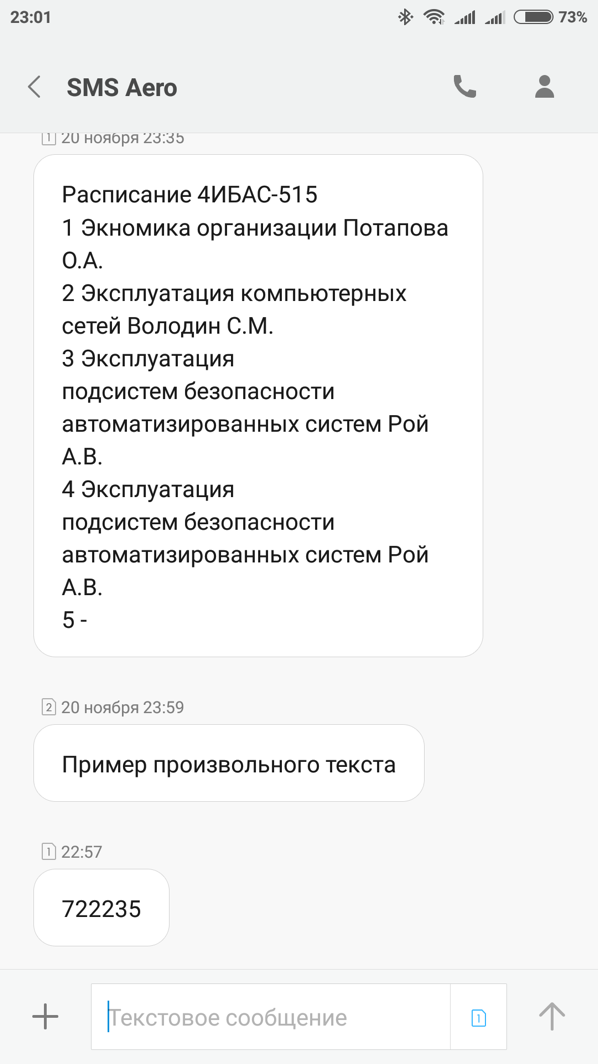


Рис. 2.43 – Пример полученного SMS с кодом подтверждения

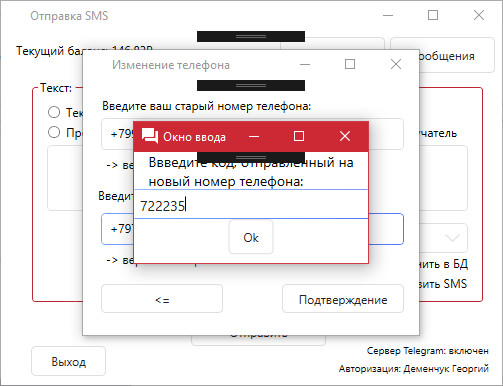


Рис. 2.44 – Пример ввода полученного кода подтверждения

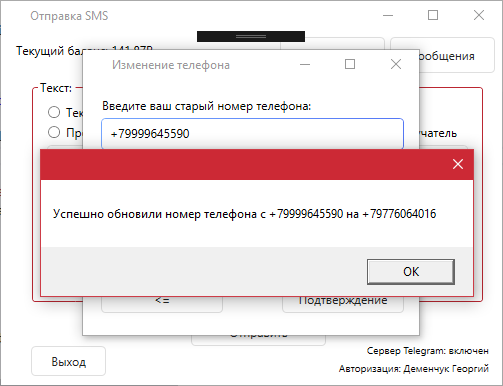


Рис. 2.45 – Сообщение об успешном изменении номера телефона

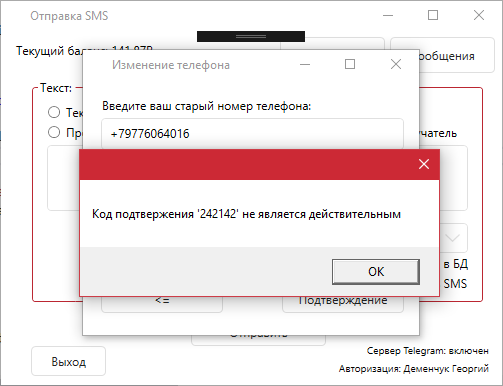


Рис. 2.46 – Сообщение о недействительном коде подтверждения

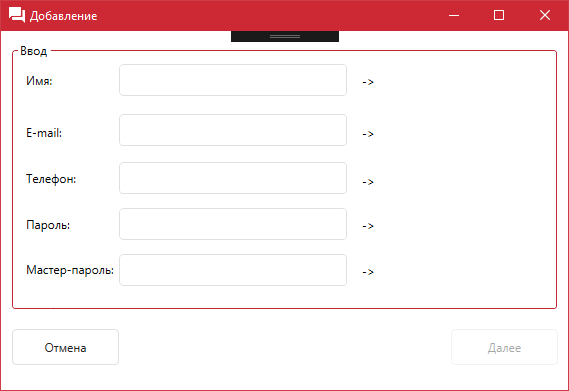


Рис. 2.47 – Форма добавления нового пользователя системы

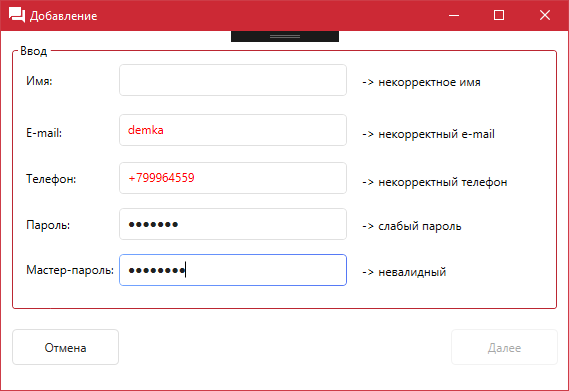


Рис. 2.48 – Пример работы фильтрации при отрицательных результатах проверки

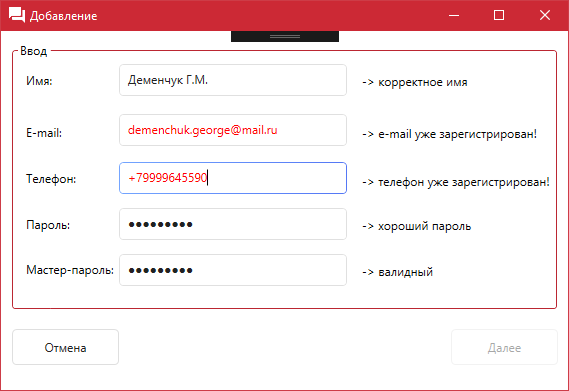


Рис. 2.49 – Реализация проверки на уже зарегистрированного пользователя системы

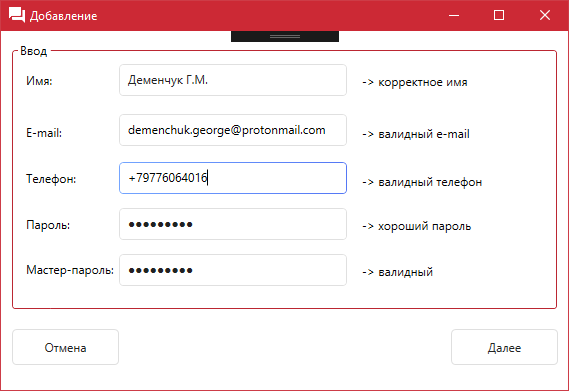


Рис. 2.50 – Пример работы фильтрации при положительных результатах проверки

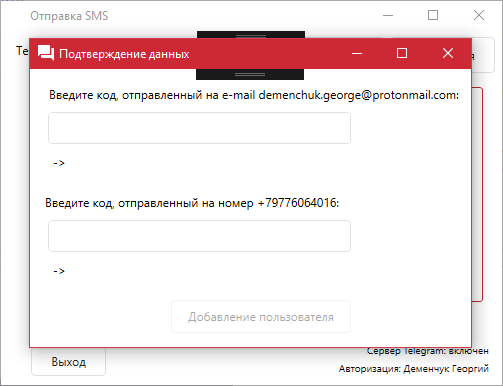
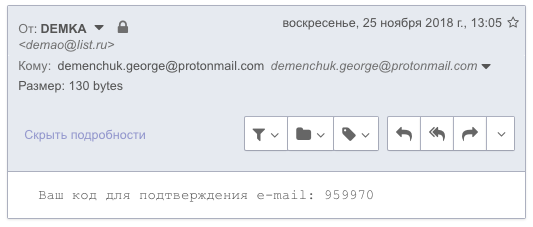


Рис. 2.51 – Форма подтверждения данных пользователя

 Рис. 2.52 – Пример полученного e-mail с кодом подтверждения

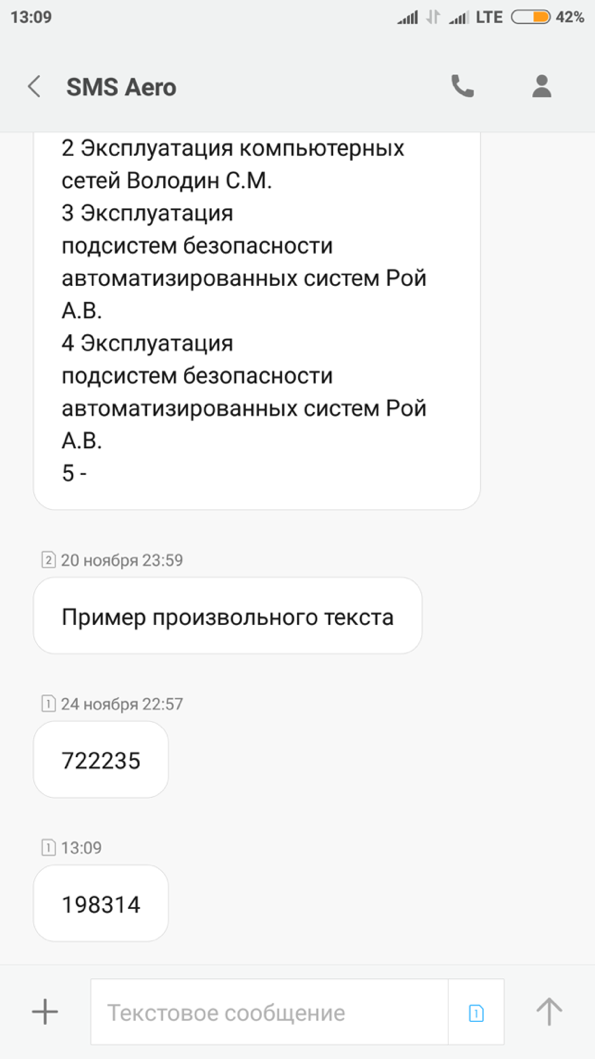
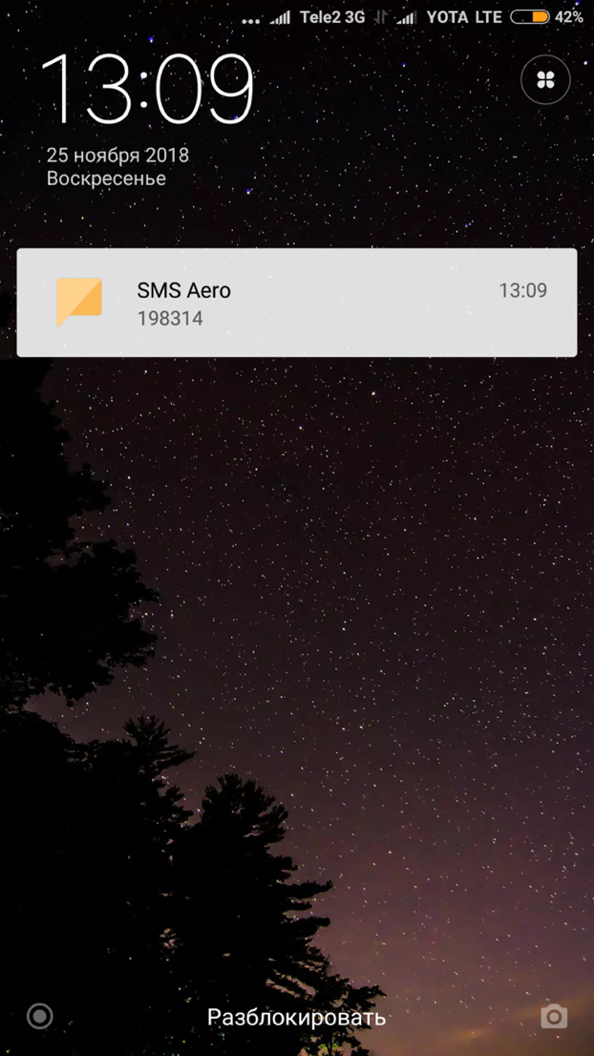


Рис. 2.53 – Пример полученного SMS с кодом подтверждения

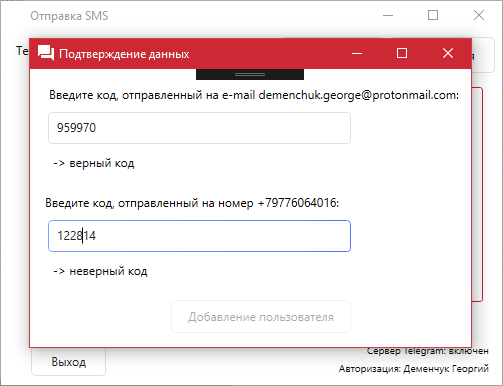


Рис. 2.54 – Пример ввода неверного кода подтверждения

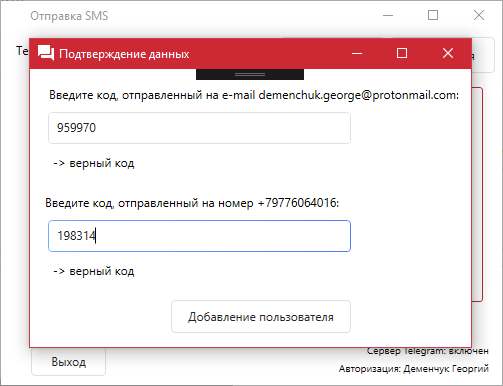


Рис. 2.55 – Пример ввода верного кода подтверждения

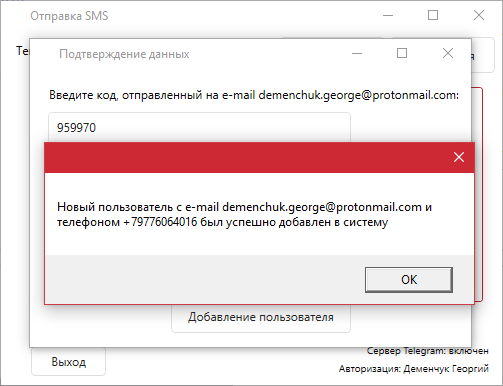


Рис. 2.56 – Сообщение об успешном добавлении нового пользователя в систему

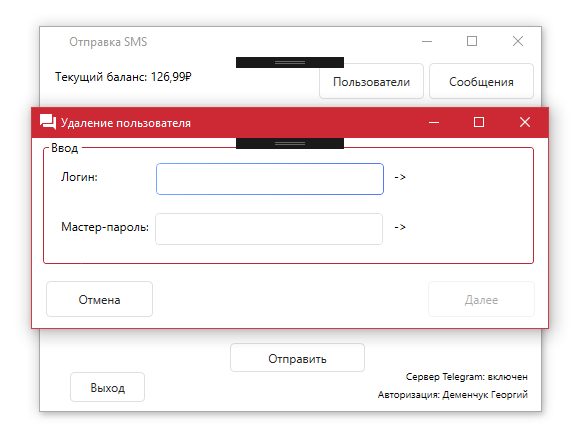


Рис. 2.57 – Форма удаления пользователей системы

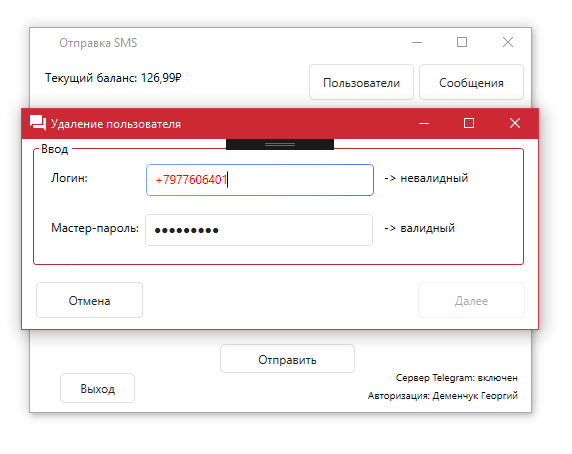


Рис. 2.58 – Пример ввода некорректного ввода номера телефона

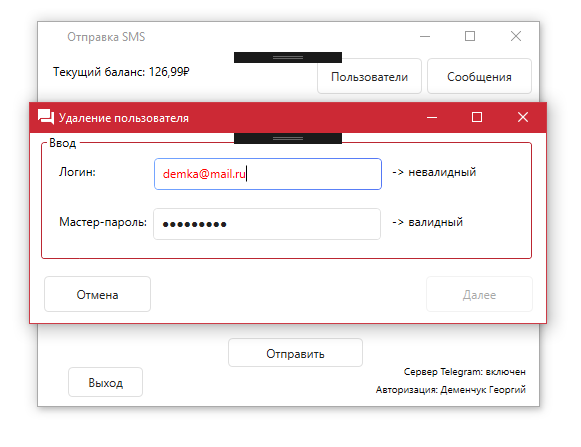


Рис. 2.59 – Пример ввода незарегистрированного e-mail в системе

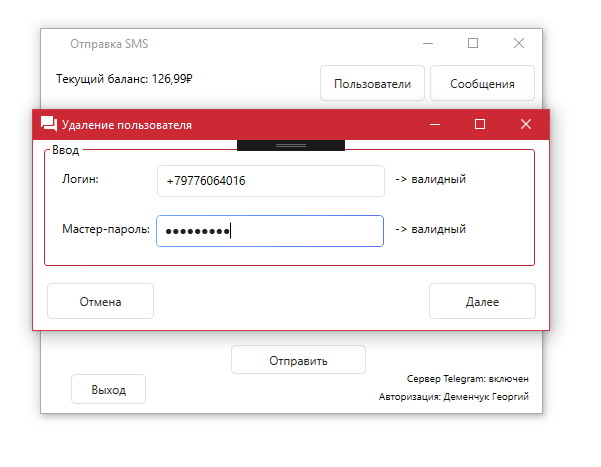


Рис. 2.60 – Пример корректного ввода логина пользователя

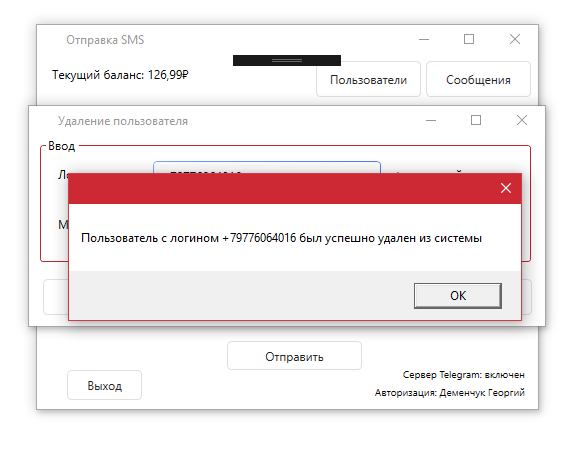


Рис. 2.61 – Сообщение об успешном удалении пользователя

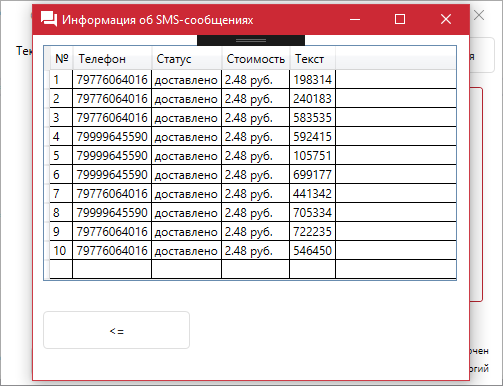


Рис. 2.62 – Форма статистики SMS-оповещений

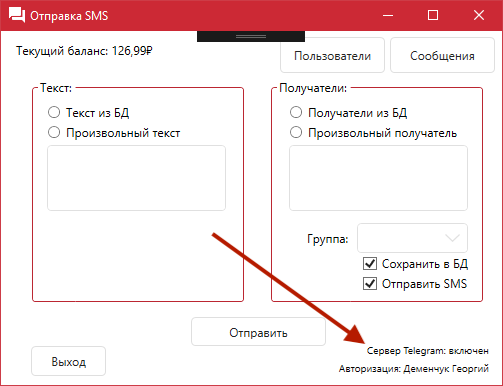


Рис. 2.63 – Информация о состоянии сервера Telegram

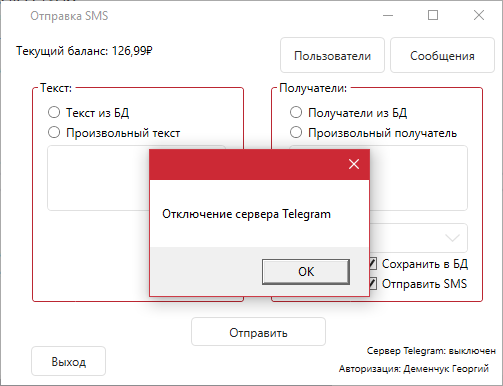


Рис. 2.64 – Сообщение об отключении сервера Telegram

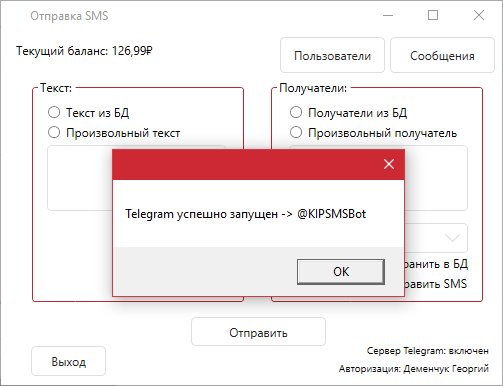


Рис. 2.65 – Сообщение о повторном запуске сервера Telegram

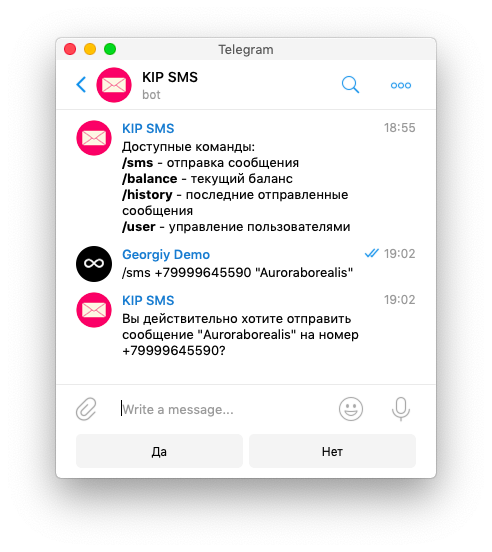


Рис. 2.66 – Запрос подтверждения отправки сообщения у пользователя

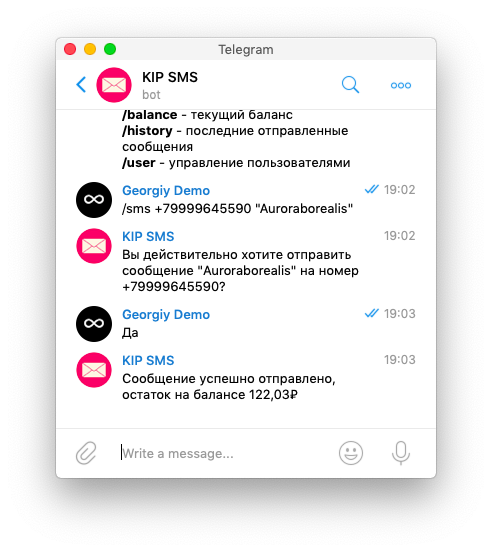


Рис. 2.67 – Отправка сообщения и информирование пользователя о текущем балансе

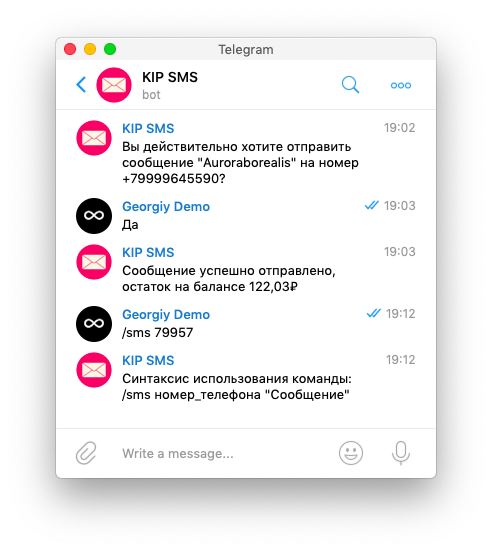


Рис. 2.68 – Сообщение о неверной передаче параметров команды



Рис. 2.69 – Пример запроса текущего баланса аккаунта



Рис. 2.69 – Пример запроса текущего баланса пользователя

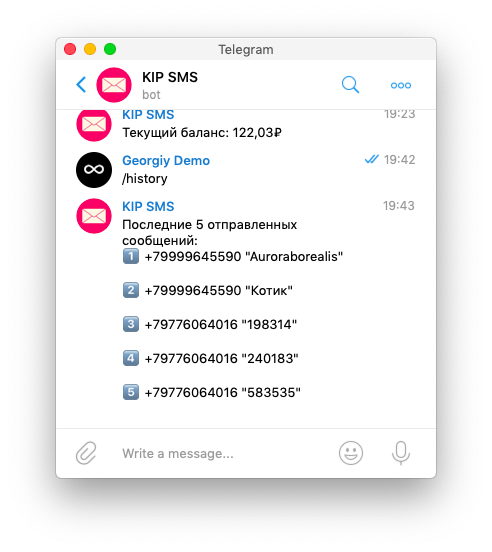


Рис. 2.70 – Запрос пяти последних отправленных сообщений

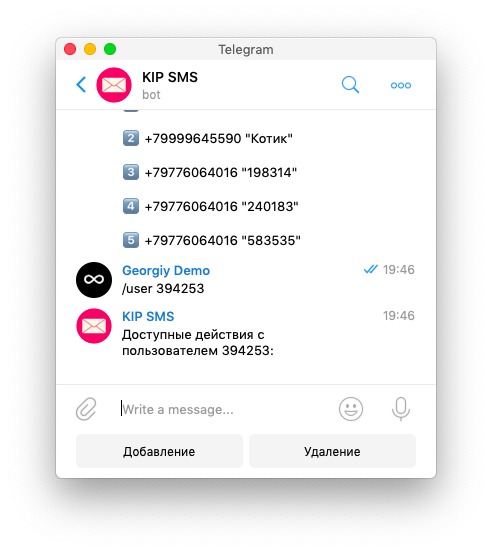


Рис. 2.71 – Пример доступных действий с пользователем

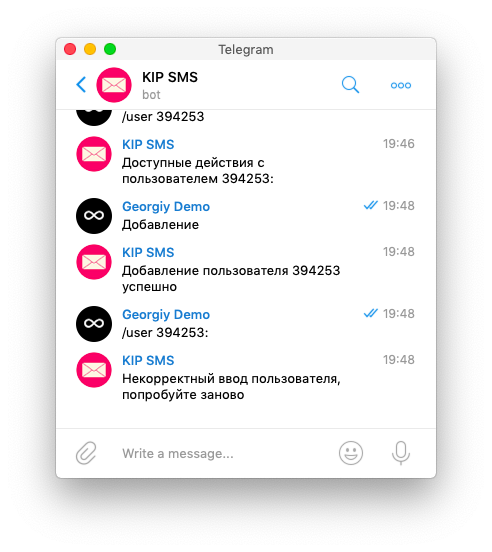


Рис. 2.72 – Сообщение о неверной передаче параметров команды

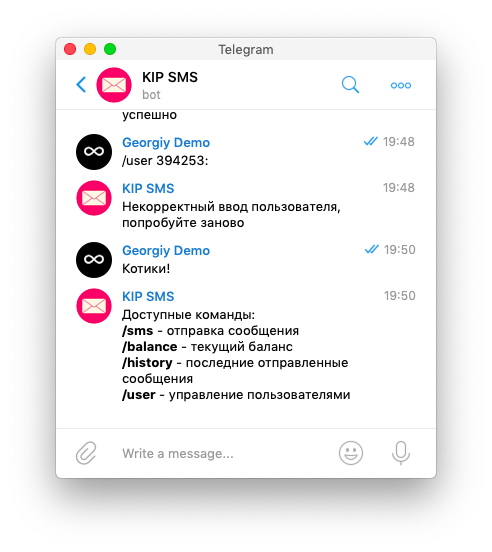


Рис. 2.73 – Пример ввода несуществующей команды

# 3 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

## 3.1 Технологии, применяемые при разработке ПО

## 3.2 Тестирование и отладка программного обеспечения

Тестовые данные представлены в таблице 3.1.

Таблица 3.1 – Тестовые данные

|  |  |  |
| --- | --- | --- |
| Номер теста | Тестовый набор данных | Реакция программы |
| 1 | Табельный номер:  Имя:  Фамилия:  Отчество: | Сообщение об ошибке, блокировка кнопок до тех пор, пока пустые поля не будут заполнены |
| 2 | Табельный номер: вавыпп  Имя: 1323  Фамилия: 123214  Отчество: 34324 | Сообщение об ошибке, блокировка кнопок до тех пор, пока не будут введены корректные данные |
| 3 | Табельный номер: 998700  Имя: Курганов  Фамилия: Альберт  Отчество: Викторович | Продолжение работы программы, добавление записи в БД и отображение в ListView (если есть на форме) |
| 4 | Инвентарный номер: 100100100  Серийный номер: D545645212146  Наименование оборудования: Процессор 2,66 Ghz Intel Core 2 Duo  Код кабинета:  Дата установки: | Отображение соответствующее этому значению записи по запросу из БД в поле формы или сохранение записи в локальной переменной (некоторые формы) |

# 4 РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ

## 4.1 Руководство программиста

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): 4,00 ГБ;
* Тип системы: 64-разрядная операционная система, процессор х64;
* Клавиатура;
* USB-мышь либо наличие сенсорной панели (touchpad).

В решении присутствуют 22 дополнительных объекта, среди них 11 классов .cs и 11 форм WPF (.xaml).

К формам относятся:

* MainWindow.xaml – форма авторизации пользователя, при корректном вводе пары логина/пароля вызывает SenderWindow.xaml;
* ConfirmAllWindow.xaml – форма подтверждения кодов, отправленных на номер телефона и e-mail при добавлении нового пользователя, вызывается формой UserAddWindow.xaml;
* SenderWindow.xaml – форма главного меню программы с отправкой сообщений, возможен вызов форм UserManagmentWindow.xaml и SMSStatusWindow.xaml;
* SMSStatusWindow.xaml – форма отображения статистики отправленных сообщений, включающей в себя номер телефона, статус сообщения, стоимость и его исходный текст;
* UserAddWindow.xaml – форма добавления нового пользователя в систему, вызывается UserManagmentWindow.xaml. При прохождении фильтрации форма вызывает ConfirmAllWindow.xaml для процесса подтверждения данных;
* UserChangeEmailWindow.xaml – форма изменения контактного e-mail текущего пользователя, вызывается UserChangerWindow.xaml;
* UserChangePasswordWindow.xaml – форма изменения пароля текущего пользователя, вызывается UserChangerWindow.xaml;
* UserChangePhoneWindow.xaml - форма изменения контактного телефона текущего пользователя, вызывается UserChangerWindow.xaml;
* UserChangerWindow.xaml – форма управления изменениями данных текущего пользователя, вызывается через UserManagmentWindow.xaml, вызывает UserChangePhoneWindow.xaml, UserChangePasswordWindow.xaml, UserChangeEmailWindow.xaml;
* UserManagmentWindow.xaml – форма менеджмента пользователей, включает в себя формы UserRemoveWindow.xaml, UserAddWindow.xaml, а также UserChangerWindow.xaml. Вызывается формой UserChangerWindow.xaml;
* UserRemoveWindow.xaml – форма удаления пользователя в систему, вызывается UserManagmentWindow.xaml.

Иерархию форм и структуру их вызова можно посмотреть на рисунке 3.1.

кпкупуккппк



К классам относятся:

* CryptoClass.cs – класс для работы с функциями шифровки/дешифровки данных, содержит методы кодинга и декодинга в Base64, преобразования строки в MD5-хеш и получения псевдорандомных чисел;
* DatabaseLogicClass.cs – класс, содержащий логику работы с базами данных MySQL и SQLite3. Для MySQL существуют как синхорнные, так и асинхронные методы для внесения и получения данных;
* DEMKAInputBox.cs – класс для реализации InputBox для C# WPF, активно используется формами подтверждения кодов;
* EmailSenderClass.cs – класс для отправки электронной почты на определенный адрес, использует System.Net.Mail;
* JustTokenClass.cs – класс, содержащий настройки для подключения к базе данных MySQL, путь к SQLite, токены для API, логины и пароли. Вся информация берется из БД на удаленном сервере, кроме мастер-пароля (он прописан локально в SQLite и для каждой копии программы должен выставляться отдельно);
* ParseJSONLogicClass.cs – класс для запроса от backend-сервера данных об изменении в расписании в формате JSON с последующей его обработкой;
* PingClass.cs – класс для проверки активного интернет-соединения;
* SMSSenderClass.cs – класс с логикой взаимодействия с API сервиса рассылок SMSAero. Чаще всего используется формой главного меню - SenderWindow.xaml;
* SMSStatusClass.cs – cодержит методы оформления DataGrid для формы SMSStatusWindow.xaml;
* TelegramClass.cs – класс для развертки локального сервера Telegram, его инициализацию и остановку, парсинг JSON и активное взаимодействие с большинством классов;
* ValidatorClass.cs – класс для фильтрации вводимых значений, содержит методы для проверки e-mail, телефона, пароля и мастер-пароля. Используется всеми формами, где происходит ввод данных, корме MainWindow.xaml.

Рассмотрим методы и поля каждой формы более подробно.

Форма MainWindow.xaml

Содержит следующие поля:

* TelegramEnabled типа Boolean – флаг запущенного сервера Telegram;
* ThisAutoLoginEnabled типа Boolean – флаг активации автологина на форме;
* Объект TG\_obj класса TelegramClass для дальнейшей его инициализации.

Содержит следующие методы:

* Конструктор MainWindow для инициализации окна;
* CheckEmailLogin – проверка совпадения пары e-mail/пароль с записями в базе данных MySQL;
* CheckPhoneLogin - проверка совпадения пары телефон/пароль с записями в базе данных MySQL;
* Событийная процедура EnterButton\_Click – обрабатывает нажатие на кнопку для авторизации пользователя;
* Событийная процедура Window\_Initialized, вызываемая после инициализации окна;
* Процедура ColorTimer\_Tic – предназначена для выполнения таймером каждый n промежуток времени, вызывающая SetColor;
* Процедура SetColor – окрашивает некоторые элементы формы (в частности groupbox) цвет системной темы;

Форма SenderWindow.xaml

Содержит следующие поля:

* LocalLoginedName типа string – используется для отображения имени пользователя, который произвел авторизацию;
* TelegramEnabled типа Boolean – флаг запущенного сервера Telegram;
* Объект TG\_obj класса TelegramClass для работы с классом Telegram.

Содержит следующие методы:

* Конструктор SenderWindow для инициализации окна, а также получения значений, переданных с помощью конструктора;
* FinalSendButton\_Click – асинхронная событийная процедура нажатия на кнопку
* AddGroupListBoxAsync – асинхронная процедура добавления названий групп в элемент GroupsComboBox;
* Window\_Initialized – событийная процедура, вызываемая после инициализации окна;
* SMSTimer\_Tick – процедура таймера, каждый n промежуток времени вызывает GetSMSBalance;
* ColorTimer\_Tick – процедура таймера, каждый n промежуток времени вызывает SetColor;
* SetColor – окрашивает некоторые элементы формы в цвет системной темы;
* GetSMSBalance – процедура для отображения текущего баланса пользователя сервиса SMSAero;
* SMSStatusButton\_Click – событийная процедура нажатия на кнопку «Сообщения», вызывает форму SMSStatusWindow;
* TelegramServerLabel\_MouseLeftButtonUp – событийная процедура нажатия на текст со статусом Telegram-сервера. Останавливает или запускает сервер мессенджера в зависимости от предыдущего его статуса;
* OpenCloseChooseOptions – метод для адаптации интерфейса главной формы относительно выбранных параметров отправки сообщений.
* UsersButton\_Click – событийная процедура нажатия на кнопку «Пользователи», вызывает форму UserManagmentWindow;
* ExitButton\_Click – событийная процедура кнопки выхода из приложения, обнуляет данные авторизованного пользователя в SQLite
* NumbersSecondRadioButton\_Checked, TextFirstRadioButton\_Checked, TextSecondRadioButton\_Checked и NumbersFirstRadioButton\_Checked – событийные процедуры, вызывающие OpenCloseChooseOptions;

Форма СonfirmAllWindow.xaml

Содержит следующие поля:

* EmailString типа string – используется для хранения строки email имени пользователя, который произвел авторизацию, передается через конструктор;
* SMSString типа string – используется для хранения номера телефона, передается через конструктор;
* EmailCode типа string – хранение кода подтверждения, отправленного по e-mail, передается через конструктор;
* SMSCode типа string – хранение кода подтверждения, отправленного по SMS, передается через конструктор;
* UserPassword типа string – пароль добавленного пользователя, записывается в БД при правильном вводе кодов подтверждения SMSCode и EmailCode;
* Salt типа string – набор цифр, используется для записи уникальных кодов подтверждения в SQLite;
* Username типа string – имя нового пользователя для записи в SQLite;
* ValidSMSCode типа Boolean – флаг валидности кода подтверждения, отправленный по номеру телефона;
* ValidEmailCode типа Boolean – флаг валидности кода подтверждения, отправленный по электронной почте.

Содержит следующие методы:

* ConfirmAllWindow – конструктор, инициализирует поля EmailString, SMSString, UserPassword и username;
* FinalConfirmButton\_Click – асинхронная событийная процедура нажатия на кнопку “Добавление пользователя”, записывает данные нового пользователя в БД;
* EmalTextBox\_TextChanged – событийная процедура изменения текста, осуществляет валидацию кода подтверждения, отправленного на e-mail;
* PhoneTextBox\_TextChanged – событийная процедура изменения текста, осуществляет валидацию кода подтверждения, отправленного на номер телефона;
* Window\_Initialized – событийная процедура, срабатываемая после инициализации окна. Получает коды подтверждения из SQLite.

Форма SMSStatusWindow.xaml

Содержит следующие методы:

* SMSStatusWindow – конструктор, инициализирует форму;
* ExitButton\_Click – событийная процедура для закрытия текущей формы;
* Window\_Initialized – событийная процедура, происходит определение источника данных для DataGrid на форме и помощью класса SMSStatusClass;

Форма UserAddWindow.xaml

Содержит следующие поля:

* ValidUsername типа Boolean – флаг валидации имени нового пользователя;
* ValidEmail типа Boolean – флаг валидации электронной почты нового пользователя;
* ValidPhone типа Boolean – флаг валидации номера телефона нового пользователя;
* ValidPassword типа Boolean – флаг валидации пароля нового пользователя;
* ValidMasterPassword типа Boolean – флаг валидации мастер-пароля программы;

Содержит следующие методы:

* UserAddWindow – конструктор, инициализирует текущую форму;
* NextButton\_Click – событийная процедура нажатия на кнопку “Далее”. Генерурует уникальное число salt, отправляет коды подтверждения по e-mail и номеру телефона, записывает их в MySQL, открывает форму ConfirmAllWindow с подтверждением данных;
* PhoneTextBox\_TextChanged – событийная процедура фильтрации вводимого пользователем номера телефона;
* EmalTextBox\_TextChanged – событийная процедура фильтрации вводимого пользователем e-mail;
* Window\_Initialized – инициализация формы и таймера ColorTimer;
* ColorTimer\_Tick – событийная процедура таймера, каждый n промежуток времени вызывает SetColor;
* SetColor – окрашивает некоторые элементы формы в цвет системной темы;
* UsernameTextBox\_TextChanged – событийная процедура фильтрации вводимого пользователем имени;
* PasswordBox\_PasswordChanged – событийная процедура фильтрации вводимого пользователем пароля;
* MasterPasswordBox\_PasswordChanged – событийная процедура проверки вводимого пользователем мастер-пароля;
* ExitButton\_Click – событийная процедура для закрытия текущей формы;

Форма UserChangeEmailWindow.xaml

Содержит следующие поля:

* OldValidEmail типа Boolean – флаг валидации старого e-mail пользователя;
* NewValidEmail типа Boolean – флаг валидации новой электронной почты пользователя.

Содержит следующие методы:

* UserChangeEmailWindow – конструктор, инициализирует текущую форму;
* ValidOldEmail – процедура для проверки старого e-mail. Производит сопоставление введённого e-mail с данными текущего авторизованного пользователя, хранимыми в SQLite;
* EmailConfirmButton\_Click – событийная процедура нажатия на кнопку подтверждения изменения email, отправляет сгенерированный код подтверждения e-mail. Если код подтверждения верный, то обновляет данные в MySQL;
* NewEmailTextBox\_TextChanged – событийная процедура для фильтрации введенного обновленного e-mail;
* OldEmailTextBox\_TextChangedAsync – событийная процедура изменения текста старого e-mail. Взывает метод асинхронный метод ValidOldEmail;
* ExitButton\_Click – событийная процедура для закрытия текущей формы.

Форма UserChangePasswordWindow.xaml

Содержит следующие поля:

* ValidOldPassword типа Boolean – флаг валидации старого пароля пользователя;
* ValidNewPassword типа Boolean – флаг валидации нового пароля пользователя;
* ValidNewRepeatPassword типа Boolean – флаг валидации повторения нового пароля пользователя;

Содержит следующие методы:

* UserChangePasswordWindow – конструктор, инициализирует текущую форму;
* ValidationOldPassword – функция сравнения старого пароля пользователя из БД и введенного. Берет логин текущего залогиненного пользователя из SQLite и асинхронно получает пароль в MD5 из MySQL, далее сравнивает их;
* PasswordConfirmButton\_Click – событийная процедура нажатия на кнопку подтверждения изменения пароля. Получает из SQLite данные текущего залогиненного пользователя и асинхронно выполняет обновление данных в MySQL, информирует об успешном обновлении пароли и закрывает текущую форму;
* StopButton\_Click – событийная процедура для закрытия текущей формы;
* OldPassword\_PasswordChanged – событийная процедура изменения старого пароля. Взывает асинхронный метод ValidationOldPassword;
* NewPassword\_PasswordChanged – событийная процедура изменения нового пароля. Осуществляет фильтрацию нового пароля.
* NewRepeatPassword\_PasswordChanged – событийная процедура изменения повторения нового пароля. Осуществляет проверку на дубликат пароля.

Форма UserChangerWindow.xaml

Содержит следующие методы:

* UserChangerWindow – конструктор, инициализирует текущую форму;
* ChangeEmailButton\_Click – событийная функция кнопки для изменения e-mail текущего пользователя. Открывает форму UserChangeEmailWindow;
* ChangePasswordButton\_Click – событийная функция кнопки для изменения пароля текущего пользователя. Открывает форму UserChangePasswordWindow;
* ChangePhoneButton\_Click – событийная функция кнопки для изменения номера телефона текущего пользователя. Открывает форму UserChangePhoneWindow;
* Window\_Initialized – событийная функция для инициализации таймера;
* ColorTimer\_Tick – событийная процедура таймера, каждый n промежуток времени вызывает SetColor;
* SetColor – окрашивает некоторые элементы формы в цвет системной темы;

Форма UserManagmentWindow.xaml

Содержит следующие методы:

* UserManagmentWindow. – конструктор, инициализирует текущую форму;
* ChangeThisUserButton\_Click – событийная функция кнопки для изменения данных текущего пользователя. Открывает форму UserChangerWindow;
* AddUserButton\_Click – событийная функция кнопки для добавления новых пользователей систему. Открывает форму UserAddWindow;
* RemoveUserButton\_Click – событийная функция кнопки для удаления пользователей системы. Открывает форму UserRemoveWindow;

Форма UserRemoveWindow.xaml

Содержит следующие поля:

* ValidLogin типа Boolean – флаг валидации логина удаляемого пользователя;
* ValidMasterPassword типа Boolean – флаг валидации мастер-пароля программы для подтверждения прав.

Содержит следующие методы:

* UserRemoveWindow – конструктор, инициализирует текущую форму;
* CheckUserLogin – асинхронная функция для проверки логина на существование в MySQL БД;
* LoginTextBox\_TextChanged – событийная процедура, осуществляет фильтрацию вводимого логина и его принадлежность к номеру телефона или e-mail.
* MasterPasswordBox\_PasswordChanged – событийная процедура проверки вводимого пользователем мастер-пароля;
* NextButton\_Click – событийная процедура, непосредственно удаляет пользователя из системы за счет удаления записи БД MySQL и сообщает об этом текущему пользователю. Закрывает форму.
* Window\_Initialized – событийная функция для инициализации таймера;
* ExitButton\_Click – событийная процедура для закрытия текущей формы;
* ColorTimer\_Tick – событийная процедура таймера, каждый n промежуток времени вызывает SetColor;
* SetColor – окрашивает некоторые элементы формы в цвет системной темы;

Форма UserChangePhoneWindow.xaml

Содержит следующие поля:

* OldPhoneValidation типа Boolean – флаг валидации старого номера телефона пользователя;
* NewPhoneValidation типа Boolean – флаг валидации нового номера телефона пользователя;

Содержит следующие методы:

* ValidationOldPhone – функция сравнения старого номера телефона из БД и введенного. Берет логин текущего залогиненного пользователя из SQLite и асинхронно получает номер телефона в MD5 из MySQL, сравнивает результаты;
* NewPhoneTextBox\_TextChanged – событийная функция изменения нового телефона пользователя. Осуществляет фильтрацию нового номера телефона.
* OldPhoneTextBox\_TextChanged – событийная процедура изменения старого номера телефона. Взывает асинхронный метод ValidationOldPhone;
* PhoneConfirmButton\_Click – событийная процедура кнопки подтверждения изменения номера телефона. Записывает в SQLite сгенерированные коды подтверждения, отправляет по SMS код на новый номер телефона. Если код введен верно, то асинхронно записывает обновленные данные в MySQL;
* ExitButton\_Click – событийная процедура для закрытия текущей формы;

Перейдем к рассмотрению полей и методов отдельных класcов, не являющихся явной частью форм .xaml.

Класс CryptoClass.cs

Содержит следующие методы:

* MD5Hash – функция хеширования строки согласно алгоритму MD5. Используется для обеспечения безопасности логинов/паролей;
* Base64Encode – функция кодирования строки в стандарт Base64. Используется для хранения и передачи данных;
* Base64Decode – функция декодирования строки из стандарта Base64 в обычный текст;
* GetRandomNumber – функция для генерации псевдорандомного кода подтверждения, содержащего набор из 6 цифр;

Класс DatabaseLogicClass.cs

Содержит следующие методы:

* MySQLGet – синхронная функция с возвратом полученных данных из MySQL. Пример: использование команды SELECT;
* MySQLExecuteAsync – асинхронная функция без возврата данных данных из MySQL. Пример: использование команды INSERT;
* MySQLGetAsync – асинхронная функция с возвратом полученных данных из MySQL.
* SQLiteExecute – синхронная функция без возврата данных из SQLite;
* SQLiteGet – синхронная функция с возвратом данных из SQLite;

Класс DEMKAInputBox.cs

Содержит следующие методы:

* Конструкторы DEMKAInputBox – три перегруженных конструктора, предназначенных для генерации элементов на динамичной форме по типу InputBox.
* Windowdef – процедура, задающая параметры окна и его сгенерированных элементов;
* input\_MouseDown – событийная процедура, обрабатывающая нажатие манипулятора, меняет параметры сгенерированного textbox;
* ok\_Click - событийная процедура, обрабатывающая нажатие на кнопку подтверждения кода, осуществляет фильтрацию вводимого кода;
* ShowDialog – функция для возвращаты строкового значения введенных данных в textbox.

Класс EmailSenderClass.cs

Содержит единственный асинхронный метод SendEmailAsync для отправки e-mail на определенный адрес почты;

Класс JustTokenClass.cs

Содержит следующие поля:

* SQL\_ConnectionString – строка конфигурации подключения к MySQL. В параметрах содержит IP-адрес сервера, имя пользователя, пароль, название базы данных, а также параметры SSL;
* SMS\_Login – строка, содержащая логин для сервиса SMSAero;
* SMS\_APIKey – строка, содержащая ключ для взаимодействия с методами API сервиса SMSAero;
* Email\_Login – строка, содержащая логин e-mail, с которого происходит рассылка кодов подтверждений;
* Email\_Password – строка, содержащая пароль от e-mail, с которого происходит рассылка кодов подтверждений;
* TG\_APIKey – строка, содержащая ключ API (токен) от бота Telegram;
* TGProxy\_ServerIP – строка, содержащая IP-адрес прокси-сервера для взаимодействия с API Telegram;
* TGProxy\_ServerPort – строка, содержащая номер порта прокси-сервера для взаимодействия с API Telegram;
* TGProxy\_User – строка, содержащая имя пользователя прокси-сервера для взаимодействия с API Telegram;
* TGProxy\_Pass – строка, содержащая пароль пользователя прокси-сервера для взаимодействия с API Telegram;
* MyMasterPassword – строка, содержащая мастер-пароль программы в MD5 для подтверждения прав на добавление и удаление пользователей;
* SQLiteDBPatch – строка с относительным путем для взаимодействия с локальной базой SQLite.

Все данные для полей класса JustTokenClass.cs берутся из БД MySQL при запуске программы (за исключением SQLiteDBPatch и SQL\_ConnectionString).

Класс ParseJSONLogicClass.cs

Содержит единственное строковое поле JSON для хранения изменений в расписании в формате JSON, полученных от Flask API backend-сервера.

Содержит следующие методы:

* GetTimetableJSON – процедура получения JSON с изменениями в расписании: отправляет GET-запрос на IP-адрес backend-сервера, записывает ответ в поле json;
* GetTimetableByGroup – функция для преобразования JSON в надлежащий вид для отправки по SMS;

Класс PingClass.cs содержит единственный метод CheckForInternetConnection для проверки активного соединения путем обращения к странице https://www.google.com/.

Класс SMSSenderClass.cs

Содержит следующие методы:

* Sms\_send – функция формирования запроса для отправки сообщения по SMS;
* Check\_send - функция формирования запроса для проверки статуса SMS;
* Sms\_list - функция формирования запроса для полученияя списка отправленных SMS;
* Balance - функция формирования запроса для получения баланса текущего пользователя сервиса отправки сообщений;
* Send – функция для взаимодействия запроса с SMS-гейтом, вызывается всеми методами выше;

Класс SMSStatusClass.cs

Содержит следующие методы:

* GetSMSInfoAPI – функция запроса истории отправленных сообщений: получает JSON, возвращает двумерный строковый массив с данными;
* DataGridInput – функция для преобразования матрицы в тип DataTable для отображения данных в элементе DataGrid;

Класс TelegramClass.cs

Содержит следующие поля:

* SMS\_number – строка для хранения номера телефона для отправки уведомления;
* SMS\_message – строка для хранения текста сообщение;
* SMS\_User – строка для хранения Telegram user id пользователя. Используется для фильтрации доступа к функциям бота;
* TG\_Proxy – объект HttpToSocks5Proxy для хранения информации о прокси-сервере и альтернативного взаимодействия с BOT API Telegram (IP-адрес, порт, имя пользователя, пароль);
* Bot – объект TelegramBotClient для взаимодействия с BOT API Telegram.

Содержит следующие методы:

* IntChecker – функция проверки строки на принадлежность к числу. Используется для фильтрации ввода Telegram user id при добавлении или удалении пользователей бота.
* AdminCheckerAsync – асинхронная функция для проверки возможности доступа пользователя к функциям бота: запрашивает данные из БД MySQL;
* TelegramInit – процедура для инициализации или остановки сервера Telegram;
* BotOnMessageReceived – асинхронная событийная процедура обработки входящий сообщений пользователя боту. Содержит всю логику возможных входящих и исходящих сообщений, а также базовую фильтрацию входных данных;
* BotOnReceiveError – событийная процедура для логики обработки ошибок Telegram-сервера.

Класс ValidatorClass.cs

Содержит следующие методы:

* IsValidEmail – функция проверки на действительный e-mail с помощью System.Net.Mail.MailAddress.
* IsPhoneNumber – функция проверки на корректный номер телефона;
* ValidateMasterPassword – функция сопоставления введенного пароля с мастер-паролем, хранимым в JustTokenClass.cs;
* ValidatePassword – функция проверка пароля на сложность;

Общее взаимодействие между формами и классами можно посмтореть на рисунке №………

## 3.2 Руководство пользователя

# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

# ПРИЛОЖЕНИЕ А

# ПРИЛОЖЕНИЕ Б