ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. Н. П. ОГАРЁВА»**

Факультет математики и информационных технологий

Кафедра систем автоматизированного проектирования

УТВЕРЖДАЮ

И. о. зав. кафедрой
канд. техн. наук, доц.

_____ А. В. Шамаев
(подпись)

«27» 06 2020 г.

БАКАЛАВРСКАЯ РАБОТА

РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ АВТОМАТИЗАЦИИ РАБОТЫ ОТДЕЛА КОНТРОЛЯ КАЧЕСТВА CALL-ЦЕНТРА

Автор бакалаврской работы _____ 12.06.2020 В. О. Казанцев
(подпись) (дата)

Обозначение бакалаврской работы БР–02069964–09.03.04–08–20

Направление 09.03.04 Программная инженерия

Руководитель работы

канд. физ.-мат. наук, доц. _____ 11.06.2020 С. А. Фирсова
(подпись) (дата)

Нормоконтролер

канд. техн. наук, доц. _____ 17.06.2020г. А. А. Буткина
(подпись) (дата)

Саранск
2020

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. Н. П. ОГАРЁВА»**

Факультет математики и информационных технологий

Кафедра систем автоматизированного проектирования

УТВЕРЖДАЮ

И. о. зав. кафедрой
канд. техн. наук, доц.

_____ А. В. Шамаев
(подпись)

«30» 01 2020 г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

(в форме бакалаврской работы)

Студент Казанцев Владислав Олегович 404 группа

1 Тема «Разработка программного модуля для автоматизации работы отдела контроля качества call-центра»

Утверждена приказом № 531-С от 30.01.2020 г.

2 Срок представления работы к защите 12.06.2020 г.

3 Исходные данные для научного исследования (проектирования) документация к платформе WPF, документация к программному обеспечению Dialer Express, анкеты и аудиозаписи социологических исследований, документация к программному обеспечению X-Lite, документация по проектированию UML диаграмм, научные статьи по теме исследования.

4 Содержание выпускной квалификационной работы

4.1 Анализ предметной области разработки программного модуля для автоматизации работы отдела контроля качества call-центра

4.2 Разработка архитектуры программного модуля

4.3 Реализация программного модуля

5 Приложения

Приложение А Описание классов разрабатываемого модуля на языке C#

Приложение Б Описание класса Common_functions

Руководитель работы 31.01.2020 С. А. Фирсова

подпись, дата — инициалы, фамилия

Задание принял к исполнению 31.01.2020

подпись, дата

РЕФЕРАТ

Выпускная квалификационная работа содержит 118 страниц, 36 рисунков, 12 использованных источников, 2 приложения.

ТРАНСКРИБАЦИЯ, CALL-ЦЕНТР, ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ CALL-ЦЕНТРОВ, WEB-ПРИЛОЖЕНИЕ ДЛЯ ПРОВЕДЕНИЯ СОЦИОЛОГИЧЕСКИХ ОПРОСОВ, СОЦИАЛОГИЧЕСКОЕ ИССЛЕДОВАНИЕ, ПРИЛОЖЕНИЕ ДЛЯ WINDOWS НА ПЛАТФОРМЕ WPF, СИСТЕМА РАСПОЗНАВАНИЯ РЕЧИ С ЗАКРЫТЫМ ИСХОДНЫМ КОДОМ.

Объект исследования — автоматизация работы отдела контроля качества call-центра.

Цель работы — разработка программного модуля для автоматизации работы отдела контроля качества call-центра.

В ходе исследования были изучены принципы работы систем автоматизации процессов, тесно связанных с функционированием call-центра, методы интеграции собственной разработки со сторонними модулями, состав и особенности применения инструментов платформы WPF при разработке приложения для Windows.

Результат работы — модуль, осуществляющий проверку качества проведенных при социологическом исследовании анкет.

Степень внедрения — частичная.

Область применения — автоматизированная проверка качества работы интервьюеров при проведение социологического исследования

Эффективность — сокращение времени, требуемого на проверку качества, проведенной при социологическом исследовании анкеты; уменьшение финансовых затрат по выплатам сотрудникам отдела контроля качества заработной платы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В процессе развития технологий происходит автоматизация и компьютеризация различных областей деятельности. Это влечет за собой изменение подхода к выполнению задач, в частности отказ от труда человека в пользу автоматизированных систем. Такой переход позволяет экономить ресурсы и увеличивает скорость выполнения работы.

В данной работе мы будем говорить об автоматизации работы отдела контроля качества call-центра, а именно call-центра, занимающегося проведением социологических исследований. Социологические исследования уже давно пользуются большой популярностью как среди государственных органов, так и среди коммерческих организаций. Одним они позволяют понять, по различным критериям и индикаторам, отношение народа к ситуации в стране; другим помогают получить наибольшую выгоду путем улучшения качества оказываемых услуг или реализуемых товаров.

Как и в других сферах, с развитием технологий, в call-центрах появилась возможность автоматизации части выполняемых задач, таких как: учет рабочего времени или расчет эффективности сотрудников, автоматическое составление отчета о проведенном исследовании, да и вообще само понятие call-центра в том виде, в котором оно существует сейчас, стало возможным благодаря технологическому прогрессу: интервьюеры больше не сидят около проводных телефонов и не записывают в блокноты ответы респондентов. Напротив, у многих есть возможность работать удаленно на собственном ПК и все действия выполнять внутри автоматизированной системы.

Но не все возможные типы задач в call-центрах подверглись автоматизации. Так, например, качество проведенных анкет до сих пор контролируется людьми. Существует специальный отдел, который решает эту задачу путем просушивания проведенных анкет и последующей проверкой соответствия между текстом анкеты ответами респондента. В этом и заключается **актуальность** данной работы.

Целью выпускной квалификационной работы является разработка программного модуля для автоматизации работы отдела контроля качества call-центра (далее Модуль).

Для достижения поставленной цели определены следующие задачи:

– анализ предметной области разработки программного модуля для автоматизации работы отдела контроля качества call-центра;

– разработка архитектуры программного модуля;

– реализация программного модуля.

Разработанный программный модуль даст возможность сократить круг задач, которые выполняют сотрудники отдела контроля качества. Он будет представлять собой приложение для Windows, разработанное на платформе WPF (Windows Presentation Foundation). Обрабатываться и храниться информация будет на сервере, не требуя пользовательских ресурсов памяти. Проверку качества модуль будет осуществлять по тому же принципу, по которому сейчас это делают люди. Сложность состоит в том, что анализировать аудиозаписи анкет программными средствами неэффективно, намного удобнее работать с текстовыми данными. Для перевода аудиоданных в текст будем использовать сервисы транскрибации. Для того, чтобы была возможность встроить работу сервиса в разрабатываемое приложение, он должен иметь интерфейс API.

Таким образом, разработанный модуль позволит сэкономить деньги на зарплатах сотрудникам отдела контроля качества, а также ускорит процесс проверки работы интервьюеров. В этом состоит **практическая значимость** работы.

1. Анализ предметной области разработки программного модуля для автоматизации работы отдела контроля качества call-центра

1.1 Ознакомление с работой call-центра. Постановка задачи автоматизации

Как было сказано ранее, автоматизацию контроля качества будем производить в call-центре, который занимается проведением социологических исследований (далее call-центр-СИ).

Социологическое исследование (далее СИ) – это индивидуальное телефонное интервью, которое используется с целью получения информации, как от физических, так и от юридических лиц. Организация качественного СИ задача непростая и требует определенных знаний и навыков. В наше время уже не пристают к прохожим на улицах с целью задать интересующие вопросы. Сейчас наибольшую ценность представляет актуализация баз данных с помощью call-центров, в которых сформирована подходящая база данных и работают опытные сотрудники.

Актуализация баз данных позволяет оценить [1]:

- репутацию того или иного продукта в глазах целевой аудитории;
- объём рынка товара или услуги;
- уровень активности конкурентов и принципы их работы.

Респонденты могут различаться по своему финансовому положению и социальному статусу, поэтому к каждому нужен свой подход и свое направление в разговоре. Таким образом, грамотная работа сотрудника позволяет задействовать в СИ разные слои населения.

СИ помогает как крупным компаниям, так и компаниям средней руки, работающим в различных сферах бизнеса, выстроить правильную стратегию поведения на рынке и понять потребности и желания своих клиентов. Это эффективный способ построения современного бизнеса.

Выверенное маркетинговое исследование позволяет сэкономить средства и избежать издержек при продаже товаров, которые не пользуются спросом или попросту неактуальны. Этот метод так же используется для оценки общественного мнения по различным вопросам, оценки настроения и расположения определенных слоев населения.

СИ позволяет:

– создать точный портрет потенциального потребителя товаров или услуг;

– оценить и минимизировать риски;

– выявить новых клиентов и определить их отношение, потребности и желания;

– получить информацию о деятельности конкурентов;

– выяснить результаты всех проведенных рекламных ходов;

– собрать большой объем сопутствующей информации, которая позволит эффективнее распоряжаться своими ресурсами.

К СИ помимо коммерческих организация также прибегают и государственные органы. В большинстве случаев это происходит для понимания позиции населения страны, или отдельного региона, по ряду интересующих власть вопросов. Например, для понимания за кого в большей степени настроены голосовать на следующих выборах, как население относится к введению тех или иных реформ и т.д.

В настоящий момент уже достаточно много процессов тесно связанных с работой call-центра-СИ подверглись автоматизации. Учет рабочего времени сотрудника ведется специализированной программой, например, Dealer express, которая может не только сказать, сколько сотрудник находился на рабочем месте, но и просчитать насколько эффективно он это время провел: оценить количество пауз, вычислить так называемый коэффициент полезности: разделить общее время в разговоре на количество выполненных анкет, уведомить сотрудника о превышении допустимой нормы обработки и т.д.

Выплата интервьюерам заработной платы, также, как и решение о премировании, происходит на основании данных программы учета времени: сотруднику, устроенному на почасовую оплату, будут выплачивать сумму, равную количеству отработанных минут, умноженную на стоимость одной минуты; в то время как сотруднику на поанкетной оплате оплатят каждую проведенную анкету.

Для совершения звонков используют эмулятор телефона с виртуальными номерами и переадресацией, например, X-Lite. Это позволяет повысить процент успешных звонков, потому что люди с большей вероятностью возьмут трубку с известного номера исследовательского агентства, нежели с разных номеров из разных регионов. Также использование виртуальных номеров дает возможность сотрудникам работать удаленно со своего собственного компьютера, на который предварительно устанавливается и настраивается все необходимое программное обеспечение. Эмулятор телефона устроен таким образом, что сотрудник не набирает номера самостоятельно, за него это делает система автоматического дозвона, он только выставляет статусы звонка, такие как: «Прервано», «Отказ», «Не подходит», «Перезвон». Методику работы сотруднику объясняют при трудоустройстве, этим занимается специальный человек, в компетенции которого входит обучение новых кадров.

Сами анкеты, если мы говорим про дистанционный опрос, заполняют на специальной web-странице (wcd.survey-studio.com), функционал которой позволяет после завершения интервью в один клик переслать копию анкеты на сервер.

Описанные выше системы автоматизации работают в совокупности, что позволяет значительно упростить рабочий процесс интервьюера, так, например, отправка анкеты на сервер фиксируется программой учета рабочего времени как выполненная анкета, а проставление статуса звонка в программе учета рабочего времени дает понять эмулятору телефона, что нужно осуществлять следующий дозвон. При разработке модуля необходимо руководствоваться не только

соображениями функционала, но и исходить из соображений удобства и возможности встраивания.

При проведении СИ сохраняется аудиозапись разговора интервьюера и респондента. После завершения опроса интервьюер отправляет на сервер заполненную анкету, в которой отражено мнение респондента по всем вопросам, которые включает в себя опрос. Затем сотрудник отдела контроля качества начинает проверять соответствие между ответами респондента в аудиозаписи разговора и текстом анкеты, составленным интервьюером. Так же проверяющий смотрит на манеру общения интервьюера, и по возможности фиксирует недочеты в работе сотрудника.

Разрабатываемый модуль не сможет в полной мере заменить работу сотрудника отдела контроля качества, но позволит сократить круг задач, которые этот сотрудник выполняет, в частности автоматизировать контроль качества анкеты.

Существует много разных типов вопросов в социологических анкетах.

Типы вопросов в социологической анкете по форме [2]:

– Вопрос называется открытым, если ответ на него может быть дан в любой форме. Открытые вопросы не подразумевают подсказки и не подталкивают респондента к выбору конкретного вариант ответа. Они дают возможность выразить своё мнение во всей полноте и до мельчайших подробностей. При помощи открытых вопросов можно собрать более богатую по содержанию информацию.

– Закрытым вопрос называется в том случае, если на него в анкете приводится полный выбор вариантов ответов. Прочитав их, опрашиваемый только называет код того варианта (или нескольких), который совпадает с его мнением.

– Возможна полузакрытая форма вопроса, когда исследователь не уверен в полноте известных ему вариантов ответов на поставленный вопрос и предоставляет респонденту возможность самому дополнить их (в перечне альтернатив имеется альтернатива типа "другое").

– Вопрос, который нацелен на получение прямой непосредственной информации от респондента, называется прямым. Обычно он формулируется в личной форме: "Знаете ли Вы…?", "Что Вы думаете…?" и т. д.

– Косвенный вопрос ставится в безличной, или полу безличной форме: " Некоторые люди считают, что… А как Вы думаете?". Данный тип предполагает интерпретацию ответа на поставленный вопрос в скрытом от респондента виде. Например, вопрос о доходах может быть заменён целой серией косвенных вопросов, позволяющих составить представление о величине дохода.

Мы будем автоматизировать контроль качества опросов, состоящих только из закрытых вопросов. Это связано с большой сложностью программной интерпретации части аудиозаписи, в которой респондент дает ответ в свободной форме. Ведь разговор с респондентом не всегда складывается так, как это предписывает инструкция: человек может быть не в настроении; ответ респондента бывает нечетким или плохо сформулированным, тогда интервьюеру приходится обрабатывать такую ситуацию. И разговор в контексте одного заданного вопроса затягивается, тем самым, не позволяя программными средствами вычленить тот ответ на вопрос, который интервьюер записал в анкету.

С закрытыми вопросами ситуация обстоит иначе. Если существует конечный список ответов на вопрос, то выслушав респондента интервьюер всегда, перед переходом к следующему вопросу, сможет назвать номер ответа, который отражает мнение опрашиваемого, например, сказать: «Ответ под цифрой 2». Тем самым создав для программной интерпретации индикатор, который позволит определить номер ответа.

Ключевой идей автоматизации работы отдела контроля качества является применение процесса транскрибации – перевода аудиофайла в текст. Транскрибировав полученный в процессе интервью аудиофайл, мы перейдем к работе с текстовым файлом. Путем синтаксического анализа (парсинга) текстового файла мы выясним, верно ли сотрудник call-центра-СИ отразил мнение респондента в социологической анкете, проведя таким образом контроль

качества работы сотрудника. Подмодуль, который отвечает за парсинг, разработаем самостоятельно. А для транскрибации записи интервью (аудиофайла) воспользуемся существующей системой распознавания речи с закрытым исходным кодом, но открытым API. Открытый API подразумевает возможность встраивания готового решения в разработку.

Понятно, что функционал разрабатываемого модуля не будет ограничен исключительно проверкой качества проведенных анкет. Помимо проверки качества, существует необходимость анализа результатов такой проверки. Так же интервьюеру было бы полезно знать, насколько хорошо он справляется со своими обязанностями по расчетам системы.

Основными задачами, которые должен выполнять разрабатываемый модуль являются:

– выполнение проверки качества проведенных анкет на основе механизма транскрибации;

– информирование интервьюеров о результатах проверки качества проведенных ими анкет;

– реализация механизма администрирования (добавление новых пользователей, редактирование данных работающих сотрудников, удаление пользователей);

– реализация механизма просмотра и выгрузки истории проверок;

– реализация механизма обмена сообщениями между пользователями приложения.

Основным средством разработки выберем платформу WPF (Windows Presentation Foundation), потому что на данный момент она является самой современной полноценной платформой разработки приложений для Windows.

В качестве базы данных выберем MS Access, это обусловлено следующими факторами [3]:

– Доступный и понятный интерфейс. Большое количество справочных пособий.

– Практически неограниченные возможности экспорта данных: данные из таблиц можно переносить в Excel, Word, экспортировать в XML, публиковать в PDF одним щелчком мыши.

– Широкий выбор конструкторов для построения форм, отчетов и запросов, которые можно использовать для фильтрации данных и их удобного отображения.

– Широкие возможности импорта данных: если у вас есть табличные данные, созданные с помощью MS Word или MS Excel, вы можете легко перенести их в свою базу данных. Вы также можете осуществить импорт из простого текстового документа, XML-документа или из файлов базы данных, созданных в других СУБД (таких как dBASE или PARADOX).

1.2 Обзор сервисов транскрибации

Внедрение собственной системы распознавания речи является очень сложной, трудоемкой и ресурсозатратной задачей, которую трудно выполнить в этой работе. Поскольку системы распознавания речи с закрытым исходным кодом реализованы более эффективно и точность распознавания речи в них выше и, следовательно, их интеграция с разрабатываемым модулем является более перспективной по сравнению с аудиосистемами распознавания речи на основе открытого исходного кода.

Что касается определения закрытого исходного кода, то необходимо сказать – оно означает, что распространяются только бинарные (откомпилированные) версии программы и лицензия предполагает ограничение доступа к исходному коду программы, что препятствует ее модифицированию. Доступ к исходному коду третьим лицам обычно предоставляется при подписании соглашения о неразглашении.

Определим наиболее оптимальную, для поставленной задачи транскрибации, аудиосистему распознавания речи на базе закрытого исходного кода, лицензия которой не подходит под определение открытого ПО.

Dragon Mobile SDK

Сам инструментарий называется NDEV. Чтоб получить доступ к документации и коду, необходимо зарегистрироваться на сайте в «программе сотрудничества».

Инструментарий (SDK) содержит в себе как компоненты клиента, так и компоненты сервера. Диаграмма иллюстрирует их взаимодействие на верхнем уровне (см. рисунок 1).

Комплект Dragon Mobile SDK [4] состоит из различных вариаций кода и примеров проектов, документации, а также программной платформы, адаптирующей встраивание речевых сервисов в любое приложение.

Платформа Speech Kit framework позволяет свободно и быстро добавлять в приложения сервисы распознавания и синтеза (TTS, Text-to-Speech) речи. Данная платформа также дает возможность воспользоваться компонентами обработки речи, расположенными на сервере, через асинхронные «чистые» сетевые API, минимизируя расходы и затрачиваемые ресурсы.

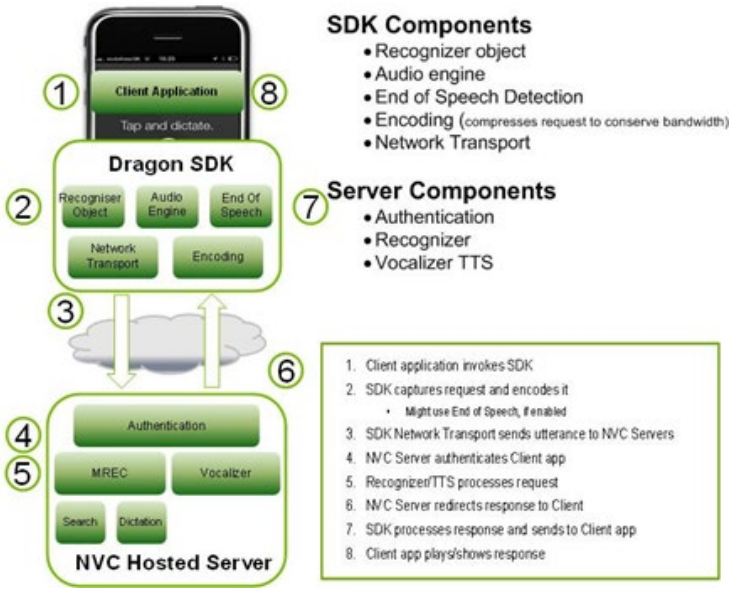


Рисунок 1 – Инструментарий SDK

Платформа Speech Kit (см. рисунок 2) является полнофункциональным высокоуровневым «фреймворком», который в автоматическом режиме администрирует все низкоуровневые сервисы.

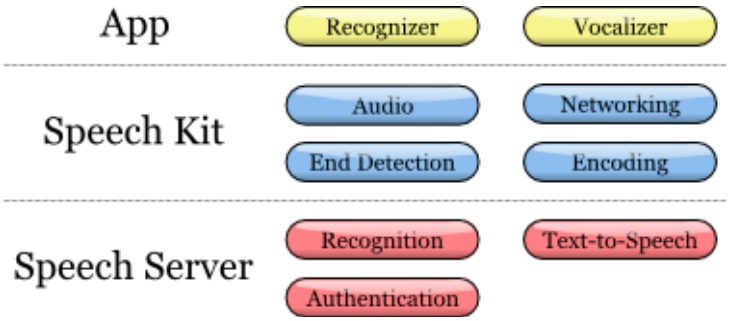


Рисунок 2 – Архитектура Speech Kit

Платформа выполняет следующие согласованные процессы:

– осуществляет управление аудио системой для записи и воспроизведения;

– сетевой компонент управляет соединением с сервером и автоматически восстанавливает соединения, разорванные по причине истекшего времени ожидания при каждом новом запросе;

– детектор окончания речи определяет, когда пользователь закончил говорить, и при необходимости может в автоматическом режиме остановить запись;

– кодирующий компонент обрабатывает потоковую аудиозапись, снижая требования к полосе пропускания и уменьшая среднее время задержки.

Система серверов отвечает за большинство выполняемых операций. Процесс транскрибации или синтеза речи полностью выполняется на сервере, обрабатывая или синтезируя аудио-поток. Кроме того, сервер проводит аутентификацию в соответствии с системными параметрами разработчика.

Google Speech Recognition API

Это продукт компании Google выпущен 14 июня 2011 года. Он позволяет осуществлять голосовой поиск, основываясь на алгоритме распознавания речи. Технология адаптирована к использованию на мобильных телефонах и ПК. Первоначально Google Voice Search поддерживал исключительно короткие голосовые запросы длительностью не более 30 секунд, необходимо было самостоятельно включать и выключать микрофон, чтобы было неудобно в использовании. В 2013 году в браузер Google Chrome была добавлена возможность распознавания потоковой непрерывной речи и фактически Google Voice Search трансформировался в Speech Input.

Для того чтобы использовать технологию Google Voice Search необходимо создать запрос типа POST, содержащий звуковые данные в формате WAV, и отправить этот запрос на актуальный сервер. Затем принять ответ от сервера, содержащий текстовые данные. Количество запросов в сутки не ограниченно [5].

Microsoft Speech API

Компания Microsoft также развивается в сфере речевых технологий. Последними их достижениями в этой области были разработки голосового ассистента Cortana и автоматической системы синхронного перевода с английского на немецкий язык и наоборот.

На текущий момент существует 3 варианта использования Microsoft Speech API [6]:

1. Добавление речевого движка для Windows-приложения и непосредственное взаимодействие с ним.
2. Speech Platforms – встраивание платформы в приложения, которые используют распространяемые компанией Microsoft диструбутивы (дополнительные пакеты для распознавания речи или же средства перевода текста в речь).

3. Embedded – встроенные решения, которые позволяют человеку взаимодействовать с автоматикой при помощи голосовых команд. Например, управление автомобилями Ford при помощи голосового ввода в ОС Windows Automotive.

Рассмотрев самые распространенные системы распознавания речи с закрытым исходным кодом стоит заметить, что по своей библиотеке данных наиболее точным, в контексте процесса транскрибации, следует считать продукт на базе Dragon Mobile SDK. Однако необходимо заметить, что этот инструмент имеет очень сложную систему лицензирования и строгие правила использования данной технологии. Поэтому возникает сложность интеграции Dragon Mobile SDK с пользовательским продуктом.

Microsoft Speech API удобно использовать там, где происходит непосредственная запись голоса и последующая транскрибация. В нашей задаче записывать голос не требуется, поэтому подключение стороннего движка или дополнительных пакетов к создаваемому приложению может уменьшить скорость его работы.

В таком случае, более правильным для наших целей и задач следует считать использование речевого инструментария Google, который является быстрым за счет больших вычислительных мощностей и легче встраиваемым. Абстрагирование от самого процесса перевода аудио в текст также можно считать положительным моментом, ведь если мы подключаем сторонние пакеты или движок, могут возникнуть вопросы совместимости, что непременно усложнит задачу. В случае с Google, от разработчика требуется только сформировать запрос, а затем, пользуясь встроенными библиотеками языка C# (программная часть в WPF разрабатывается на языке C#), получить от сервера ответ с искомым текстовым представлением. Также преимуществом распознавания речи от Google стало отсутствие ограничений по количеству запросов в сутки (у многих систем распознавания речи с закрытым исходным кодом есть такое ограничение). Единственный момент, который обязательно нужно учесть, заключается в том, что Google в последнее время начал

распространять свой речевой сервис на базе лицензионного соглашения. Поэтому для получения актуальной ссылки на соединение с серверами, необходимо зарегистрироваться на сайте Google под статусом Developer.

2 Разработка архитектуры программного модуля.

2.1 Построение UML диаграмм

UML – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем [7].

Моделирование –применяют не только для больших систем. Даже моделирование на первый взгляд простейшей системы может принести пользу. Тем не менее, определенно верно то, что чем больше и сложнее система, тем более важным становится моделирование.

Основная причина использования UML - это возможность донести свою точку зрения того, как должно выглядеть решение поставленной задачи. Некоторые понятия более ясны и наглядны, чем их альтернативы; некоторые допускают двойные толкования. Естественный язык слишком неточен и может запутать, когда речь заходит о формулировании технических особенностей. Поэтому разумно использовать UML, когда необходимо достаточно точно сформулировать и, одновременно с этим не запутаться в деталях.

Построение UML диаграмм также применяется для того чтобы взглянуть на поставленную задачу «с разных сторон». Каждая диаграмма отражает задачу с определенной точки зрения. По факту построения диаграмм складывается общая картина, дающая представление об архитектурных идеях в рамках реализуемого модуля.

Проанализировав поставленную задачу было принято решение о построении трех UML диаграмм:

- диаграммы вариантов использования;
- диаграммы состояний;
- диаграммы базы данных;

2.1.1 Диаграмма вариантов использования

Диаграмма вариантов использования - это диаграмма, которая отражает набор возможных сценариев и действующих лиц, а также структурные отношения между ними.

Действующих лиц называют актерами (Actors), а сценарии прецедентами (Case). Актер-это роль, которую пользователь играет по отношению к системе.

Один актер может быть связан с множеством прецедентов; и наоборот, один прецедент может иметь несколько актеров, которые его исполняют.

Первостепенно принято выделять действующих лиц, нежели разрабатывать возможные сценарии. Столкнувшись с большой системой, часто бывает трудно разработать список прецедентов, в таких ситуациях легче сначала составить список актеров, а затем уже для каждого актера разработать все возможные прецеденты. Но бывают так же и случаи, когда перед выделением действующих лиц, стоит разработать возможные сценарии.

Это актуально в следующих ситуациях:

– Система предоставляет различный функционал разным типам пользователей. В этом случае каждый тип пользователя является актером, и разработанные прецеденты помогут выделить действующих лиц системы.

– Некоторые прецеденты не имеют четких связей с конкретными участниками. В таком случае понятно, что, выделив актеров системы, такие сценарии могут быть не учтены.

Действующими лицами могут быть не только люди, но и внешние системы, которые обмениваются информацией с текущей системой.

Хорошим источником для определения прецедентов являются внешние события. Рассмотрение возможных событий, на которые разрабатываемая система должна реагировать, поможет выявить неучтенные сценарии.

Помимо связей между актерами и прецедентами, так же выделяют связи исключительно между прецедентами. Такие связи могут возникнуть, если в системе присутствует зависимость одного сценария от другого. В таком случае

зависимые сценарии объединяются в блок, а действующие лицо связывается непосредственно с этим блоком.

Невозможно представить ситуацию, в которой при UML моделировании отказывались бы от использования диаграммы вариантов использования. Эта диаграмма является важным инструментом в планировании и управлении итеративной разработкой. Построение диаграммы вариантов использования – это один из основных этапов разработки.

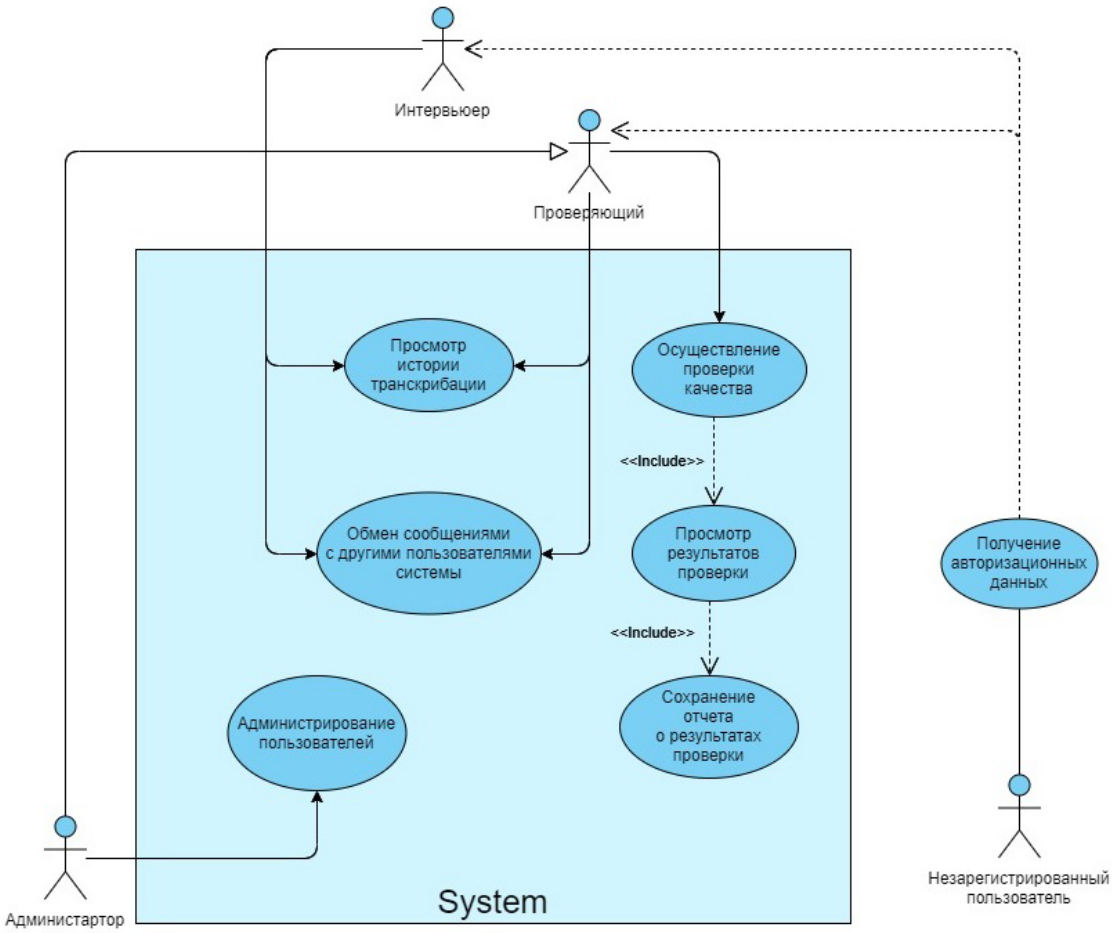


Рисунок 3 – Диаграмма вариантов использования

На рисунке 3 изображена диаграмма вариантов использования для разрабатываемого программного модуля.

На диаграмме отражены четыре действующих лица: незарегистрированный пользователь, проверяющий, интервьюер и администратор. При этом видно, что при получении авторизационных данных,

незарегистрированный пользователь может выступать как в роли проверяющего, так и в роли интервьюера. Роль незарегистрированного пользователя определяет администратор и выдает ему соответствующие авторизационные данные. Некоторые сценарии, а именно: «Осуществление проверки качества», «Просмотр результатов проверки» и «Сохранение отчета о результатах проверки» объединены в блок, так как являются зависимыми.

Самый широкий функционал предоставляется администратору системы. Помимо прецедентов, доступных проверяющему, ему так же доступен сценарий «Администрирование пользователей».

Проверяющий в свою очередь может осуществлять: «Просмотр истории транскрибации», «Обмен сообщениями с другими пользователями системы»; а также имеет доступ к блоку сценариев, в котором проводится проверка качества анкет.

Интервьюеру доступен узкий круг сценариев: «Обмен сообщениями с другими пользователями системы» и «Просмотр истории проверок качества». Это связано с тем, что основной функцией разрабатываемого модуля является проверка качества работы интервьюеров, соответственно, самим интервьюерам в системе доступны исключительно вспомогательные прецеденты.

Так же из диаграммы видно, что все сценарии, кроме одного расположены внутри прямоугольника с заголовком System. Такие прецеденты реализуются средствами разрабатываемого модуля, без использования сторонних приложений или сервисов, а сценарий «Получение авторизационных данных» выполняется вне системы, после исполнения соответственного сценария «Регистрация пользователей».

2.1.2 Диаграмма состояний

Диаграмма состояний – это широко используемый метод описания поведения системы. Она описывают все возможные состояния, которые конкретный объект может принимать и как изменяется состояние объекта в результате происходящих событий.

Существует множество форм диаграмм состояний, каждая из которых имеет различную семантику. Любая диаграмма состояний начинается с так называемой стартовой точки (Start point) и инициации перехода к начальному состоянию. Переходы между состояниями, как и связи между действиями, обозначаются однонаправленными стрелками. Рядом со стрелками может располагаться текстовая метка, синтаксис которой будет отражать смысл перехода: переход, связанный с возникновением события или переход, связанный с выполнением действия. Если переход не имеет текстовой метки, это означает, что переход происходит сразу же, как только действие, связанное с ним, завершается.

Общепринято, что из каждого состояния, если это состояние не является промежуточным (не включает ни одного действия) выполняется только один переход, поэтому все действия, которые ведут к смене состояния должны быть взаимоисключающими.

Если состояние реагирует на возникающее событие действием, которое не вызывает смены состояния, то такое действие необходимо выделить в отдельный блок.

Диаграммы состояний хорошо описывают поведение объекта в контексте нескольких прецедентов, в тоже время, данная диаграмма не так хороша, когда речь заходит о взаимодействии нескольких объектов. Используя диаграмму состояний, не нужно пытаться построить ее для каждого класса системы. Построение данной диаграммы актуально только в том случае, если это поможет понять поведение системы в целом. Диаграмма состояний для разрабатываемого модуля изображена на рисунке 4.

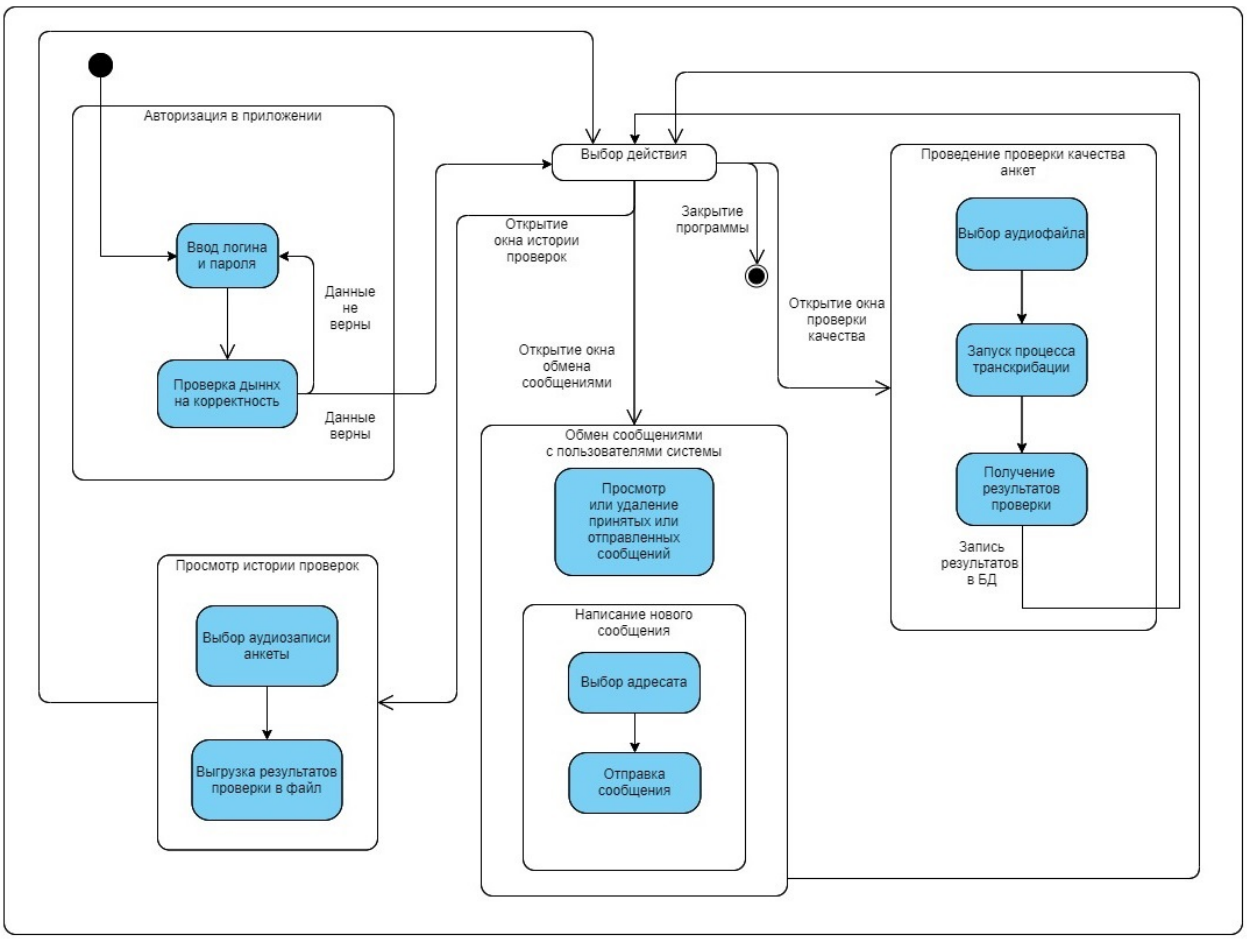


Рисунок 4 – Диаграмма состояний

Для демонстрации поведения разрабатываемого программного модуля было выделено 6 состояний, одно из которых промежуточное: «Авторизация в приложении», «Просмотр истории проверок», «Проведение проверки качества анкет», «Обмен сообщениями с пользователями системы», «Написание нового сообщения» и «Выбор действия». Данная диаграмма отражает поведение системы при работе проверяющего. Доступный ему функционал описан в пункте 2.1.1.

Начальным состоянием является «Авторизация в приложении». В случае ввода корректных авторизационных данных пользователь попадает на главное окно программы, которое соответствует состоянию «Выбор действия». Выбор нужной пользователю опцию переводит систему в одно из следующих состояний: «Просмотр истории проверок», «Обмен сообщениями с пользователями системы» или «Проведение проверки качества анкет». Событие,

связанное с этим переходом, указано в соответственной текстовой метке. В какое бы состояние система не перешла, пользователь всегда может вернуться на главное окно программы.

Во всех состояниях, кроме состояния «Обмен сообщениями с пользователями системы», действия объединены в блоки (связаны однонаправленными стрелками), это показывает, что порядок выполнения действий важен. В свою очередь, внутри состояния «Обмен сообщениями с пользователями системы»» порядок действий не принципиален: напишет ли пользователь новое сообщение или сначала перейдет к редактированию уже существующего, не играет никакой роли.

Корректно завершить работу программы пользователь сможет находясь только в главном окне приложения, которому, как уже было сказано, соответствует состояние «Выбор действия». Это связано с тем, что новые окна будут открываться в диалоговом режиме поверх главного окна, таким образом, чтобы закрыть главное окно, необходимо будет закрыть все окна, открытые поверх него, что в свою очередь переведет систему в состояние «Выбор действия».

2.1.3 Диаграмма базы данных (ER-диаграмма)

Диаграммы баз данных обеспечивают визуальное представление структуры и отношений таблиц в базе данных [8]. Они используют определенный набор элементов, таких как: прямоугольники и соединительные линии, чтобы изобразить взаимосвязь сущностей и отношение их атрибутов. Любая реляционная база данных может быть представлена совокупностью эквивалентных реляционных таблиц, поэтому структуру таких баз данных достаточно легко отобразить на данной диаграмме.

Основная цель создания диаграммы баз данных – формирование пользовательского восприятия данных и согласование большого количества технических аспектов, связанных с проектированием.

Дигамма базы данных для разрабатываемого программного модуля представлена на рисунке 5.

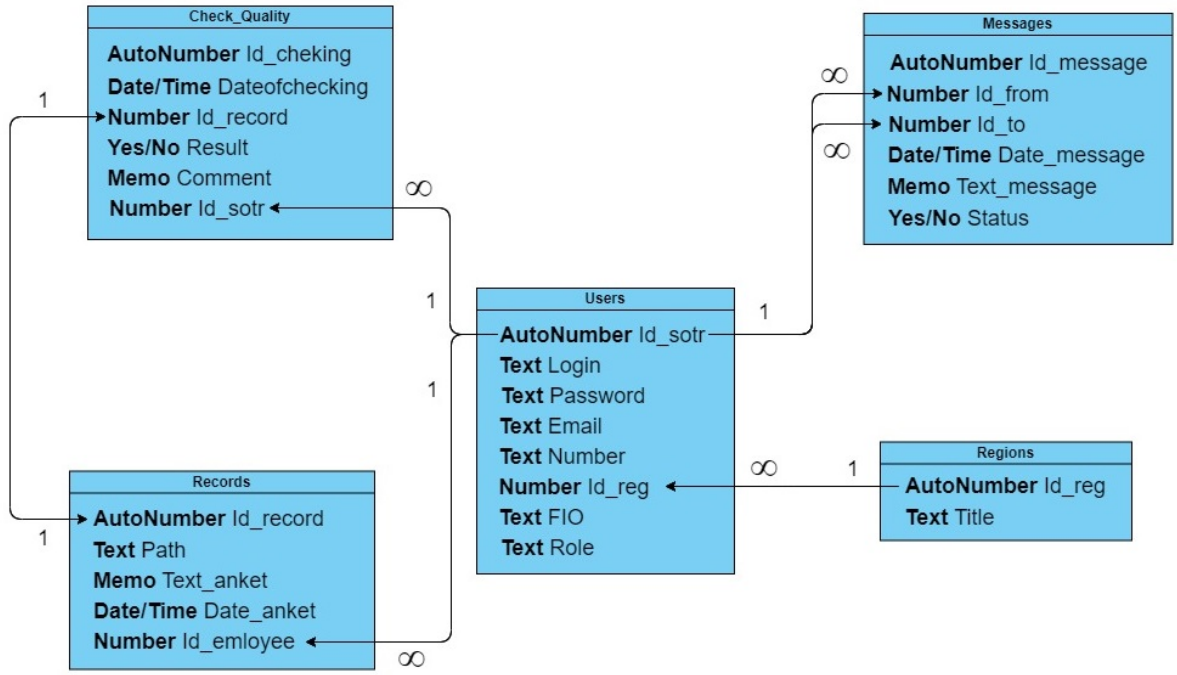


Рисунок 5 – Диаграмма базы данных

Как было сказано ранее, при разработке в качестве базы данных будем использовать MS Access. На диаграмме прямоугольники отображают таблицы базы данных, а текстовые метки внутри прямоугольников соответствующие столбцы.

Спроектированная база данных состоит из 5 таблиц:

– таблица Users хранит данные обо всех пользователях системы: об администраторах, о проверяющих и об интервьюерах; данные пользователя содержат: идентификатор сотрудника, логин, пароль, адрес электронной почты, номер телефона, идентификатор региона, ФИО и должность;

– таблица Regions содержит идентификаторы и названия регионов;

– таблица Records хранит информацию о проведенных анкетах: идентификатор записи, путь до файла аудиозаписи анкеты, текст анкеты, дата проведения анкеты и идентификатор интервьюера;

– таблица Check_Quality содержит информацию о проверках качества анкет: идентификатор проверки, дата проверки, идентификатор аудиозаписи, результат проверки, комментарий к результату и идентификатор проверяющего;

– таблица Messages хранит информацию о сообщениях между пользователями: идентификатор сообщения, идентификатор адресанта, идентификатор адресата, дату отправки сообщения, текст сообщения и статус о прочтении.

В MS Access используются специфические названия типов данных. Ниже представлены описания использованных при проектировании типов [9]:

– Текстовый (Text) — используется для хранения текста или комбинаций алфавитно-цифровых знаков, не применяемых в расчетах (например, код товара). Максимальная длина поля 255 знаков.

– Поле MEMO (Memo) — используется для хранения обычного текста или комбинаций алфавитно-цифровых знаков длиной более 255 знаков. Максимальный размер поля 1 Гбайт знаков или 2 Гбайт памяти (2 байта на знак) при программном заполнении полей, и 65 535 знаков при вводе данных вручную в поле и в любой элемент управления, связанный с этим полем.

– Числовой (Number) — служит для хранения числовых значений (целых или дробных), предназначенных для вычислений, исключением являются денежные значения, для которых используется тип данных Денежный (Currency). Размер поля 1, 2, 4 и 8 байтов, или 16 байтов (если используется для кода репликации) зависит от типа чисел, вводимых в поле.

– Дата/время (Date/Time) — используется для хранения значений даты и времени в виде 8-байтовых чисел двойной точности с плавающей запятой. Целая часть значения, расположенная слева от десятичной запятой, представляет собой дату. Дробная часть, расположенная справа от десятичной запятой, — это время.

– Счетчик (AutoNumber) — используется для уникальных числовых 4-байтовых значений, которые автоматически вводит Access при добавлении записи. Вводимые числа могут последовательно увеличиваться на указанное

приращение или выбираться случайно. Обычно используются в первичных ключах.

– Логический (Yes/No) — применяется для хранения логических значений, которые могут содержать одно из двух значений: Да/Нет, Истина/Ложь или Вкл/Выкл.

Спроектированная база данных приведена к третьей нормальной форме:

– каждое поле таблицы содержит только одно значение;

– в каждой таблице присутствует первичный ключ (первый столбец с типом данных AutoNumber), причем каждый столбец таблицы зависит от этого первичного ключа;

– в таблицах отсутствуют транзитивные зависимости (хранение «лишней» информации).

Поговорим о связях между таблицами. Все таблицы связаны с таблицей Users: таблица Regions по полю Id_reg, остальные по полю Id_sotr. Для некоторых таблиц одной связи недостаточно, так, например, таблица Check_Quality связана с таблицей Users и с таблицей Records, потому что содержит результаты проверок, а эти результаты включают как данные о аудиозаписи анкеты, так и данные о проверяющем. Таблица Records так же связана с таблицей Users, и эта связь может, на первый взгляд, показаться избыточной, ведь данные о сотруднике можно получить через таблицу Check_Quality. Но на самом деле ситуация обстоит иначе. Это связано с тем, что таблица Users хранит в себе информацию о разных группах работников. В таблице Records поле id_employee хранит идентификатор интервьюера, а поле Id_sotr в таблице Check_Quality хранит информацию о проверяющем. Понятно, что эти идентификаторы не могут быть равны.

Типы связей между таблицами:

– таблица Users связана с таблицей Check_Quality соотношением один ко многим, потому что один проверяющий может осуществлять проверку качества нескольких анкет;

– таблица Users связана с таблицей Records соотношением один ко многим, так как за одним интервьюером может числиться несколько анкет;

– таблица Records связана с таблицей Check_Quality соотношением один к одному, потому что не имеет смысла осуществлять проверку качества одной и той же анкеты несколько раз;

– таблица Regions связана с таблицей Users соотношением один ко многим, так как несколько сотрудников могут работать в одном и том же регионе;

– таблица Users связана с таблицей Messages в обоих случаях соотношением один ко многим, потому что адресат может получать сообщения сразу от нескольких пользователей, также, как и адресант может переписываться с несколькими людьми.

2.2 Взаимодействие модуля с существующими автоматизированными системами

Взаимодействие модуля с существующими автоматизированными системами представлено на рисунке 6.

Данный рисунок отражает принцип встраивания разрабатываемого модуля в работу существующих автоматизированных систем.

Dialer Express – это программное обеспечение для учета рабочего времени интервьюеров. X-Lite – это эмулятор телефона с возможностью переадресации и подключения виртуальных номеров. Survey Studio – это интернет сайт, на web – страницах которого заполняют анкеты при социологическом опросе. Все эти системы работают совокупно. Так, например, Dialer Express дает команду X-Lite осуществлять следующий звонок, когда текущий уже обработан, в свою очередь, X-Lite пересылает аудиозапись Dialer Express. Если по окончанию разговора интервьюер проставляет в интерфейсе Dialer Express статус «Успешно», то текст анкеты вместе с аудиозаписью разговора отправляются на сервер.

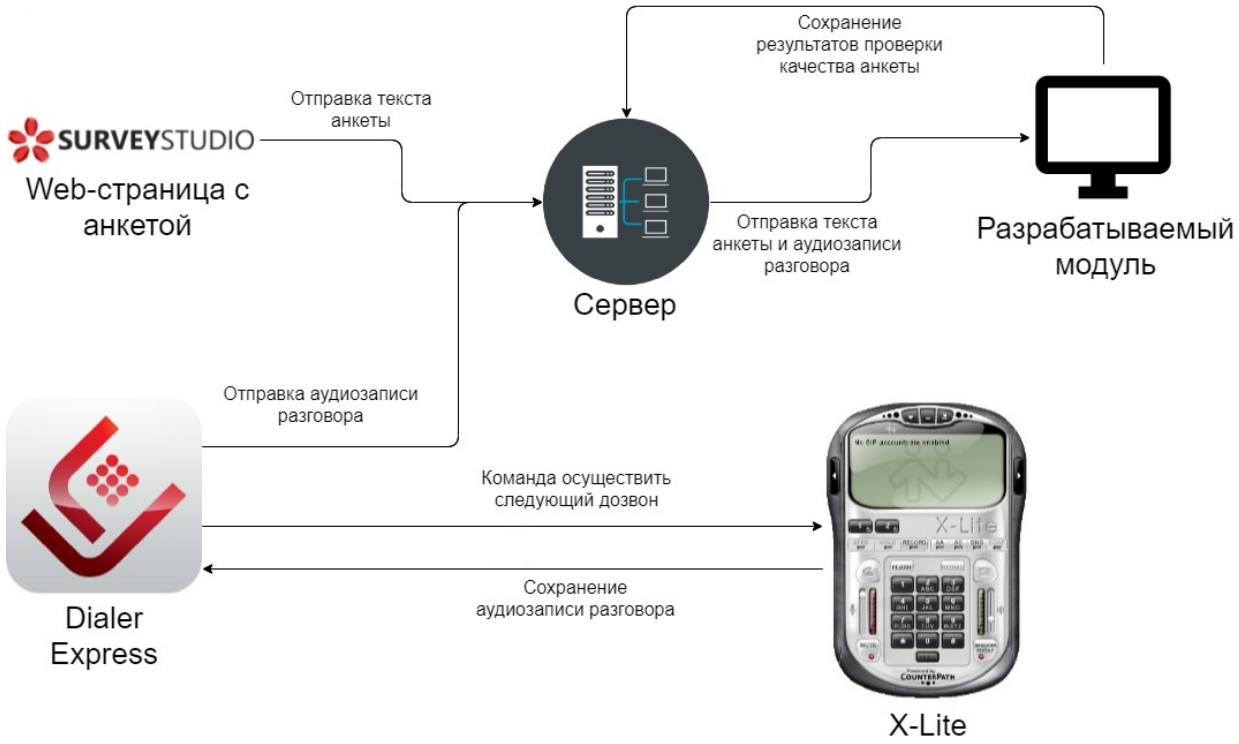


Рисунок 6 – Взаимодействие модуля с существующими автоматизированными системами

Разрабатываемый программный модуль должен дополнять уже имеющийся функционал, не влияя на совокупную работу существующих систем.

Как видно из рисунка, модуль обменивается данными только с сервером: получает аудиозапись разговора и текст анкеты, а отправляет результаты проверки качества, то есть не изменяет информацию, а добавляет новую, тем самым минимизируя влияние на существующее программное обеспечение.

2.3 Взаимосвязь компонентов модуля

Взаимосвязь компонентов модуля отражена на рисунке 7. При проектировании было выделено три компонента: WPF-приложение, сервер системы и сервера Google Voice. Понятно, что при рассмотрении внутреннего устройства приложения, можно выделить больше компонентов, но так или

иначе, они являются средствами платформы WPF, а при составлении данной схемы учитывались внешние, по отношению друг к другу, сервисы.

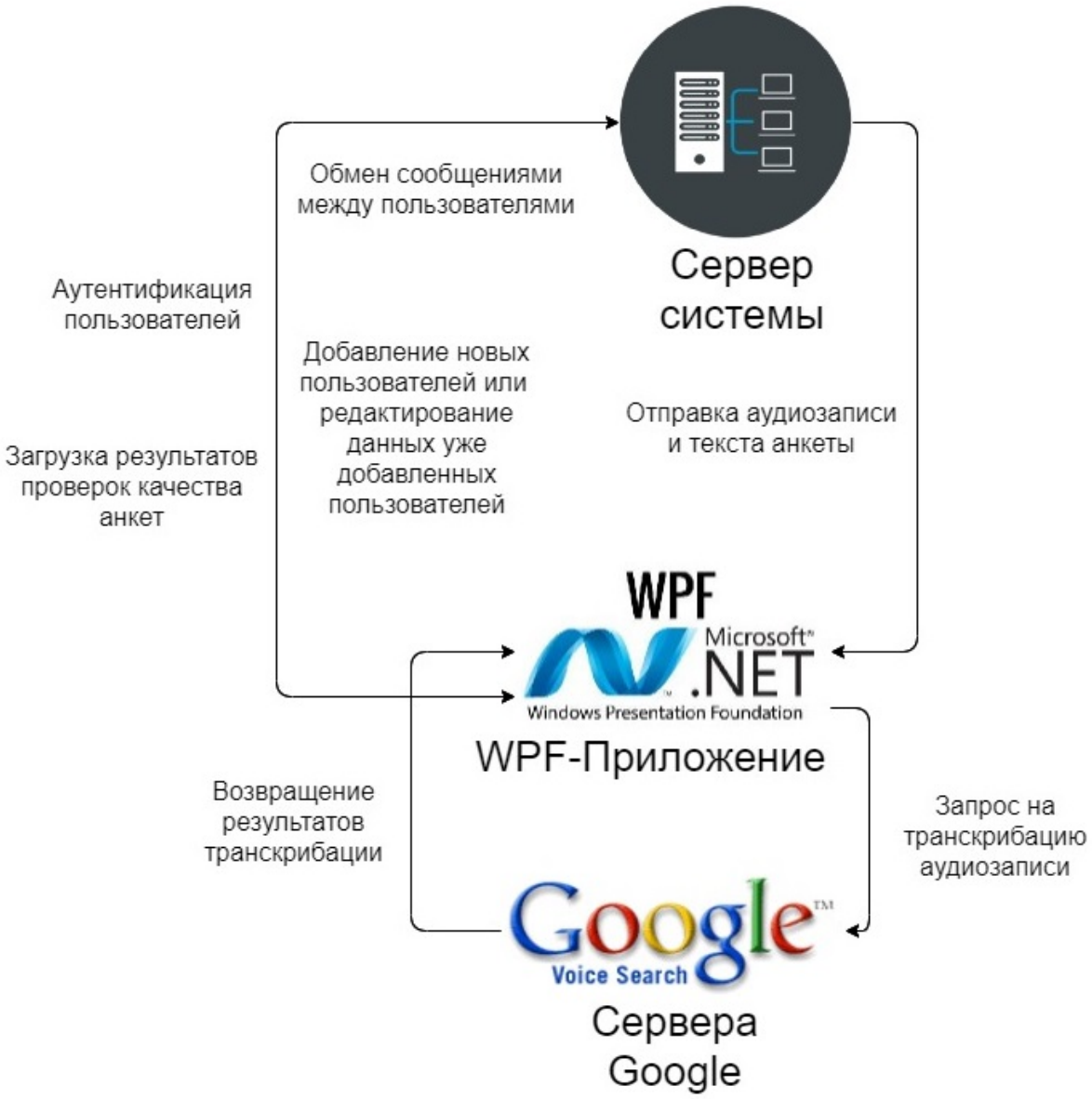


Рисунок 7 – Взаимосвязь компонентов модуля

Из рисунка видно, что процесс транскрибации происходит на серверах Google Voice: приложение посылает запрос в виде аудиофайла, переведенного в массив байтов, а в качестве ответа получает текстовую интерпретацию отправленной аудиозаписи. Абстрагироваться от данного процесса позволяет интерфейс встраиванию Google Voice, который реализуется при помощи POST

запроса. При выборе другой рассмотренной системы распознавания речи абстрагирование от процесса транскрибации было бы невозможным.

WPF-приложение обменивается с сервером данными, а именно: загружает результаты проверок качества, причем как с сервера, при загрузке окна истории проверок, так и на сервер после проведения проверки качества анкет; принимает от сервера и отправляет на сервер сообщения пользователей; сохраняет на сервере информацию о новом пользователе при регистрации и проверяет авторизационные данные при аутентификации; принимает от сервера аудиозапись и текст анкеты.

3 Реализация программного модуля

Существует несколько особенностей реализации приложения на платформе WPF. Каждое окно приложения является отдельным классом.

Часть реализации данного класса располагается в файле с расширением .xaml, который состоит из иерархического набора вложенных друг в друга именованных XML-элементов, причем каждый элемент может иметь любое количество XML-атрибутов и дочерних элементов. Элементы оформляются в виде тэгов. Между этими тэгами располагается содержимое элемента, которое может представлять собой обычный текст и/или другие (дочерние) элементы. Число уровней вложенности элементов может быть любым [10].

Все атрибуты одного элемента должны иметь различные имена, в то время как его дочерние элементы могут иметь совпадающие имена. Любой XML-файл должен содержать единственный XML-элемент верхнего уровня, называемый корневым элементом [10].

Добавлять новые элементы и изменять атрибуты существующих можно как с помощью непосредственного редактирования файла, так и с помощью окна дизайнера. Окно дизайнера - это визуальный редактор, который отображает макет окна приложения в соответствии выбранным XML-файлом.

При выводе текста XML-файлов мы не будем указывать атрибуты корневого элемента, предшествующие атрибуту Title, поскольку они генерируются автоматически и не требуют изменения.

Вторая часть определения класса окна хранится в файле с расширением .cs, который содержит частичное описание класса на языке C#. Данный файл по умолчанию включает конструктор без параметров, в котором вызывается метод InitializeComponent(), обеспечивающий начальную инициализацию компонентов окна. Все действия с компонентами можно выполнять только после их начальной инициализации, поэтому пользовательский код добавляется в конструктор после вызова данного метода [10].

Так как, и файл с расширением .xaml, и файл с расширением .cs хранят информацию о классе окна, то при реализации методов класса на языке C# можно обращаться к XML-элементам, потому что они находятся в одной области видимости.

При выводе текста .cs файлов, указывать используемые в файле библиотеки, а также пространства имен мы не будем, потому что используемые библиотеки в разных файлах часто дублируются, а пространство имен у всех классов разрабатываемого модуля совпадает. Полный код файлов с расширением .cs представлен в Приложении А.

В процессе реализации выяснилось, что ряд классов разрабатываемого модуля имеет схожие по смыслу методы, некоторые из них полностью дублируются, другие по большей части совпадают. Поэтому было принято решение вынести такие методы в отдельный класс, названный Common_functions, задав им модификатор static.

Поговорим о предназначении таких методов:

– Метод RGBConver (string) позволяет привести строковое представление цвета к объекту типа Brush. Он используется для задания цвета элементов.

– Метод Show_message (TextBlock, string, bool) используется для вывода строки в соответствующий текстовый блок. Переменная типа bool используется для задания цвета текста: значение true задает зеленый цвет, а значение false – красно-оранжевый.

– Метод HideMessage (TextBlock) используется для скрытия заданного текстового блока.

– Метод SQL_question (string) осуществляет запрос к базе данных. При помощи данного метода могут выполняться все SQL инструкции, за исключением инструкции Select. Это связано с тем, что инструкция Select подразумевает возвращение объекта класса OleDbDataReader, а метод SQL_question (string) имеет тип void.

– Метод Regular (string, string, bool) используется для поиска в строке частей, удовлетворяющих заданному регулярному выражению. Данный метод имеет тип dynamic, это означает, что тип возвращаемого значения может быть различным. Если переменная типа bool имеет значение true, то возвращается объект класса MatchCollection, содержащий в себе все найденные совпадения, а если она принимает значение false, то возвращается только первое найденное совпадение.

Код методов класса Common_functions представлен в Приложении Б.

3.1 Окно авторизации

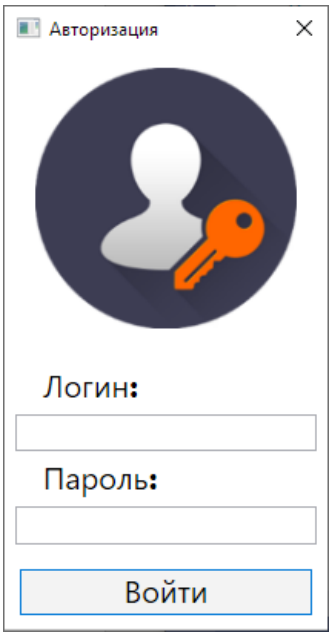


Рисунок 8 – Окно авторизации

Код XML-файла окна авторизации:

```
<Window x:Class="Check_Quality.MainWindow"
…
    Title="Авторизация" ResizeMode="NoResize" WindowStartupLocation="CenterScreen"
SizeToContent="WidthAndHeight" Background="White">
  <StackPanel>
    <Image Source="Resources/authorization.png" Margin="20,10,20,20" Width="170"/>
    <StackPanel>
```

```
        <Label Content="Логин:" FontFamily="Elephant" FontSize="20" Margin="20,0,0,0"/>
        <TextBox Name="Log" FontFamily="Arial" FontSize="18" Margin="7,0" Padding="1"  />
        <Label Content="Пароль:" FontFamily="Elephant" FontSize="20" Margin="20,0,0,0"/>
        <PasswordBox Name = "pass" FontFamily="Arial" FontSize="18" Margin="7,0,7,8"
Padding="1"  />
    </StackPanel>
    <TextBlock Margin="0,-16,0, 0" Name ="Error_message" Foreground="#FFFF6600"
TextAlignment="Center" />
    <Button Content="Войти" FontFamily="Elephant" FontSize="20" IsDefault="True"
Margin="10,8,10,10" Click="auth_but_click"
            Background="#FFF3F3F3" />
  </StackPanel>
</Window>
```

При анализе структуры следующих окон приложения, будем останавливаться только на элементах и атрибутах, которые не встречались ранее.

Рассмотрим использованные при реализации окна авторизации атрибуты элементов:

– Свойство окна Title задает текст заголовка.

– Свойство окна ResizeMod показывает доступны ли в правом верхнем углу кнопки изменения размеров окна. Оно может принимать значения: NoResize – пользователь не может изменить размер окна, CanMinimize – окно может быть только свернуто или восстановлено, CanResize – размер окна может быть изменен; а также CanResizeWithGrip.

– Свойство окна WindowStartupLocation отвечает за расположение окна в момент его открытия. Оно может принимать значения: CenterOwner – расположение в центре окна-родителя, CenterScreen – расположение в центре экрана, Manual – расположение окна задается в коде .cs файла.

– Свойство окна SizeToContent указывает на возможность изменения размеров окна в соответствии с размерами его содержимого. Оно может принимать значения: Height – изменяется только высота окна, Manual – размер

окна не изменяется, Width – изменяется только ширина окна, WidthAndHeight – изменяется как высота, так и ширина окна.

– Свойство окна Background задает цвет фона окна.
– Свойство Foreground задает цвет текста элемента.
– Свойство Margin задает внешние поля элемента.
– Свойство Padding задает внутренние поля элемента.
– Свойство Width задает фиксированную ширину элемента.
– Свойство FontFamily задает тип шрифта элемента.
– Свойство FontSize задает размер шрифта элемента.
– Свойство Content задает содержимое элемента (данное свойство очень схоже со свойством Text).

Поговорим об элементах окна авторизации.

Изображение в верхней части окна задает элемент Image. Свойство Source содержит путь до файла изображения. Для того, чтобы не привязывать работу модуля к конкретной операционной системе, создадим внутри проекта каталог Resources, в который поместим все используемые ресурсы.

Элемент StackPanel является группирующим элементом. Это означает, что внутри него могут располагаться другие элементы. В разрабатываемом приложении в качестве группирующих элементов будем использовать только элементы StackPanel. Это связано с тем, что в процессе работы приложения нет необходимости изменять расположение элементов окна. Такой функционал предоставляют другие группирующие элементы, такие как Canvas или Grid, в которых существует возможность, при помощи средств языка C#, полностью видоизменить маета окна в процессе работы программы.

Элемент Label используется для задания текстовых меток, текст которых недоступен пользователю для изменения.

Элемент TextBox представляет собой прямоугольное поле ввода с выделенной границей. Этот элемент используется для ввода пользовательских данных.

Элемент PasswordBox по структуре схож с элементом TextBox, только вводимые пользователем данные отображаются в недоступном для прочтения виде. Этот элемент чаще всего используется для ввода паролей.

Элемент TextBlock схож с элементом Label и используется для вывода текстовой информации, но он предоставляет больше возможностей для форматирования текста.

Элемент Button задает кнопку. Кнопки как правило используются для исполнения какого-либо действия. Действия напрямую связаны с событиями, которые возникают при определенных обстоятельствах, так, например, событиями могут являться: клик по кнопке мыши, потеря или прием фокуса элементом, попадание курсора на элемент.

События могут возникать по инициации практически любых XML-элементов. При разработке приложения на WPF, для задания функционала используют обработчики событий. Обработчик события – эта функция, которая выполняет ряд действий при возникновении определенного события, одним из аргументов которой является объект это событие сыниициировавший.

Обработчики событий определяются в файле с расширением .cs, при помощи средств языка C#, как методы класса.

Код класса окна авторизации на языке C#:

```
public partial class MainWindow : Window
{
    Main Main = new Main();
    public MainWindow()
    {
        InitializeComponent();
    }
    OleDbConnection connect = new OleDbConnection(Common_functions.connection);
    private void auth_but_click(object sender, RoutedEventArgs e)
    {
        OleDbDataReader read;
        try
        {
```

```
            if (connect.State != ConnectionState.Open)
                connect.Open(); // открываем базу данных
            OleDbCommand cmd = new OleDbCommand("SELECT * FROM Users WHERE Login
= '" + Log.Text + "'", connect); // создаём запрос
            read = cmd.ExecuteReader(); // получаем данные
        }
        catch
        {
            read = null;
        }
        if (read != null)
        {
            if (read.Read() && pass.Password == read.GetString(2))
            {
                string Role = read.GetString(7);
                if (Role == "Admin")
                {
                    Main.admin_button.Visibility = Visibility.Visible;
                    Main.admin_button.Margin = new Thickness(10, -5, 10, 15);
                }
                else if(Role == "Employee")
                {
                    Main.Transc_but.Visibility = Visibility.Hidden;
                    Main.Transc_but.Margin = new Thickness(10,0,10,-30);
                }
                Main.id = (int)read.GetValue(0);
                Main.Show();
                Close();
            }
            else
                Common_functions.Show_message(Error_message, "Введены некорректные
данные", false);
        }
        else
```

```
            Common_functions.Show_message(Error_message, "Не удается получить данные из БД.\nОбратитесь к адмиристратору", false);
            connect.Close();
        }
    }
```

Окно авторизации задает класс MainWindow.

Функции библиотеки System.Data.OleDb используются для работы с базой данных. В переменной connection, расположенной в классе Common_functions хранится строковое представление пути до базы данных.

Метод auth_but_click(…) является обработчиком события клика по кнопке «Войти».

В первую очередь при помощи конструкции try{} catch{} происходит поиск введённого логина в базе данных. В случае отсутствия связи с базой данных или неверно введенной авторизационной информации в текстовый блок, предназначенный для коммуникации с пользователем, выводится соответствующее сообщение. Пример такой ситуации отображен на рисунке 9.

Если введенный логин существует в базе данных, то происходит проверка соответствия пары логин-пароль. Так как разным группам пользователей доступен разный функционал, то разумно отслеживать тип пользователя при авторизации в приложении, и в соответствии с этим изменять содержимое главного окна. Значение типа пользователя хранится в столбце Role таблицы Users. Открытие главного окна происходит при помощи метода Show() объекта класса Main.

При работе с базой данных важно отслеживать значения свойства ConnectionState объекта класса OleDbConnection. После реализации необходимых SQL инструкций нужно в обязательном порядке закрывать соединение при помощи метода Close(), иначе дальнейшая работа с базой данных может вызывать необрабатываемое исключение AccessViolationException.

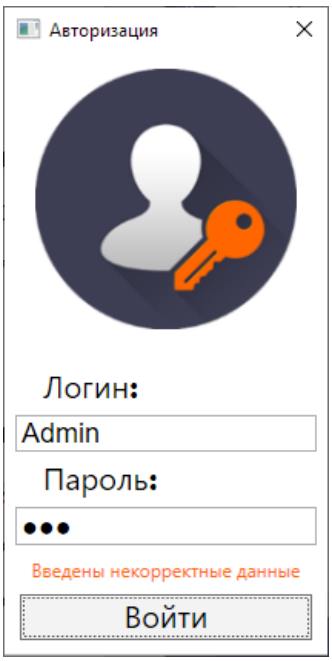


Рисунок 9 – Вид окна авторизации при неверно введенных авторизационных данных

3.2 Главное окно программы

Как было сказано ранее, вид главного окна программы зависит от типа пользователя. На рисунках 10, 11 и 12 отображен вид главного окна для администратора, для проверяющего и для интервьюера.

Код XML-файла главного окна программы:

```
<Window x:Class="Check_Quality.Main"
…
      Title="Главное окно" WindowStartupLocation="CenterScreen" SizeToContent="WidthAndHeight" ResizeMode="CanMinimize">
   <StackPanel>
      <Image Source="Resources/Main.png" Margin="20,10,20,20" Width="170" />
      <StackPanel>
         <Button Name="Transc_but" Content="Транскрибация" FontFamily="Elephant" FontSize="20" IsDefault="True" Margin="10,0,10,10"
            Background="#FFF3F3F3" Click="Transc_but_click" />
         <Button Content="История проверок" FontFamily="Elephant" FontSize="20" Margin="10,0,10,10"
```

```
            Background="#FFF3F3F3" Click="History_but_click" />
        <Button Name="Message_but" Content="Сообщения" FontFamily="Elephant"
FontSize="20" Margin="10,0,10,15"
            Background="#FFF3F3F3"  />
        <Button Name ="admin_button"  Content="Администрирование" FontFamily="Elephant"
FontSize="20" Margin="10,-40,10,10"
            Background="#FFF3F3F3" Padding="5,1" Visibility="Hidden"
Click="Admin_button_click" />
      </StackPanel>
   </StackPanel>
</Window>
```

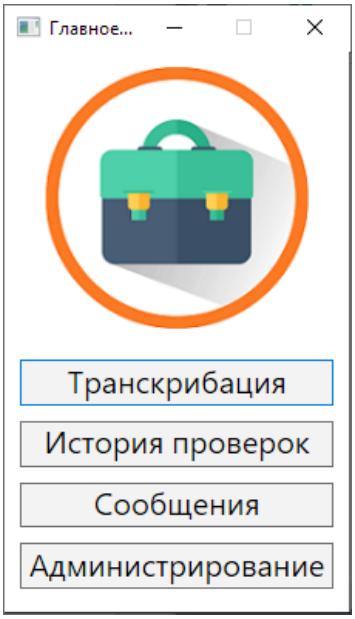


Рисунок 10 – Вид главного окна программы для администратора

Класс Main задает главное окно программы, содержимое которого располагается внутри иерархического набора группирующих элементов StackPanel. Причем панель с кнопками выделена в отдельный блок. Подобное представление позволяет редактировать расположение всего блока элементов в целом, задавая соответствующие значения внутренних или внешних полей.

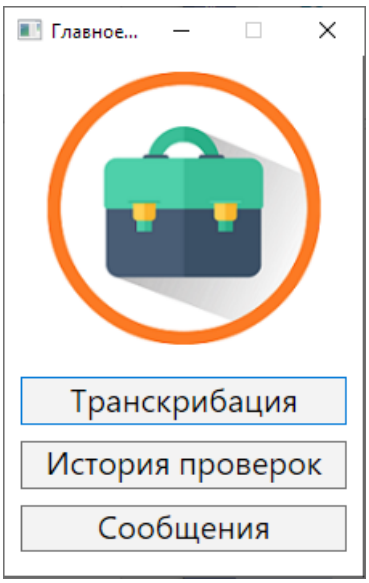


Рисунок 11 – Вид главного окна программы для проверяющего

Поговорим об изменении свойств элементов окна в соответствии с типом пользователя. Как можно заметить, свойство Visibility последнего в списке элемента Button, при инициализации объекта класса главного окна, имеет значение Hidden, то есть элемент скрыт и не доступен пользователю. Изменения свойства Visibility для ряда кнопок происходит в обработчике auth_but_click класса MainWindow до открытия главного окна.

Если типом пользователя является администратор, то свойство Visibility элемента Button с названием «Администрирование» примет значение Visible, делая элемент доступным пользователю.

Если типом пользователя является проверяющий, то главное окно откроется в том виде, который задает приложенный выше листинг XML-файла, без изменений.

Если типом пользователя является интервьюер, то элемент Button с названием «Администрирование» останется скрытым, а свойство Visibility элемента Button с названием «Транскрибация» изменит значение с Visible на Hidden.

Поговорим об обработчиках событий клика по кнопкам, расположенных в главном окне.

Клик по каждой кнопке вызывает открытие соответствующего окна: создается объект класса окна, затем его свойство Owner, хранящее сведения об элементе-родителе, связывается с объектом класса главного окна, после чего вызывается метод ShowDialog(), который открывает окно в диалоговом режиме.

Помимо методов-обработчиков, класс Main содержит поле int id, которое хранит информацию об идентификаторе пользователя, работающего в системе. Это информация необходима для корректной работы остальных окон системы.

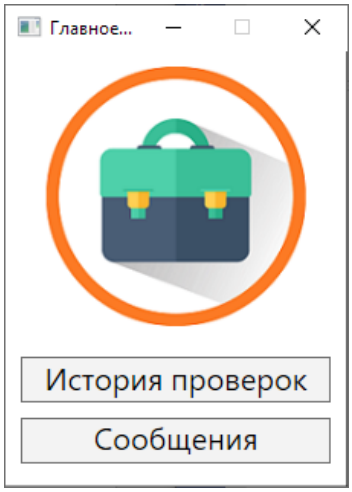


Рисунок 12 – Вид главного окна программы для интервьюера

3.3 Окно проверки качества анкет

Окно проверки качества анкет задает класс Tranc_window, оно доступно только проверяющим и администраторам.

При помощи средств класса Tranc_window происходит проверка качества проведенных интервьюерами анкет. Окно проверки качества анкет изображено на рисунке 13, оно открывается после клика по кнопке «Транскрибация» в главном окне.

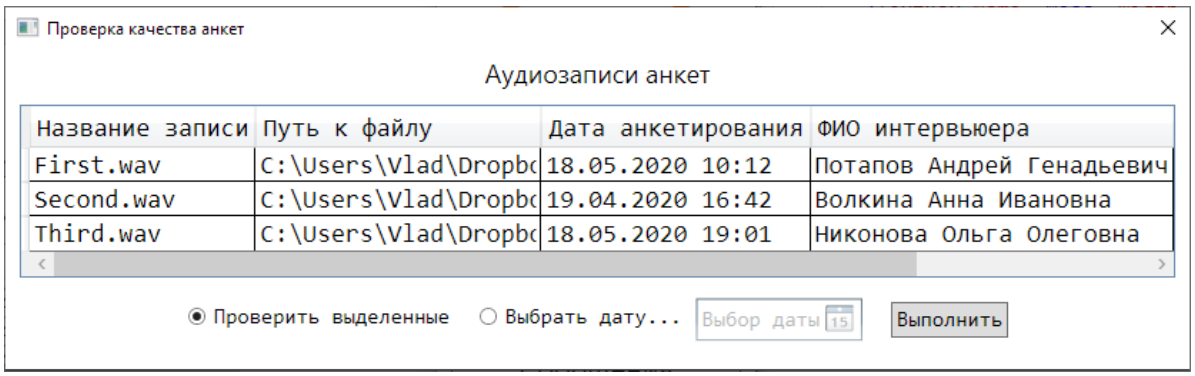


Рисунок 13 – Окно проверки качества анкет

Код XML-файла окна проверки качества анкет:

```
Window x:Name="Transc_window" x:Class="Check_Quality.Tranc_window"
…
    Title="Проверка качества анкет" SizeToContent="Height"
WindowStartupLocation="CenterScreen" ResizeMode="NoResize" Width="845">
  <StackPanel Margin="10,0,10,10">
    <Label Content="Аудиозаписи анкет" FontSize="18" FontFamily="Elephant"
HorizontalContentAlignment="Center" Margin="0,0,0,5" />
    <DataGrid x:Name="Records" Loaded="Rec_grid_load"
          CanUserDeleteRows="False" CanUserAddRows="False" FontFamily="Consolas"
FontSize="18" IsReadOnly="True"/>
    <StackPanel Orientation="Horizontal" HorizontalAlignment="Center" >
      <RadioButton Name ="Choice_radio" Content="Проверить выделенные"
IsChecked="True" FontFamily="Consolas" FontSize="15" GroupName="First"
Margin="10,20,10,10" Checked="Date_checked"/>
      <RadioButton Name="Date_radio" Content="Выбрать дату..." FontFamily="Consolas"
FontSize="15" GroupName="First" Margin="10,20,10,10" Checked="Date_checked"/>
      <DatePicker Name ="Date" VerticalContentAlignment="Center" FontFamily="Consolas"
FontSize="15" Margin="0,15,10,10" IsEnabled="False"/>
      <Button Margin="10,17,10,12" FontFamily="Consolas" FontSize="15"
Content="Выполнить" Click="Transc_but_click" Padding="3"/>
    </StackPanel>
    <TextBlock Name ="Error_message2" Foreground="#FFFF6600" TextAlignment="Center"
HorizontalAlignment="Center"
          FontFamily="Elephant" FontSize="15" Margin="0,-20, 0, 0"/>
  </StackPanel>
```

</Window>

Данные о проведенных анкетах отображаются в виде таблицы с использованием элемента DataGrid. Как видно из листинга XML-файла, структура элемента DataGrid не задается при инициализации. Она задается программно в обработчике события загрузки данного элемента.

Рассмотрим принцип вывода данных при помощи элемента DataGrid.

Для того, чтобы программно отобразить данные с использованием элемента Datagrid нужно создать объект класса List и приравнять свойство ItemSource объекта Datagrid к созданному списку. Тогда каждый столбец таблицы будет представлять соответствующий элемент в списке, а количество строк будет совпадать с размером списка.

Поговорим о создании объекта List<>. Так как данные, которые мы хотим вывести в окно приложения, могут иметь разные типы, то разумно создать список на основе пользовательского класса, типы данных полей которого будут соответственно совпадать с типом данных необходимой для вывода информации.

Заполнять созданный список будем в цикле: каждая строка таблицы базы данных будет соответствовать одному элементу списка, содержащему объект пользовательского класса, хранящий необходимые данные из базы данных.

В структуре окна проверки качества анкет пользовательский класс называется Records, его поля хранят информацию о названии анкеты, о пути до анкеты, о дате проведения анкеты, о фамилии проверяющего и о названии региона проверяющего. Представление таблицы Records в MS Access отображено на рисунке 14.

Records

id_record	Path	text_anket	date_anket	Id_emloyee
1	C:\Users\Vlad\Dropbox\diplom\au	1 вопрос - 3 (Затрудняюсь ответить	18.05.2020 10:12:42	119
2	C:\Users\Vlad\Dropbox\diplom\au	1 вопрос - 2 (С беспокойством и тре	19.04.2020 16:42:15	120
3	C:\Users\Vlad\Dropbox\diplom\au	1 вопрос - 2 (Недостаточные) | 2 во	18.05.2020 19:01:22	121
(№)				0

Рисунок 14 – Представление таблицы Records в MS Access

Значение false свойств CanUserDeleteRows и CanUserAddRows элемента DataGrid запрещает пользовательское удаление или добавление строк.

Элементы RadioButton позволяют выбрать тип проверки: проверка выбранных в таблице аудиозаписей анкет или проверка аудиозаписей анкет за выбранную дату. Два этих элемента объединены в группу, это означает что, в одно и то же время, активным может быть только один из двух элементов.

Элемент DataPicker позволяет пользователю выбрать дату, за которую нужно осуществить проверку качества анкет.

Клик по кнопке в окне проверки качества анкет запускает ряд процессов. В первую очередь происходит проверка на возможные несоответствия, такие как:

– не выбрана ни одна аудиозапись анкеты при активированном элементе RadioButton с текстовой меткой «Проверить выделенные» (см. рисунок 15);

– в элементе DataPicker не выбрана дата, при активированном элементе RadioButton с текстовой меткой «Выбрать дату…» (см. рисунок 16).

Затем происходит открытие диалогового окна, заданного объектом класса SaveFileDialog, в котором пользователю предлагается выбрать место сохранения результатов проверки качества анкеты.

Дальнейшее развитие событий зависит от того, какой элемент RadioButton активирован:

– Если активирован элемент RadioButton с текстовой меткой «Проверить выделенные», то в цикле для всех выделенных в DataGrid элементе строк поочередно запускается процесс проверки качества. Выделенные строки хранятся в свойстве SelectedItems элемента Datagrid.

– Если активирован элемент RadioButton с текстовой меткой «Выбрать дату…», то в списке, задающем отображение данных в элементе DataGrid, происходит поиск элементов значение поля типа DateTime которых совпадает со значением элемента DataPicker. Затем для этих элементов поочередно запускается процесс проверки качества.

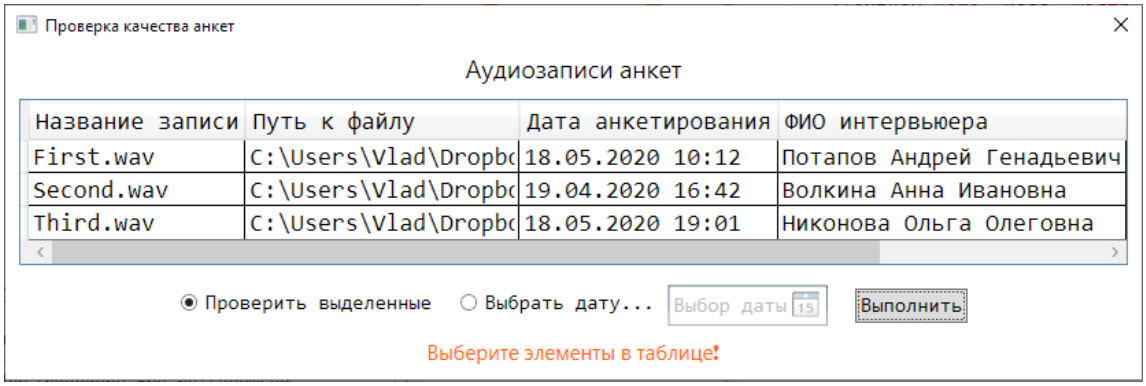


Рисунок 15 – Ошибка, возникающая если не выбрана ни одна аудиозапись

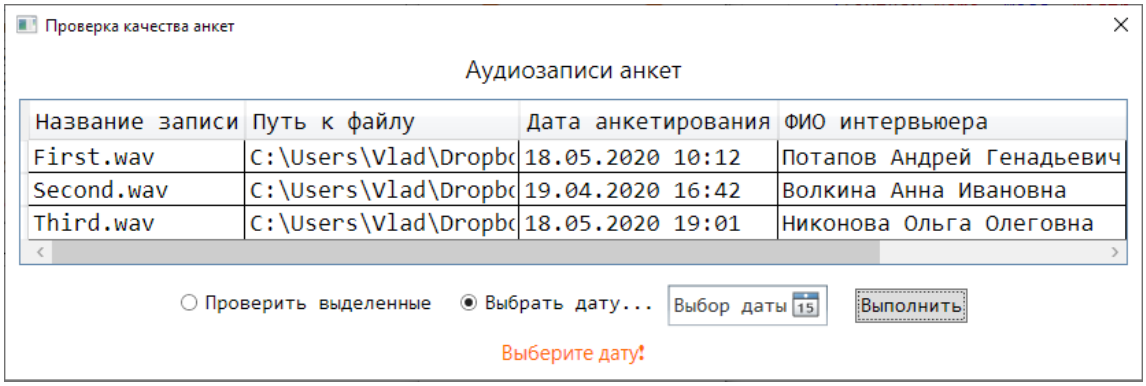


Рисунок 16 – Ошибка, возникающая если не выбрана дата проверки

Процесс проверки качества анкеты состоит из следующих этапов:

1. С использованием методов классов пространства имен System.Net отправляем Post запрос на сервер Google, содержащий аудиозапись в формате массива байтов, и принимаем от сервера текстовое представление аудиофайла.
2. Путем синтаксического анализа с использованием методов классов String и Regex на основе фразы «ответ под цифрой …» вычленяем ответы респондента на вопросы.
3. Записываем в массив строк числовое представление указанных респондентом вариантов ответов.
4. Получаем из базы данных текст анкеты и записываем его в массив строк по аналогии с транскрибированным вариантом.
5. Записываем транскрибированный вариант анкеты в выбранный пользователем с помощью объекта класса SaveFileDialog файл.

6. Записываем текст анкеты в файл по аналогии с предыдущим пунктом.

7. Проверяем на совпадение транскрибированный вариант ответов респондента с текстом анкеты и записываем результат проверки в файл.

8. В базу данных в таблицу Check_quality вносим данные о результате проверки каждой анкеты.

9. Открываем сформированный текстовый файл.

Смоделируем проверку качества анкет за 18.05.2020.

Таблица базы данных Check_Quality не содержит ни одной записи. Выберем в элементе DataPicker указанную дату и нажмем на кнопку «Выполнить». В открывшимся диалоговом окне выберем файл с названием «all.txt» на рабочем столе и нажмем сохранить. После нескольких секунд ожидания в текстовом блоке, предназначенном для коммуникации с пользователем появилась надпись: «Процесс проверки завершен!» (см. рисунок 17). И открылся текстовый файл, содержащий результаты проверки качества анкет за 18.05.2020 (см. рисунок 18). Таблица базы данных Check_Quality так же изменилась, в ней появились две записи о результатах проверки качества анкет (см. рисунок 19).

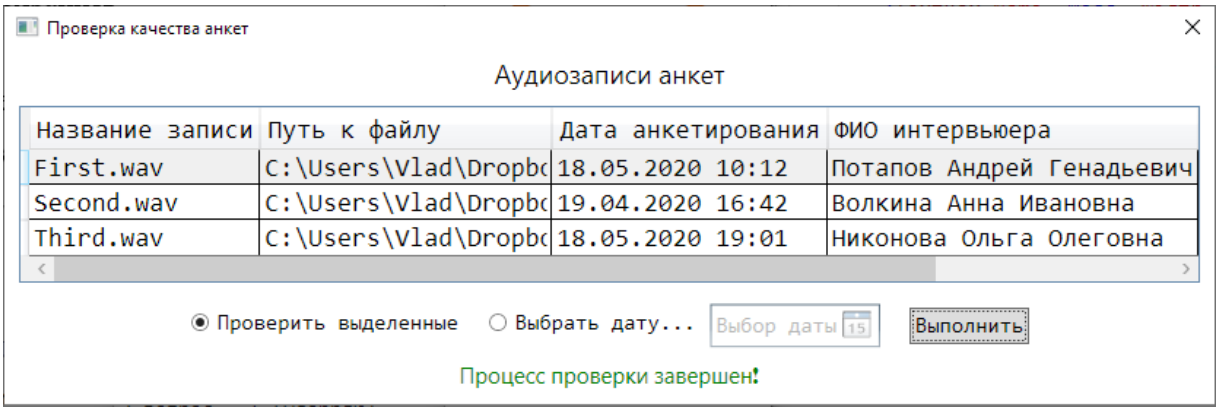


Рисунок 17 – Вид окна проверки качества анкет после завершения проверки

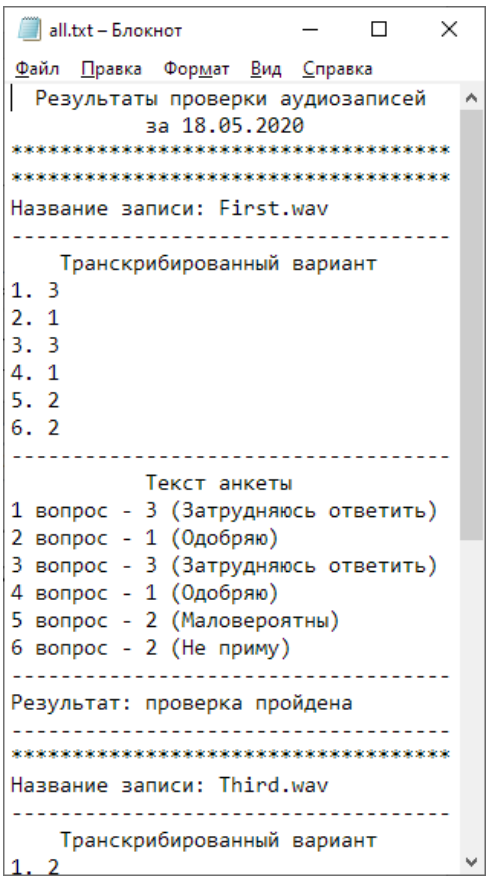


Рисунок 18 – Текстовый файл с результатами проверки качества анкет за 18.05.2020

Check_Quality

Id_checking	Dateofchecking	Id_record	Result	Comment	Id_sotr
11	15.06.2020 20:29:51	1	☑	-	2
12	15.06.2020 20:30:28	3	☑	-	2

Рисунок 19 – Таблица Check_Quality после проверки качества анкет за 18.05.2020

В случае, если анкета не пройдет проверку качества, то в открывшимся файле будет соответствующая пометка и значение столбца Result таблицы Check_Quality будет равно false (см. рисунок 20 и рисунок 21).

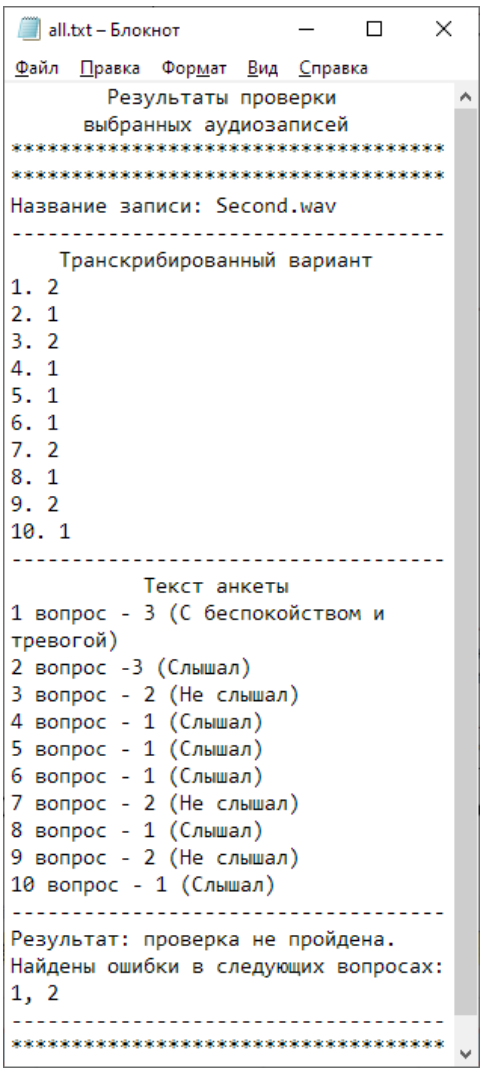


Рисунок 20 – Результат проверки качества некорректно проведенной анкеты

Check_Quality

Id_checking	Dateofchecking	Id_record	Result	Comment	Id_sotr
11	15.06.2020 20:29:51	1	☑	-	2
12	15.06.2020 20:30:28	3	☑	-	2
15	15.06.2020 20:53:11	2	☐	Результат: пр(	2

Рисунок 21 – Запись в базе данных о некорректно проведенной анкете

Если возникнет ситуация, когда одну и ту же анкету будут проверять несколько раз, то в базе данных будет храниться информация только о самой первой проверке качества анкеты.

3.4 Окно истории проверок качества анкет

Окно истории проверок качества анкет задает класс History_window, оно по аналогии с главным окном программы разным типам пользователей предоставляет различный функционал.

Код XML-файла окна истории проверок:

```
<Window x:Class="Check_Quality.History_window"
…
    Title="История" SizeToContent="Height" Width="845" WindowStartupLocation="CenterScreen" ResizeMode="NoResize" >
  <StackPanel Margin="10,0,10,10">
    <Label Content="История проверок" FontSize="18" FontFamily="Elephant" HorizontalContentAlignment="Center" Margin="0,0,0,5" />
    <DataGrid x:Name="History" Loaded="History_Loaded"
          CanUserDeleteRows="False" CanUserAddRows="False" FontFamily="Consolas" FontSize="18" IsReadOnly="True"/>
    <StackPanel Orientation="Horizontal" HorizontalAlignment="Center" >
      <RadioButton Name ="Choice_radio" Content="Выгрузить выделенные" IsChecked="True" FontFamily="Consolas" FontSize="15" GroupName="First" Margin="10,21,10,10" Checked="Date_checked"/>
      <RadioButton Name="Date_radio" Content="Выбрать дату анкетирования..." FontFamily="Consolas" FontSize="15" GroupName="First" Margin="10,21,15,10" Checked="Date_checked" />
      <DatePicker Name ="Date" VerticalContentAlignment="Center" FontFamily="Consolas" FontSize="15" Margin="0,15,10,10" IsEnabled="False"/>
      <Button Margin="10,17,10,12" FontFamily="Consolas" FontSize="15" Content="Выгрузить" Click="Button_Click" Padding="2"/>
      <Button Name="Del_but" Margin="-80,17,10,12" Visibility="Hidden" FontFamily="Consolas" FontSize="15" Content="Удалить" Padding="2" Click="Del_but_click"/>
    </StackPanel>
    <TextBlock Name ="Error_message3" Foreground="#FFFF6600" TextAlignment="Center" HorizontalAlignment="Center"
          FontFamily="Elephant" FontSize="15" Margin="0,-20, 0, 0"/>
  </StackPanel>
```

</Window>

Процесс вывода данных с использованием элемента DataGrid описан в пункте 3.3.

Пользовательский класс, использующийся для вывода данных в элемент DataGrid класса History_window, называется History. Он хранит следующую информацию: название аудиозаписи, дату проверки, дату проведения анкеты, ФИО проверяющего, результат проверки и комментарий к результату проверки.

Механизм изменения свойств элементов окна в зависимости от типа пользователя описан в пункте 3.2.

Вид окна истории проверок для администратора отражен на рисунке 22.

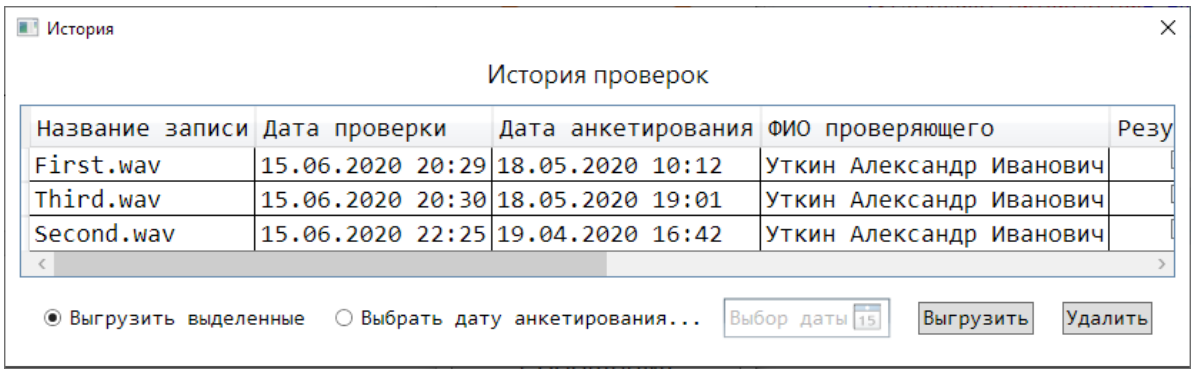


Рисунок 22 – Вид окна истории проверок для администратора

Администратору доступна как выгрузка, так и удаление информации обо всех проведенных проверках качества анкет.

Процесс удаления результатов проверки запускается нажатием на кнопку «Удалить». В первую очередь проверяется выделен ли хотя бы один элемент таблицы, в случае если ни один элемент не выбран в коммуникационный текстовый блок выводится сообщение об ошибке (см. рисунок 23). Далее для каждой выделенной строки в цикле выполняется соответствующая SQL инструкция Delete.

Вид окна истории проверок для проверяющего отображен на рисунке 24.

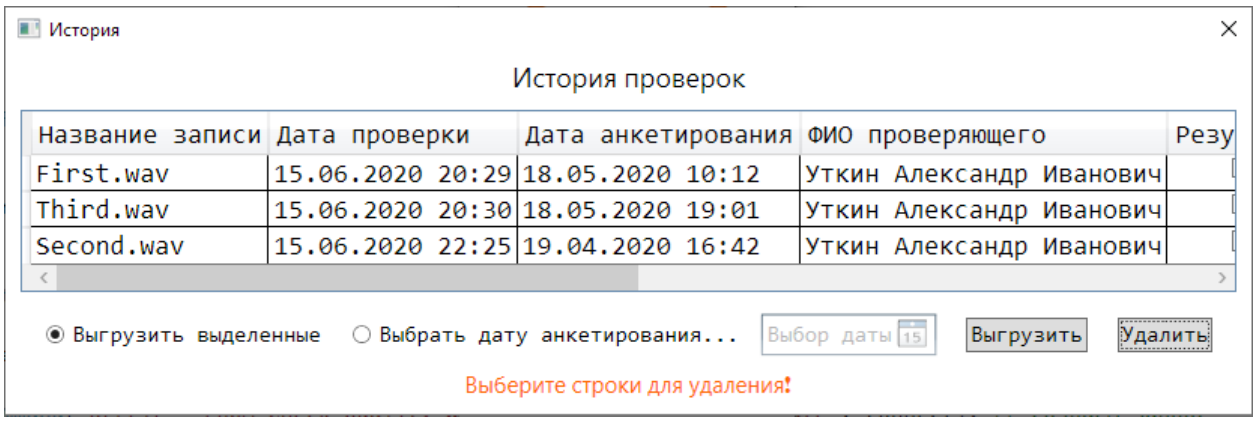


Рисунок 23 – Ошибка при удалении элементов

Проверяющему доступен тот же функционал, что и администратору, за исключением удаления результатов проверки.

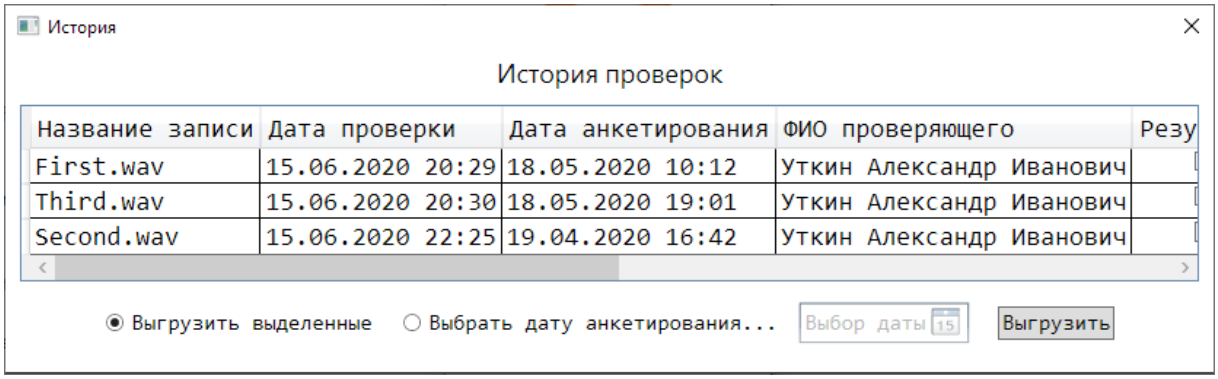


Рисунок 24 – Вид окна истории проверок для проверяющего

Интервьюеру доступна выгрузка результатов только проведенных им анкет. Вид окна проверки качества анкет для интервьюера отображен на рисунке 25.

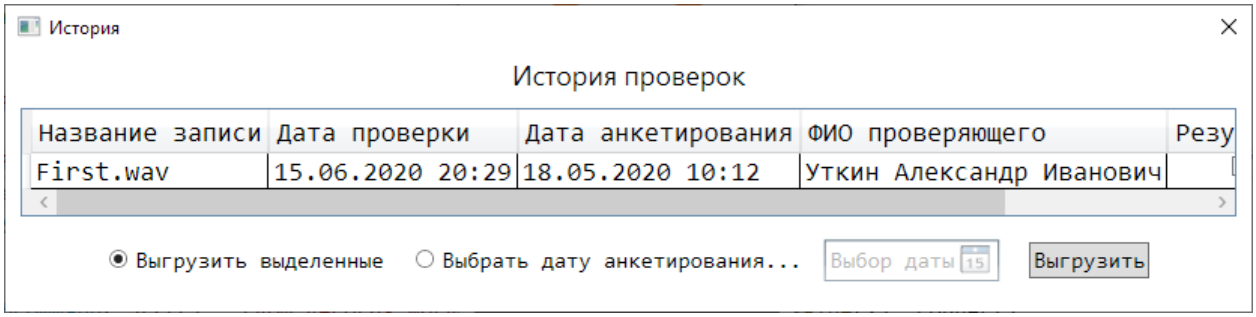


Рисунок 25 – Вид окна истории проверок для интервьюера

Выбор варианта выгрузки результатов проверки реализован при помощи элементов RadioButton и элемента DataPicker. Работа с данной совокупностью элементов описана в пункте 3.3.

Смоделируем ситуацию выгрузки результатов проверок качества анкет, проведенных нами в пункте 3.3.

После того, как администратор выберет необходимые строки в таблице и нажмет кнопу «Выгрузить», откроется выбранный им в диалоговом окне текстовый файл, содержащий результаты проверки качества (см. рисунок 26), а значение текстового блока внизу окна изменится на «Выгрузка завершена».

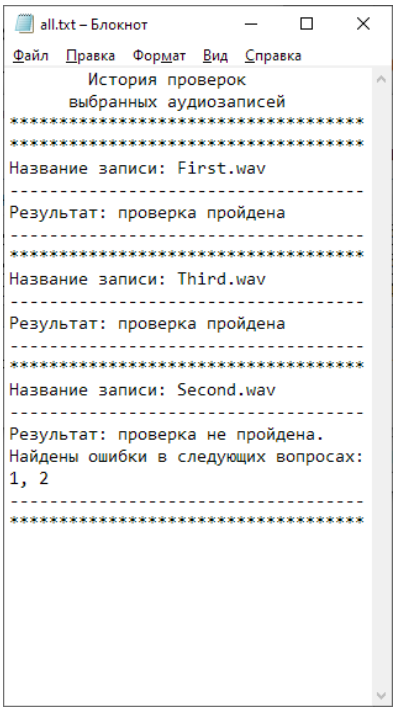


Рисунок 26 – Результаты выгрузки

3.5 Окно панели администратора

Окно панели администратора задает класс Admin_Panel, оно доступно исключительно администратору. Вид окна панели администратора отображен на рисунке 27.

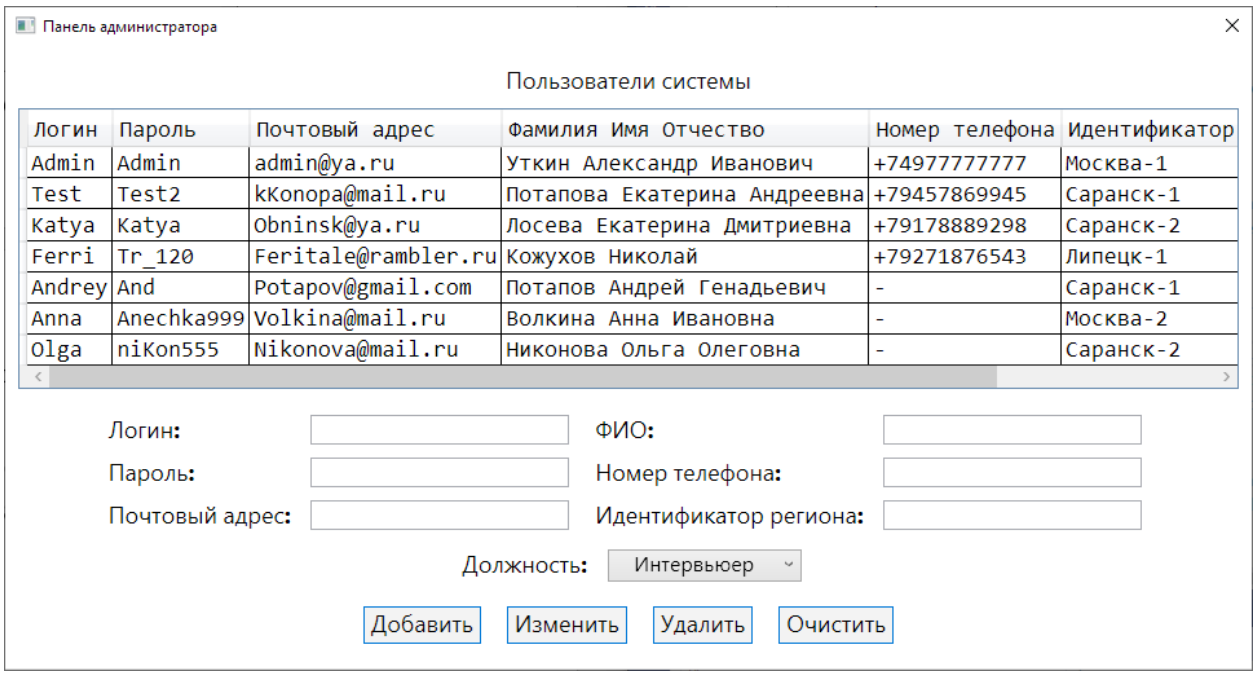


Рисунок 27 – Вид панели администратора

Код XML-файла окна панели администратора:

```
<Window x:Class="Check_Quality.Admin_Panel"
…
    Title="Панель администратора" x:Name="Adm_window" ResizeMode="NoResize" SizeToContent="WidthAndHeight" MaxWidth="980" WindowStartupLocation="CenterScreen" Height="308">
  <StackPanel Margin="10,10,10,10">
    <Label Margin="0,0,0,5" Name="Canvas_1" Content="Пользователи системы" FontSize="18" FontFamily="Elephant" HorizontalContentAlignment="Center"/>
    <DataGrid x:Name="Database" MaxHeight="350" Loaded="datagrid_load" CanUserDeleteRows="False" CanUserAddRows="False" FontFamily="Consolas" FontSize="18" IsReadOnly="True" />
    <StackPanel Orientation="Horizontal" Margin="0,15,0,0" HorizontalAlignment="Center">
      <StackPanel>
        <Label Content="Логин:" FontSize="18" FontFamily="Elephant" />
        <Label Content="Пароль:" FontSize="18" FontFamily="Elephant" />
        <Label Content="Почтовый адрес:" FontSize="18" FontFamily="Elephant" />
      </StackPanel>
      <StackPanel x:Name="TB_Own" Margin="10,5,0,0">
```

```
            <TextBox Name ="Grid_Log" Text="" HorizontalAlignment="Left" VerticalAlignment="Center" FontSize="18" FontFamily="Consolas" Width="200" Margin="0,0,0,5" LostFocus="Log_Lost_F" />
            <TextBox Name ="Grid_pass" Text="" HorizontalAlignment="Left" VerticalAlignment="Center" FontSize="18" FontFamily="Consolas" Width="200" Margin="0,5" LostFocus="Log_Lost_F" />
            <TextBox Name ="Grid_Email" Text="" HorizontalAlignment="Left" VerticalAlignment="Center" FontSize="18" FontFamily="Consolas" Width="200" Margin="0,5" />
        </StackPanel>
        <StackPanel Margin="15,0,0,0">
            <Label Content="ФИО:" FontSize="18" FontFamily="Elephant" />
            <Label Content="Номер телефона:" FontSize="18" FontFamily="Elephant" />
            <Label Content="Идентификатор региона:" FontSize="18" FontFamily="Elephant" />
        </StackPanel>
        <StackPanel x:Name="TB1_Own" Margin="10,5,10,0">
            <TextBox Name ="Grid_FIO" Text="" HorizontalAlignment="Left" VerticalAlignment="Center" FontSize="18" FontFamily="Consolas" Width="200" Margin="0,0,0,5" />
            <TextBox Name ="Grid_Num" Text="" HorizontalAlignment="Left" VerticalAlignment="Center" FontSize="18" FontFamily="Consolas" Width="200" Margin="0,5" />
            <TextBox Name ="Grid_id" Text="" HorizontalAlignment="Left" VerticalAlignment="Center" FontSize="18" FontFamily="Consolas" Width="200" Margin="0,5" />
        </StackPanel>
    </StackPanel>
    <StackPanel Orientation="Horizontal" Margin="0,5,0,0" HorizontalAlignment="Center">
        <Label Content="Должность:" FontSize="18" FontFamily="Elephant" Margin="0,0,10,0" />
        <ComboBox Name="Combo" Height="25" Width="150" IsReadOnly="True" HorizontalContentAlignment="Center">
            <TextBlock Name="Combo_default" Text="Интервьюер" FontSize="16" FontFamily="Elefphant" />
            <TextBlock Text="Проверяющий" FontSize="16" FontFamily="Elefphant" />
            <TextBlock Text="Администратор" FontSize="16" FontFamily="Elefphant" />
        </ComboBox>
    </StackPanel>
```

```
        <StackPanel Margin="0,5,0,0" Orientation="Horizontal" HorizontalAlignment="Center">
            <Button Content="Добавить" Name ="Add_but" FontFamily="Elephant" FontSize="18" IsDefault="True" Margin="10"
                Background="#FFF3F3F3" Padding="5,2" Click="Add_click" />
            <Button Content="Изменить" Padding="5,2" Name ="Change_but" FontFamily="Elephant" FontSize="18" IsDefault="True" Margin="10"
                Background="#FFF3F3F3" Click="Change_but_Click" />
            <Button Content="Удалить" Padding="5,2" FontFamily="Elephant" FontSize="18" IsDefault="True" Margin="10"
                Background="#FFF3F3F3" Click="Del_click" />
            <Button Content="Очистить" Padding="5,2" FontFamily="Elephant" FontSize="18" IsDefault="True" Margin="10"
                Background="#FFF3F3F3" Click="Cleare_click" />
        </StackPanel>
        <TextBlock Name ="Error_message1" Margin ="0,-20,0,0" Foreground="#FFFF6600" TextAlignment="Center" HorizontalAlignment="Center" FontFamily="Elephant" FontSize="15"/>
    </StackPanel>
</Window>
```

Элемент DataGrid содержит информацию о пользователях системы, а именно: логин, пароль, адрес электронной почты, ФИО, номер телефона, идентификатор региона и должность (тип пользователя). Работа с элементом DataGrid подробно описана в пункте 3.3.

Поговорим о функционале панели администратора. Администратор может добавлять, редактировать и удалять пользователей.

Совокупность элементов TextBlock и элемента ComboBox предназначена для ввода данных о новом пользователе и редактирования данных уже работающего сотрудника.

При клике по кнопке «Добавить» проверяются следующие условия: введен ли логин и пароль, существует ли введенный логин в базе данных, корректно ли задан идентификатор региона, в случае возникновения несоответствий в текстовый блок внизу окна выводится информация об ошибке. Пример сообщения об ошибке показан на рисунке 28.

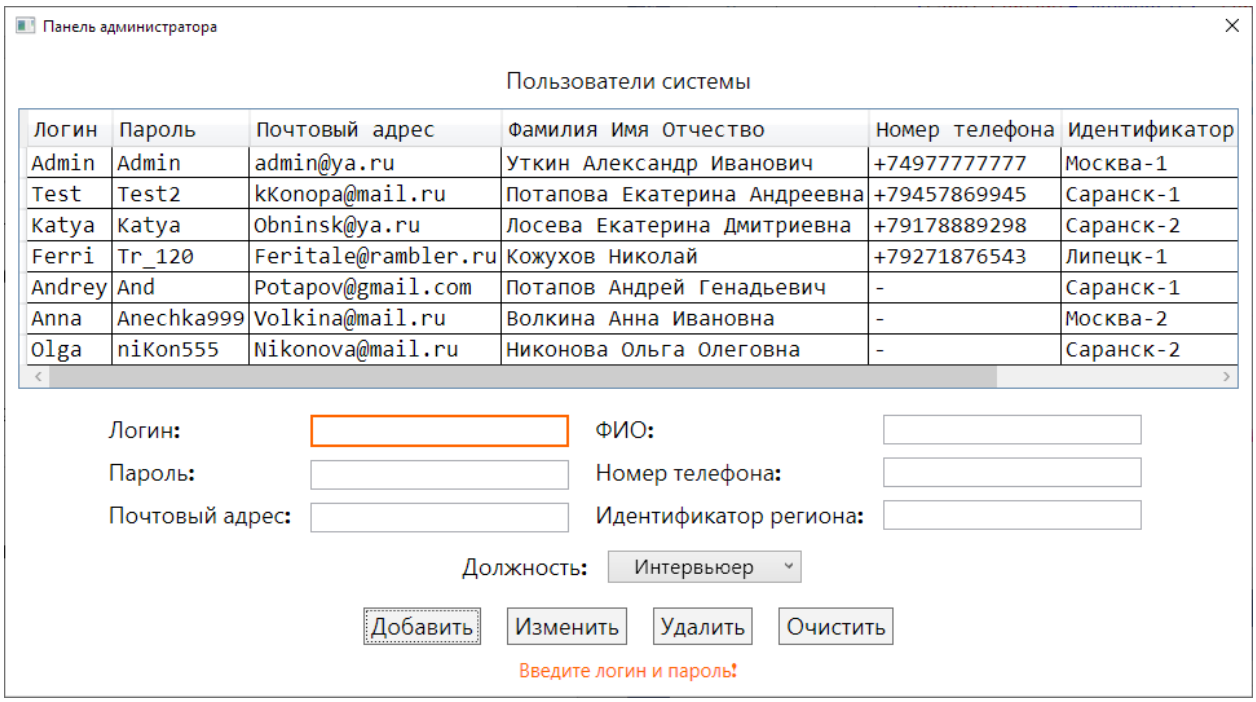


Рисунок 28 – Ошибка в окне панели администратора

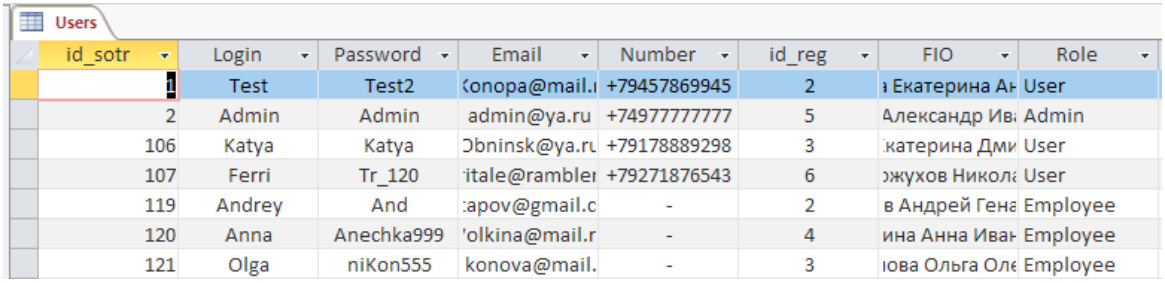
Если ошибок не возникло, то данные о новом сотруднике заносятся в базу данных, а поля ввода очищаются.

При клике по кнопке «Изменить» происходит проверка, выбран ли сотрудник в таблице, затем элементы ввода на экране принимают значения соответствующих столбцов выбранной строки, а название кнопки меняется на «Сохранить». При повторном нажатии на данную кнопку, сделанные изменения вносятся в базу данных, а название кнопки меняется на начальное.

Клик по кнопке «Удалить» для каждой выбранной строки выполняет соответствующие SQL инструкции Delete, с предварительной проверкой выбраны ли элементы.

Клик по кнопке «Очистить» очищает все поля ввода.

Смоделируем ситуацию удаления данных о сотруднике. Вид таблицы User базы данных отображен на рисунке 29.



Users

id_sotr	Login	Password	Email	Number	id_reg	FIO	Role
1	Test	Test2	konopa@mail.r	+79457869945	2	а Екатерина Ан	User
2	Admin	Admin	admin@ya.ru	+74977777777	5	Александр Ива	Admin
106	Katya	Katya	Obninsk@ya.ru	+79178889298	3	катерина Дми	User
107	Ferri	Tr_120	ritale@rambler	+79271876543	6	жухов Никола	User
119	Andrey	And	tapov@gmail.c	-	2	в Андрей Гена	Employee
120	Anna	Anechka999	volkina@mail.r	-	4	ина Анна Иван	Employee
121	Olga	niKon555	konova@mail.	-	3	ова Ольга Оле	Employee

Рисунок 29 – Таблица Users

Выберем в элементе DataGrid пользователя с логином Test и нажмем на кнопку «Удалить». Строка данных о пользователе пропадает из таблицы, а в текстовом блоке внизу экрана появляется соответствующая надпись (см рисунок 30).

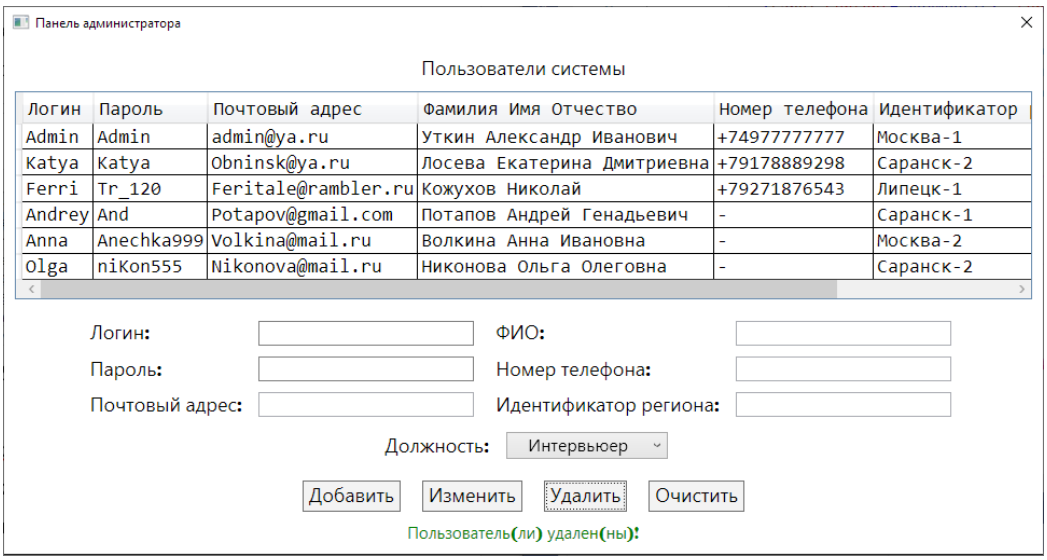
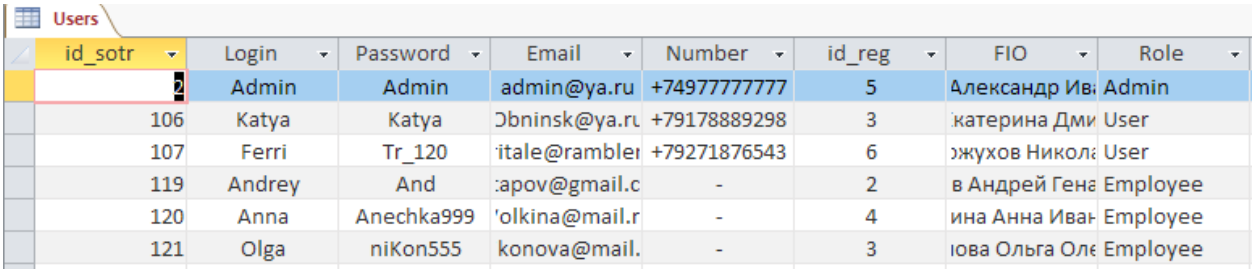


Рисунок 30 – Сообщение об удалении пользователя

Таблица Users базы данных так же изменяется (см. рисунок 31).



Users

id_sotr	Login	Password	Email	Number	id_reg	FIO	Role
2	Admin	Admin	admin@ya.ru	+74977777777	5	Александр Ива	Admin
106	Katya	Katya	Obninsk@ya.ru	+79178889298	3	катерина Дми	User
107	Ferri	Tr_120	ritale@rambler	+79271876543	6	жухов Никола	User
119	Andrey	And	tapov@gmail.c	-	2	в Андрей Гена	Employee
120	Anna	Anechka999	volkina@mail.r	-	4	ина Анна Иван	Employee
121	Olga	niKon555	konova@mail.	-	3	ова Ольга Оле	Employee

Рисунок 31 – Измененная таблица Users

3.6 Окно обмена сообщениями

Окно обмена сообщениями задает класс Messages, оно доступно всем пользователям системы (см. рисунок 32).

Рисунок 32 – Окно обмена сообщениями

XML-код окна обмена сообщениями:

```
<Window x:Class="Check_Quality.Messages"
…
    Title="Сообщения" SizeToContent="WidthAndHeight"
WindowStartupLocation="CenterScreen" ResizeMode="NoResize">
  <StackPanel Margin="10,0,10,10">
    <StackPanel Margin="0,0,0,5" Orientation="Horizontal" HorizontalAlignment="Center">
      <StackPanel Margin="0,0,20,0">
        <Label Content="Принятые" FontFamily="Elephant" FontSize="18"
HorizontalAlignment="Center" />
        <DataGrid Name="Inc" CanUserDeleteRows="False" CanUserAddRows="False"
FontFamily="Consolas" FontSize="18" IsReadOnly="True" Loaded="From_mess_load"/>
        <TextBlock Name ="Inc_empty" Visibility="Hidden" Text="Не принято ни одного
сообщения" TextAlignment="Center" HorizontalAlignment="Center"
```

```
            FontFamily="Elephant" FontSize="15" Margin="0,-20, 0, 0"/>
        <Label Content="Отправленные" Margin="0,15,0,0" FontFamily="Elephant" FontSize="18" HorizontalAlignment="Center" />
        <DataGrid Name="Send" CanUserDeleteRows="False" CanUserAddRows="False" FontFamily="Consolas" FontSize="18" IsReadOnly="True" Loaded="To_mess_load" />
        <TextBlock Name ="Send_empty" Visibility="Hidden" Text="Не отправлено ни одного сообщения" TextAlignment="Center" HorizontalAlignment="Center"
            FontFamily="Elephant" FontSize="15" Margin="0,-20, 0, 0"/>
      </StackPanel>
    </StackPanel>
    <StackPanel Orientation="Horizontal" HorizontalAlignment="Center">
      <Button Content="Новое сообщение" Name ="Add_but" FontFamily="Elephant" FontSize="18" IsDefault="True" Margin="10"
          Background="#FFF3F3F3" Padding="5,2" Click="New_mess" />
      <RadioButton Name ="Inc_radio" Content="Принятые" IsChecked="True" FontFamily="Consolas" FontSize="15" GroupName="First" Margin="10,21,10,10" Checked="Hide" />
      <RadioButton Name="Send_radio" Content="Отправленные" FontFamily="Consolas" FontSize="15" GroupName="First" Margin="10,21,15,10" Checked="Hide" />
      <Button Content="Просмотреть" Padding="5,2" Name ="Change_but" FontFamily="Elephant" FontSize="18" IsDefault="True" Margin="10"
          Background="#FFF3F3F3" Click="Check_mess_click" />
      <Button Content="Удалить" Padding="5,2" FontFamily="Elephant" FontSize="18" IsDefault="True" Margin="10"
          Background="#FFF3F3F3" Click="Del_click" />
    </StackPanel>
    <TextBlock Name ="Error_message4" Foreground="#FFFF6600" TextAlignment="Center" HorizontalAlignment="Center"
          FontFamily="Elephant" FontSize="15" Margin="0,-20, 0, 0"/>
  </StackPanel>
</Window>
```

Как видно из листинга XML-файла, вывод полученных и отправленных сообщений реализован с использованием элементов DataGrid. Процесс заполнения элемента DataGrid данными описан в пункте 3.3. Пользовательский

класс, использующийся для вывода данных, у двух этих элементов совпадает. Он содержит следующую информацию: идентификатор сообщения, ФИО (в случае таблицы полученных сообщений храниться ФИО адресата, в случае таблицы принятых сообщений - ФИО адресанта), дату отправки сообщения, идентификатор региона (таблица принятых сообщений содержит идентификатор региона отправителя, а таблица отправленных – идентификатор региона получателя) и статус о прочтении сообщения.

В обработчиках события клика по кнопкам «Просмотреть» и «Удалить» первостепенно проверяется, какой из элементов RadioButton активен и в соответствии с этим происходит взаимодействие с одной из двух таблиц в окне приложения.

Клик по кнопке «Просмотреть», если в соответствующей таблице выбрано сообщение, открывает окно, которое задает класс MessBox (см. рисунок 33), его заголовок изменяется в соответствии с типом сообщения. Данное окно является пользовательским аналогом объекта класса MessageBox, и создано по причине того, что MessageBox не поддерживает изменение размера и типа шрифта отображаемого сообщения.

Код XML-файла окна, которое задаётся классом MessBox:

```
<Window x:Class="Check_Quality.MessBox"
…
    Title="Сообщения" SizeToContent="WidthAndHeight"
WindowStartupLocation="CenterScreen" ResizeMode="NoResize">
  <StackPanel Margin="10">
    <TextBox Name="txt_message" IsReadOnly="True" Width="350" Height="100" Padding="3"
FontFamily="Consolas" FontSize="15" TextWrapping="Wrap" VerticalScrollBarVisibility="Auto"
/>
    <Button Content="Ок" Margin="135,10,135,0" Click="Close_click"/>
  </StackPanel>
</Window>
```

Окно MesBox используется для вывода текста сообщения, после его закрытия отображаемое сообщение получает статус «Прочитано».

Если при нажатии на кнопку «Прочитать» в соответствующем элементе DataGrid не выбрана ни одна строка, то внизу окна появится текст ошибки (см. рисунок 34).

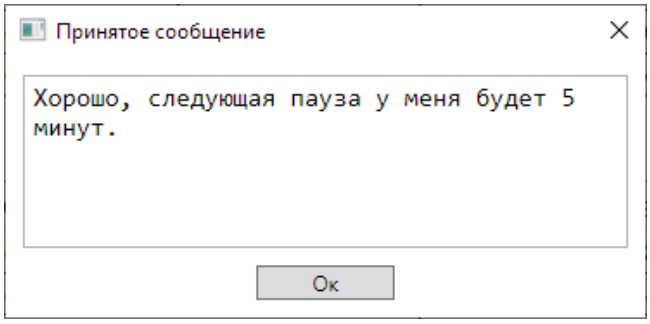


Рисунок 33 – Окно MessBox

Рисунок 34 – Ошибка в окне обмена сообщениями

Клик по кнопке удалить, запускает механизм удаления элементов из DataGrid, описанный в пункте 3.4.

Клик по кнопке «Новое сообщение» открывает окно, которое задет класс New_message (см. рисунок 35). В данном окне пользователю предлагают выбрать адресата и написать текст сообщения. Выбор адресата происходит с

использованием элемента ComboBox, структура которого задается программно в обработчике события загрузки окна и включает пары типа (Логин - ФИО).

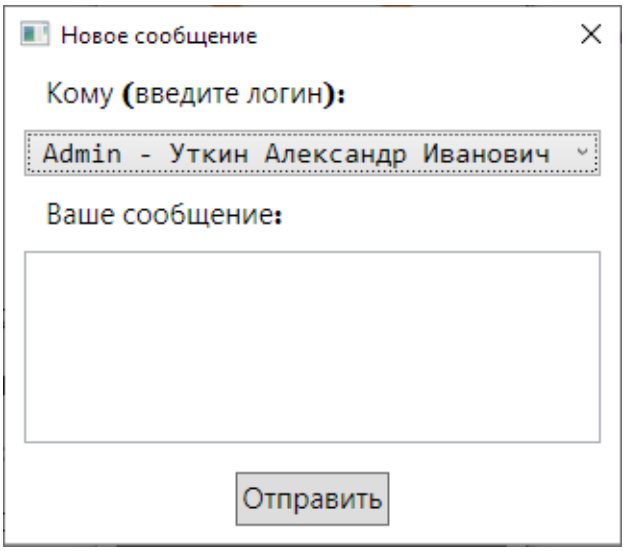


Рисунок 35 – Окно нового сообщения

Код XML-файла окна, которое задаётся классом New_message:

```
<Window x:Class="Check_Quality.New_message"
…
    Title="Новое сообщение" Name="New" SizeToContent="WidthAndHeight" WindowStartupLocation="CenterScreen" ResizeMode="NoResize" Loaded="New_message_loaded">
  <StackPanel>
    <Label Content="  Кому (введите логин):" Margin="5,0,5,0" FontFamily="Elephant" FontSize="15" />
    <ComboBox Name="Login" HorizontalContentAlignment="Center" Margin="10,5" Padding="3" FontFamily="Consolas" FontSize="15" IsReadOnly="True" MaxDropDownHeight="100"/>
    <Label Content="  Ваше сообщение:" Margin="5,0,5,0" FontFamily="Elephant" FontSize="15" />
    <TextBox Name="Mess" Width="300" Height="100" Margin="10,5" Padding="3" FontFamily="Consolas" FontSize="15" TextWrapping="Wrap" VerticalScrollBarVisibility="Auto" />
    <TextBlock Name ="Error_message5" Foreground="#FFFF6600" TextAlignment="Center" HorizontalAlignment="Center"
```

```
            FontFamily="Elephant" FontSize="15" Margin="0, -20, 0, 0"/>
        <Button Content="Отправить" HorizontalAlignment="Center" Margin="0,10,0,10"
FontFamily="Elephant" FontSize="15" Padding="3" Click="Send_click" />
    </StackPanel>
</Window>
```

Клик по кнопке «Отправить», в случае если сообщение написано, сохранит его в системе и обновит информацию в таблице, содержащей отправленные сообщения. Если сообщение не содержит ни одного символа, то в соответствующий текстовый блок внизу окна выведется информация об ошибке (см. рисунок 36).

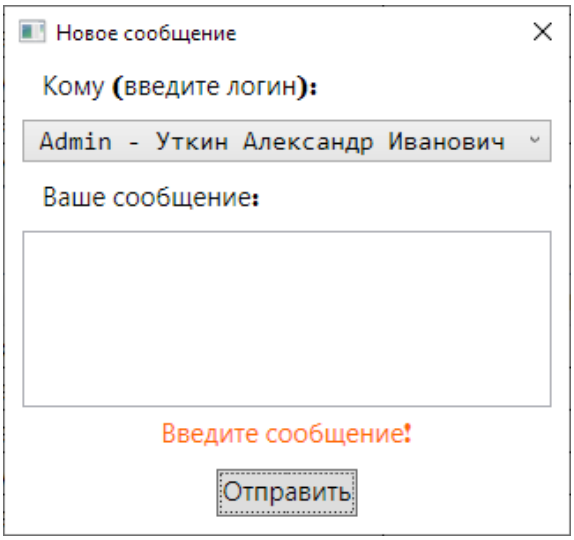


Рисунок 36 – Сообщение об ошибке

ЗАКЛЮЧЕНИЕ

Выпускная квалификационная работа посвящена разработке программного модуля для автоматизации работы отдела контроля качества call-центра.

По факту разработки были достигнуты следующие результаты:

– проведен анализ предметной области разработки программного модуля для автоматизации работы отдела контроля качества call-центра;

– разработана архитектура программного модуля;

– программный модуль полностью реализован.

При разработке была проанализирована совокупная работа существующих систем автоматизации процессов, тесно связанных с деятельностью call-центра, построены UML – диаграммы и схемы интеграции.

Внедрение новых решений по автоматизации работы отдела контроля качества call-центра облегчает и ускоряет процесс проверки качества проведенных при социологическом исследовании анкет, а также уменьшает затраты организации на оплату работы сотрудников данного отдела.

Разработанный программный модуль имеет перспективы интеграции при соответствующей доработке под деятельность конкретной организации.

Наличие опций администрирования сотрудников и обмена сообщениями между пользователями приложения позволяет сузить круг используемого организацией программного обеспечения и отказаться от использования локальных мессенджеров.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Актуализация баз данных – КоллЦентр24 [Электронный ресурс] – Режим доступа:
2. Классификация вопросов в социологической анкете – StudFiles [Электронный ресурс] – Режим доступа: https://studfile.net/preview/4217271/page:7
3. Преимущества СУБД Microsoft Access – FB [Электронный ресурс] – Режим доступа: https://fb.ru/article/289865/ms-access-bazyi-dannyih-ms-access-ms-access
4. Поиск оптимальной системы аудио распознавания речи с закрытым исходным кодом, но имеющими открытые API, для возможности интеграции – Хабр [Эллектронный ресурс] – Режим доступа: https://habr.com/ru/post/231629
5. Accessing Google Speech API / Chrome 11 – mike pultz [Электронный ресурс] – Режим доступа: https://mikepultz.com/2011/03/accessing-google-speech-api-chrome-11
6. Microsoft Speech API – Qwe.wiki [Электронный ресурс] – Режим доступа: https://ru.qwe.wiki/wiki/Microsoft_Speech_API
7. UML – Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/UML
8. Создание диаграмм баз данных – ИНТУИТ [Электронный ресурс] – Режим доступа: https://www.intuit.ru/studies/courses/4081/147/lecture/4061
9. Типа данных в MS Access – Accesshelp [Электронный ресурс] – Режим доступа: https://accesshelp.ru/tipy-dannyh-v-ms-access
10. Абрамян А.В. Разработка пользовательского интерфейса на основе системы Windows Presentation Foundation – Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2017. — 301 с
11. Мэтью Мак-Дональд. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов, 4-е издание = Pro WPF 4.5 in C#

2012: Windows Presentation Foundation in .NET 4.5, 4th edition. — М.: «Вильямс», 2013. — 1024 с. — ISBN 978-5-8459-1854-3.

12. Андерсон, Крис. Основы Windows Presentation Foundation. — СПб.: БХВ-Петербург, 2008. — 432 с. — ISBN 978-5-9775-0265-8.

ПРИЛОЖЕНИЕ А

Описание классов разрабатываемого модуля на языке C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Data.OleDb;
using System.IO;
using System.Net;
using System.Data.Common;
using System.Text.RegularExpressions;
using System.Windows.Forms;
using System.Threading;
using System.Collections.Specialized;
using System.Data;
using System.Diagnostics;

namespace Check_Quality
{
    public partial class Tranc_window : Window
    {
        public Tranc_window()

        {
```

```
            InitializeComponent();
        }

        OleDbConnection connect = new
OleDbConnection(Common_functions.connection);

        private string[] Trancribation(string path)
        {
            WebRequest request =
WebRequest.Create("https://www.google.com/speech-
api/v2/recognize?output=json&lang=ru-RU&key=AIzaSyBOti4mM-
6x9WDnZIjIeyEU21OpBXqWBgw");
            request.Method = "POST";
            byte[] byteArray = File.ReadAllBytes(path);
            request.ContentType = "audio/l16; rate=16000";
//"16000";
            request.ContentLength = byteArray.Length;
            request.GetRequestStream().Write(byteArray, 0,
byteArray.Length);
            HttpWebResponse response =
(HttpWebResponse)request.GetResponse();
            StreamReader reader = new
StreamReader(response.GetResponseStream());
            string text = reader.ReadToEnd();
            reader.Close();
            response.Close();
            int start = text.IndexOf("[{\"");
            int end = text.IndexOf("\",\"");
            text = text.Substring(start, end - start);
            MatchCollection matches =
Common_functions.Regular(text, @"ответ под цифрой (\w+) ", true);
            string[] str_matches = new string[matches.Count];
```

```
            for (int i = 0; i < matches.Count; i++)
            {
                str_matches[i] = matches[i].Value;
                str_matches[i] = str_matches[i].Trim();
                str_matches[i] =
str_matches[i].Substring(str_matches[i].LastIndexOf(" ") + 1,
str_matches[i].Length - (str_matches[i].LastIndexOf(" ") + 1));
                str_matches[i] =
Strtoint_1_15(str_matches[i]).ToString();
            }
            return str_matches;
        }

        private string[] Split_txt_anket(string path)
        {
            if (connect.State != ConnectionState.Open)
                connect.Open(); // открываем базу данных
            OleDbCommand cmd = new OleDbCommand("SELECT * FROM
Records WHERE Path ='" + path + "'", connect); // создаём запрос
            OleDbDataReader reader = cmd.ExecuteReader(); //
получаем данные
            reader.Read();
            string txt_anket = reader.GetString(2);
            reader.Close();
            connect.Close();

File.WriteAllText(@"C:\Users\Vlad\Dropbox\diplom\audio\Checking.tx
t", txt_anket);
            string[] sub_txt_anket = txt_anket.Split(new string[]
{ " | " }, StringSplitOptions.RemoveEmptyEntries);
            return sub_txt_anket;
        }
```

```
        private void Write_transc_txt_in_file(string FilePath,
string Title, string[] str_matches)
        {
            using (StreamWriter sw = new StreamWriter(FilePath,
true))
            {
                int index = 1;
                sw.WriteLine("Название записи: " +
Common_functions.Regular(Title, @"(\w*).\w{3}\z", false) + "\n----
--------------------------------\n" +
                    "    Транскрибированный вариант");
                for (int i = 0; i < str_matches.Length; i++)
                {
                    sw.WriteLine(index + ". " + str_matches[i]);
                    index++;
                }
            }
        }

        private void Write_txt_anket_in_file(string FilePath,
string[] sub_txt_anket)
        {
            using (StreamWriter sw = new StreamWriter(FilePath,
true))
            {
                sw.WriteLine("-----------------------------------
\n" + "           Текст анкеты");
                for (int i = 0; i < sub_txt_anket.Length; i++)
                    sw.WriteLine(sub_txt_anket[i]);
                sw.WriteLine("-----------------------------------
");
            }
        }
```

```
        private void Checking_and_writing_in_database(string
FilePath, string itemPath,string[] sub_txt_anket, string[]
str_matches)
        {
            for (int i = 0; i < sub_txt_anket.Length; i++)
                sub_txt_anket[i] =
sub_txt_anket[i].Substring(sub_txt_anket[i].IndexOf("-") + 2, 1);

//File.WriteAllLines(@"C:\Users\Vlad\Dropbox\diplom\audio\Checking
.txt", sub_txt_anket);
            int index = 0;
            int result = 1;
            string comment = "";
            using (StreamWriter sw = new StreamWriter(FilePath,
true))
            {
                if (str_matches.Length == sub_txt_anket.Length)
                {
                    int[] Checking = new int[str_matches.Length];
                    for (int i = 0; i < str_matches.Length; i++)
                        if (str_matches[i] != sub_txt_anket[i])
                        {
                            Checking[index] = i + 1;
                            index++;
                        }
                    if (index == 0)
                        sw.WriteLine("Результат: проверка
пройдена");
                    else
                    {
                        result = 0;
```

```
                        comment = "Результат: проверка не
пройдена. Найдены ошибки в следующих вопросах: ";
                        for (int i = 0; i < index; i++)
                        {
                            if (i != index - 1)
                                comment = comment + Checking[i] +
", ";
                            else
                                comment = comment + Checking[i];
                        }
                        sw.WriteLine(comment);
                    }
                }
                else
                {
                    comment = "Результат: проверка не пройдена.
Заданы не все впоросы";
                    sw.WriteLine(comment);
                    result = 0;
                }
                sw.WriteLine("----------------------------------
\n" + "**********************************");
            }
            if (connect.State != ConnectionState.Open)
                connect.Open(); // открываем базу данных
            OleDbCommand cmd = new OleDbCommand("SELECT * FROM
Records WHERE Path ='" + itemPath + "'", connect); // создаём
запрос
            OleDbDataReader reader = cmd.ExecuteReader(); //
получаем данные
            reader.Read();
            try
            {
```

```
                if (result == 1)
                    Common_functions.SQL_question("INSERT INTO
Check_Quality(Id_sotr, Dateofchecking, Id_record, Result,
Comment)" +
                        " VALUES ('" + (Transc_window.Owner as
Main).id + "', '" + DateTime.Now + "', '" + reader.GetValue(0) +
"', '" + result + "', '-')");
                else
                    Common_functions.SQL_question("INSERT INTO
Check_Quality(Id_sotr, Dateofchecking, Id_record, Result,
Comment)" +
                        " VALUES ('" + (Transc_window.Owner as
Main).id + "', '" + DateTime.Now + "', '" + reader.GetValue(0) +
"', '" + result + "', '" + comment + "')");
            }
            catch { }
            reader.Close();
            connect.Close();
        }

        private int Strtoint_1_15(string number)
        {
            int a = 0;
            switch (number)
            {
                case "один":
                    a = 1;
                    break;
                case "два":
                    a = 2;
                    break;
                case "три":
                    a = 3;
```

```
        break;
    case "четыре":
        a = 4;
        break;
    case "пять":
        a = 5;
        break;
    case "шесть":
        a = 6;
        break;
    case "семь":
        a = 7;
        break;
    case "восемь":
        a = 8;
        break;
    case "девять":
        a = 9;
        break;
    case "десять":
        a = 10;
        break;
    case "одиннадцать":
        a = 11;
        break;
    case "двенадцать":
        a = 12;
        break;
    case "тринадцать":
        a = 13;
        break;
    case "четырнадцать":
        a = 14;
```

```
                    break;
                case "пятнадцать":
                    a = 15;
                    break;
            }
            return a;
        }

        private void Transc_but_click(object sender,
RoutedEventArgs e)
        {
            SaveFileDialog save = new SaveFileDialog();
            save.Title = "Выберите место сохранения результатов
проверки ...";
            save.Filter = "Text file (*.txt)|*.txt";
            if ((bool)Choice_radio.IsChecked)
            {
                if (Records.SelectedItems.Count != 0)
                {
                    save.ShowDialog();
                    if (save.FileName != "")
                    {
                        using (StreamWriter sw = new
StreamWriter(save.FileName, false))
                        {
                            sw.WriteLine("\tРезультаты проверки\n"
+ "      выбранных аудиозаписей\n" +
"************************************\n" +

"************************************");
                        }
                        foreach (Records item in
Records.SelectedItems)
```

```
                        {
                            string[] str_matches =
Trancribation(item.Path);
                            string[] sub_txt_anket =
Split_txt_anket(item.Path);

//File.WriteAllLines(@"C:\Users\Vlad\Dropbox\diplom\audio\Checking
.txt", sub_txt_anket);

Write_transc_txt_in_file(save.FileName, item.Title, str_matches);
                            Write_txt_anket_in_file(save.FileName,
sub_txt_anket);

Checking_and_writing_in_database(save.FileName, item.Path,
sub_txt_anket, str_matches);
                        }

Common_functions.Show_message(Error_message2, "Процесс проверки
завершен!", true);
                        Process.Start(save.FileName);
                    }
                }
                else
                    Common_functions.Show_message(Error_message2,
"Выберите элементы в таблице!", false);
            }
            else
            {
                if (Date.SelectedDate.ToString() == "")
```

```
                    Common_functions.Show_message(Error_message2,
"Выберите дату!", false);
                else
                {
                    save.ShowDialog();
                    if (save.FileName != "")
                    {
                        int count = 0;
                        using (StreamWriter sw = new
StreamWriter(save.FileName, false))
                        {
                            sw.WriteLine("  Результаты проверки
аудиозаписей\n" + "\t   за " +
Date.SelectedDate.ToString().Substring(0, 10) +

"\n************************************\n" +
"************************************");
                        }
                        List<Records> records =
(List<Records>)Records.ItemsSource;
                        foreach (Records record in records)
                            if (record.Date.Date ==
Date.SelectedDate)
                            {
                                count++;
                                string[] str_matches =
Trancribation(record.Path);
                                string[] sub_txt_anket =
Split_txt_anket(record.Path);

Write_transc_txt_in_file(save.FileName, record.Title,
str_matches);
```

```
Write_txt_anket_in_file(save.FileName, sub_txt_anket);

Checking_and_writing_in_database(save.FileName, record.Path,
sub_txt_anket, str_matches);
                            }
                        if (count == 0)

Common_functions.Show_message(Error_message2, "За выбранную дату
анкеты не проводились!", false);
                        else
                        {

Common_functions.Show_message(Error_message2, "Процесс проверки
завершен!", true);
                            Process.Start(save.FileName);
                        }
                    }
                }
            }
        }

        private void Rec_grid_load(object sender, RoutedEventArgs
e)
        {
            if (connect.State != ConnectionState.Open)
                connect.Open(); // открываем базу данных
            OleDbCommand cmd = new OleDbCommand("SELECT * FROM
Records", connect); // создаём запрос
            OleDbDataReader read = cmd.ExecuteReader(); //
получаем данные
            List<Records> result = new List<Records>();
            while (read.Read())
```

```
            {
                OleDbCommand second_cmd = new OleDbCommand("SELECT
* FROM Users WHERE Id_sotr = " + read.GetValue(4), connect);
                OleDbDataReader second_read =
second_cmd.ExecuteReader();
                second_read.Read();
                string title = read.GetString(1);
                title = Common_functions.Regular(title,
@"(\w*).\w{3}\z", false);
                string FIO = second_read.GetString(6);
                int id_reg = (int)second_read.GetValue(5);
                second_read.Close();
                second_cmd = new OleDbCommand("SELECT * FROM
Regions WHERE Id_reg = " + id_reg, connect);
                second_read = second_cmd.ExecuteReader();
                second_read.Read();
                string str_reg = second_read.GetString(1);
                result.Add(new Records(title, read.GetString(1),
read.GetDateTime(3), FIO, str_reg));
                second_read.Close();
            }
            read.Close();
            connect.Close();
            Records.ItemsSource = result;
            Records.Columns[0].Header = "Название записи";
            Records.Columns[1].Header = "Путь к файлу";
            Records.Columns[1].Width = 200;
            Records.Columns[2].Header = "Дата анкетирования";
            (Records.Columns[2] as
DataGridTextColumn).Binding.StringFormat = "dd.MM.yyyy HH:mm";
            Records.Columns[3].Header = "ФИО интервьюера";
            Records.Columns[4].Header = "Идентификатор региона";
```

```
        }


        private void Date_checked(object sender, RoutedEventArgs
e)
        {
            if (Date_radio != null)
            {
                Common_functions.Hide_message(Error_message2);
                if (Date_radio.IsChecked == true)
                    Date.IsEnabled = true;
                else
                    Date.IsEnabled = false;
            }
        }
    }
    class Records
    {
        public Records(string Title, string Path, DateTime Date,
string FIO, string id_reg)
        {
            this.Title = Title;
            this.Path = Path;
            this.Date = Date;
            this.FIO = FIO;
            this.id_reg = id_reg;
        }

        public string Title { get; set; }
        public string Path { get; set; }
        public DateTime Date { get; set; }
        public string FIO { get; set; }
        public string id_reg { get; set; }
```

```
    }
public partial class Admin_Panel : Window
    {
        public Admin_Panel()
        {
            InitializeComponent();
            Combo.SelectedIndex = 0;
        }

        OleDbConnection connect = new
OleDbConnection(Common_functions.connection);

        private void datagrid_load(object sender, RoutedEventArgs
e)
        {
            if (connect.State != ConnectionState.Open)
                connect.Open(); // открываем базу данных
            OleDbCommand cmd = new OleDbCommand("SELECT * FROM
Users", connect);
            OleDbDataReader read = cmd.ExecuteReader();
            List<Users> result = new List<Users>();
            while (read.Read())
            {
                string Role = Check_role(read.GetString(7));
                OleDbCommand second_cmd = new OleDbCommand("SELECT
* FROM Regions WHERE Id_reg = " + read.GetValue(5), connect);
                OleDbDataReader second_read =
second_cmd.ExecuteReader();
                second_read.Read();

                result.Add(new Users(read.GetString(1),
read.GetString(2), read.GetString(3), read.GetString(4),
second_read.GetString(1), read.GetString(6), Role));
```

```
                second_read.Close();
            }
            read.Close();
            connect.Close();
            Database.ItemsSource = result;
            Database.Columns[0].Header = "Логин";
            Database.Columns[1].Header = "Пароль";
            Database.Columns[2].Header = "Почтовый адрес";
            Database.Columns[3].Header = "Фамилия Имя Отчество";
            Database.Columns[3].MaxWidth = 350;
            Database.Columns[4].Header = "Номер телефона";
            Database.Columns[5].Header = "Идентификатор региона";
            Database.Columns[6].Header = "Должность";
        }

        private int Check_and_input_id_reg()
        {
            int id = 0;
            if (Grid_id.Text != "-")
            {
                string id_region = "";
                id_region = Common_functions.Regular(Grid_id.Text,
@"[А-Я](\w+)(-[А-Я])?(\w+)-\d", false);
                if (id_region != "")
                {
                    if (connect.State != ConnectionState.Open)
                        connect.Open();
                    OleDbCommand cmd = new OleDbCommand("SELECT *
FROM Regions WHERE Title = '" + id_region + "'", connect);
                    OleDbDataReader read = cmd.ExecuteReader();
                    if (read.Read())
                        id = (int)read.GetValue(0);
                    else
```

```
                    {
                        Common_functions.SQL_question("INSERT INTO
Regions (Title) VALUES ('" + id_region + "')");
                        read.Close();
                        connect.Close();
                        connect.Open();
                        read = cmd.ExecuteReader();
                        read.Read();
                        id = (int)read.GetValue(0);
                    }
                    read.Close();
                }
                else
                    Common_functions.Show_message(Error_message1,
"Неферный формат идентификатора региона!", false);
            }
            else
                id = 1;
            return id;
        }


        private string Check_role(string Role)
        {
            string result = "";
            switch (Role)
            {
                case "Admin":
                    result = "Администратор";
                    break;
                case "Администратор":
                    result = "Admin";
                    break;
```

```
                case "User":
                    result = "Проверяющий";
                    break;
                case "Проверяющий":
                    result = "User";
                    break;
                case "Employee":
                    result = "Интервьюер";
                    break;
                case "Интервьюер":
                    result = "Employee";
                    break;
            }
            return result;
        }

        private void Add_click(object sender, RoutedEventArgs e)
        {
            if ((Grid_Log.Text == "") || (Grid_pass.Text == ""))
            {
                Common_functions.Show_message(Error_message1,
"Введите логин и пароль!", false);
                if (Grid_Log.Text == "")
                {
                    Grid_Log.BorderBrush =
Common_functions.RGBConvert("#FFFF6600");
                    Grid_Log.BorderThickness = new Thickness(2);
                }
                else
                {
                    Grid_pass.BorderBrush =
Common_functions.RGBConvert("#FFFF6600");
                    Grid_pass.BorderThickness = new Thickness(2);
```

```
                }

            }
            else
            {
                if (connect.State != ConnectionState.Open)
                    connect.Open(); // открываем базу данных
                OleDbCommand cmd = new OleDbCommand("SELECT * FROM
Users Where [Login] = '" + Grid_Log.Text + "'", connect); //
создаём запрос
                OleDbDataReader read = cmd.ExecuteReader();
                if (read.Read())
                {
                    Common_functions.Show_message(Error_message1,
"Такой логин уже зарегистрирован!", false);
                    read.Close();
                    connect.Close();
                }
                else
                {
                    read.Close();
                    Check_Empty();
                    int id = Check_and_input_id_reg();
                    connect.Close();
                    if (id != 0)
                    {
                        string Role = Check_role(Combo.Text);
                        Common_functions.SQL_question("INSERT INTO
Users (Login, [Password], [Email], [Number], [Id_reg], [FIO],
[Role] ) VALUES('" + Grid_Log.Text + "', '" + Grid_pass.Text + "',
'" + Grid_Email.Text + "', '" + Grid_Num.Text + "', " + id + ", '"
+ Grid_FIO.Text + "', '" + Role + "')");
                        ClearAll();
```

```
                        Error_message1.Foreground = Brushes.Green;

Common_functions.Show_message(Error_message1, "Пользователь
добавлен!", true);
                        datagrid_load(null, null);
                    }
                }
            }
        }

        private void Check_Empty()
        {
            foreach (TextBox item in TB_Own.Children)
                if (item.Text == "")
                    item.Text = "-";
            foreach (TextBox item in TB1_Own.Children)
                if (item.Text == "")
                    item.Text = "-";
        }

        private void ClearAll()
        {
            Grid_Log.Text = "";
            Grid_pass.Text = "";
            Grid_Email.Text = "";
            Grid_FIO.Text = "";
            Grid_Num.Text = "";
            Grid_id.Text = "";
            Common_functions.Hide_message(Error_message1);
            Grid_pass.BorderBrush = Brushes.Gray;
            Grid_Log.BorderBrush = Brushes.Gray;
            Grid_pass.BorderThickness = new Thickness(1);
            Grid_Log.BorderThickness = new Thickness(1);
```

```
            Grid_Log.IsReadOnly = false;
            Change_but.Content = "Изменить";
            Add_but.IsEnabled = true;
        }

        private void Cleare_click(object sender, RoutedEventArgs
e)
        {
            ClearAll();
        }

        private void Log_Lost_F(object sender, RoutedEventArgs e)
        {
            (sender as TextBox).BorderBrush = Brushes.Gray;
            (sender as TextBox).BorderThickness = new
Thickness(1);
        }

        private void Del_click(object sender, RoutedEventArgs e)
        {
            if (Database.SelectedItems.Count == 0)

Common_functions.Show_message(Error_message1,"Выберите
пользователей для удаления!", false);
            else
            {
                foreach (Users item in Database.SelectedItems)
                    Common_functions.SQL_question("DELETE * FROM
Users WHERE [Login] = '" + item.Login + "' and [Password] = '" +
item.Password + "'");
                datagrid_load(null, null);
                ClearAll();
```

```
Common_functions.Show_message(Error_message1,"Пользователь(ли)
удален(ны)!", true);
            }
        }

        private void Change_but_Click(object sender,
RoutedEventArgs e)
        {
            if ((string)(sender as Button).Content == "Изменить")
            {
                if (Database.SelectedItems.Count != 1)

Common_functions.Show_message(Error_message1,"Выберите одну строку
для редактирования!", false);
                else
                {
                    ClearAll();
                    Add_but.IsEnabled = false;
                    Users item = (Users)Database.SelectedItem;
                    Grid_Log.Text = item.Login;
                    Grid_Log.IsReadOnly = true;
                    Grid_pass.Text = item.Password;
                    Grid_Email.Text = item.Email;
                    Grid_FIO.Text = item.FIO;
                    Grid_Num.Text = item.Number;
                    Grid_id.Text = item.id_reg;
                    Combo.Text = item.Role;
                    (sender as Button).Content = "Сохранить";
                }
            }
            else
            {
```

```
                Check_Empty();
                int id = Check_and_input_id_reg();
                connect.Close();
                Common_functions.SQL_question("UPDATE Users SET
[Password] = '" + Grid_pass.Text + "', [Email] = '" +
Grid_Email.Text + "', [Number] = '" + Grid_Num.Text + "', [Id_reg]
= " +
                    id + ", [FIO] =  '" + Grid_FIO.Text + "',
[Role] = '" + Check_role(Combo.Text) + "' WHERE [Login] = '" +
Grid_Log.Text + "'");
                ClearAll();

Common_functions.Show_message(Error_message1,"Изменения
сохранены!", true);
                datagrid_load(null, null);
            }
        }
    }
    class Users
    {

        public Users(string Login, string Password, string Email,
string Number, string id_reg, string FIO, string Role)
        {
            this.Login = Login;
            this.Password = Password;
            this.Email = Email;
            this.Number = Number;
            this.id_reg = id_reg;
            this.FIO = FIO;
            this.Role = Role;

        }
```

```
        public string Login { get; set; }
        public string Password { get; set; }
        public string Email { get; set; }
        public string FIO { get; set; }
        public string Number { get; set; }
        public string id_reg { get; set; }
        public string Role { get; set; }
    }
    public partial class History_window : Window
    {
        public History_window()
        {
            InitializeComponent();
        }

        public int id = -1;
        OleDbConnection connect = new
OleDbConnection(Common_functions.connection);


        private void History_Loaded(object sender, RoutedEventArgs
e)
        {
            if (connect.State != ConnectionState.Open)
                connect.Open(); // открываем базу данных
            OleDbCommand cmd;
            if (id == -1)
                cmd = new OleDbCommand("SELECT * FROM
Check_Quality", connect); // создаём запрос
            else
                cmd = new OleDbCommand("SELECT * FROM
Check_Quality WHERE id_record IN (SELECT id_record FROM Records
WHERE id_emloyee = " + id + ")", connect);
```

```
            OleDbDataReader read = cmd.ExecuteReader(); //
получаем данные
            List<History> result = new List<History>();
            while (read.Read())
            {
                OleDbCommand second_cmd = new OleDbCommand("SELECT
* FROM Records WHERE id_record = " + read.GetValue(2), connect);
                OleDbDataReader second_read =
second_cmd.ExecuteReader();
                second_read.Read();
                string title = second_read.GetString(1);
                DateTime date_anket = second_read.GetDateTime(3);
                second_read.Close();
                second_cmd = new OleDbCommand("SELECT * FROM Users
WHERE Id_sotr = " + read.GetValue(5), connect);
                second_read = second_cmd.ExecuteReader();
                second_read.Read();
                string FIO = second_read.GetString(6);
                second_read.Close();
                title = Common_functions.Regular(title,
@"(\w*).\w{3}\z", false);
                result.Add(new History(title, read.GetDateTime(1),
date_anket,FIO, read.GetBoolean(3), read.GetString(4)));
            }
            read.Close();
            connect.Close();
            History.ItemsSource = result;
            History.Columns[0].Header = "Название записи";
            History.Columns[1].Header = "Дата проверки";
            (History.Columns[1] as
DataGridTextColumn).Binding.StringFormat = "dd.MM.yyyy HH:mm";
            History.Columns[2].Header = "Дата анкетирования";
```

```
            (History.Columns[2] as
DataGridTextColumn).Binding.StringFormat = "dd.MM.yyyy HH:mm";
            History.Columns[3].Header = "ФИО проверяющего";
            History.Columns[4].Header = "Результат";
            History.Columns[5].Header = "Комментарий";
        }

        private void Date_checked(object sender, RoutedEventArgs
e)
        {
            if (Date_radio != null)
            {
                Common_functions.Hide_message(Error_message3);
                if (Date_radio.IsChecked == true)
                    Date.IsEnabled = true;
                else
                    Date.IsEnabled = false;
            }
        }

        private void Button_Click(object sender, RoutedEventArgs
e)
        {
            SaveFileDialog save = new SaveFileDialog();
            save.Title = "Выберите место сохранения результатов
проверки ...";
            save.Filter = "Text file (*.txt)|*.txt";
            if ((bool)Choice_radio.IsChecked)
            {
                if (History.SelectedItems.Count != 0)
                {
                    save.ShowDialog();
                    if (save.FileName != "")
```

```
                    {
                        using (StreamWriter sw = new
StreamWriter(save.FileName, false))
                        {
                            sw.WriteLine("\tИстория проверок\n" +
"      выбранных аудиозаписей\n" +
"************************************\n" +

"************************************");
                        }
                        foreach (History item in
History.SelectedItems)
                        {
                            using (StreamWriter sw = new
StreamWriter(save.FileName, true))
                            {
                                if (item.result == true)

                                    sw.WriteLine("Название записи:
" + Common_functions.Regular(item.Title, @"(\w*).\w{3}\z", false)
+ "\n-------------------------------------\n" +
                                    "Результат: проверка пройдена"
+ "\n-------------------------------------\n" +
"************************************");
                                else
                                    sw.WriteLine("Название записи:
" + Common_functions.Regular(item.Title, @"(\w*).\w{3}\z", false)
+ "\n-------------------------------------\n" +
                                        item.comment + "\n--------
---------------------------\n" +
"************************************");
                            }
                        }
```

```
Common_functions.Show_message(Error_message3, "Выгрузка
завершена!", true);
                        Process.Start(save.FileName);
                    }
                }
                else
                    Common_functions.Show_message(Error_message3,
"Выберите элементы в таблице!", false);
            }
            else
            {
                if (Date.SelectedDate.ToString() == "")


Common_functions.Show_message(Error_message3,"Выберите дату!",
false);
                else
                {
                    save.ShowDialog();
                    if (save.FileName != "")
                    {
                        List<History> items =
(List<History>)History.ItemsSource;
                        using (StreamWriter sw = new
StreamWriter(save.FileName, false))
                        {
                            sw.WriteLine("\t История проверок\n" +
"      аудиозаписей анкет за\n" + "\t    " +
Date.SelectedDate.ToString().Substring(0, 10) +

"\n***********************************\n" +
"***********************************");
```

```
                        }
                        int count = 0;
                        foreach (History item in items)
                        {
                            if (item.Date_anket.Date ==
Date.SelectedDate)
                            {
                                count++;
                                using (StreamWriter sw = new
StreamWriter(save.FileName, true))
                                {
                                    if (item.result)
                                        sw.WriteLine("Название
записи: " + Common_functions.Regular(item.Title,

@"(\w*).\w{3}\z", false) + "\n----------------------------------
\n" +
                                        "Результат: проверка
пройдена" + "\n----------------------------------\n" +
"**********************************");
                                    else
                                        sw.WriteLine("Название
записи: " + Common_functions.Regular(item.Title, @"(\w*).\w{3}\z",
false) + "\n----------------------------------\n" +
                                            item.comment + "\n----
--------------------------------\n" +
"**********************************");
                                }
                            }
                        }
                        if (count == 0)
```

```
Common_functions.Show_message(Error_message3, "За выбранную дату
анкеты не проводились!", false);
                        else
                        {

Common_functions.Show_message(Error_message3, "Выгрузка
завершена!", true);
                            Process.Start(save.FileName);
                        }
                    }
                }
            }
        }

        private void Del_but_click(object sender, RoutedEventArgs
e)
        {
            if (History.SelectedItems.Count == 0)
                Common_functions.Show_message(Error_message3,
"Выберите строки для удаления!", false);
            else
            {
                if (connect.State != ConnectionState.Open)
                    connect.Open();
                foreach (History item in History.SelectedItems)
                {
                    OleDbCommand cmd = new OleDbCommand("DELETE *
FROM Check_Quality WHERE [Dateofchecking] = @Date", connect); //
создаём запрос
                    cmd.Parameters.Add("@Date",
OleDbType.Date).Value = item.Date;
                    cmd.ExecuteNonQuery();
```

```
                }
                connect.Close();
                History_Loaded(null, null);
                Common_functions.Show_message(Error_message3,
"Удаление завершено!", true);
            }
        }
    }
    class History
    {
        public History (string Title, DateTime Date, DateTime
Date_anket, string FIO, bool result, string comment)
        {
            this.Title = Title;
            this.Date = Date;

            this.Date_anket = Date_anket;
            this.FIO = FIO;
            this.result = result;
            this.comment = comment;
        }

        public string Title { get; set; }
        public DateTime Date { get; set; }
        public DateTime Date_anket { get; set; }
        public string FIO { get; set; }
        public bool result { get; set; }
        public string comment { get; set; }

    }
    public partial class Main : Window
    {
        public int id;
```

```
        public Main()
        {
            InitializeComponent();
        }
        public OleDbConnection connect = new
OleDbConnection(Common_functions.connection);

        private void Admin_button_click(object sender,
RoutedEventArgs e)
        {
            Admin_Panel Admin_Panel = new Admin_Panel();
            Admin_Panel.Owner = this;
            Admin_Panel.ShowDialog();
        }

        private void Transc_but_click(object sender,
RoutedEventArgs e)
        {
            Tranc_window Tranc_Window = new Tranc_window();
            Tranc_Window.Owner = this;
            Tranc_Window.ShowDialog();
        }

        private void History_but_click(object sender,
RoutedEventArgs e)
        {
            if (connect.State != ConnectionState.Open)
                connect.Open();
            OleDbCommand cmd = new OleDbCommand("SELECT * FROM
Users WHERE Id_sotr = " + id, connect); // создаём запрос
            OleDbDataReader read = cmd.ExecuteReader();
            read.Read();
            History_window History_Window = new History_window();
```

```
            if (read.GetString(7) == "Employee")
                History_Window.id = id;
            else
                if (read.GetString(7) == "Admin")
                {
                    History_Window.Del_but.Visibility =
Visibility.Visible;
                    History_Window.Del_but.Margin = new
Thickness(10, 17, 10, 12);
                }
            read.Close();
            connect.Close();
            History_Window.Owner = this;
            History_Window.ShowDialog();
        }
    }
    public partial class MainWindow : Window
    {
        Main Main = new Main();
        public MainWindow()
        {
            InitializeComponent();
        }
        OleDbConnection connect = new
OleDbConnection(Common_functions.connection);
        private void auth_but_click(object sender, RoutedEventArgs
e)
        {
            OleDbDataReader read;
            try
            {
                if (connect.State != ConnectionState.Open)
                    connect.Open();
```

```
                OleDbCommand cmd = new OleDbCommand("SELECT * FROM
Users WHERE Login = '" + Log.Text + "'", connect); // создаём
запрос
                read = cmd.ExecuteReader();
            }
            catch
            {
                read = null;
            }
            if (read != null)
            {
                if (read.Read() && pass.Password ==
read.GetString(2))
                {
                    string Role = read.GetString(7);
                    if (Role == "Admin")
                    {
                        Main.admin_button.Visibility =
Visibility.Visible;
                        Main.admin_button.Margin = new
Thickness(10, -5, 10, 15);
                    }
                    else if(Role == "Employee")
                    {
                        Main.Transc_but.Visibility =
Visibility.Hidden;
                        Main.Transc_but.Margin = new
Thickness(10,0,10,-30);
                    }
                    Main.id = (int)read.GetValue(0);
                    Main.Show();
                    Close();
                }
```

```
                else
                    Common_functions.Show_message(Error_message,
"Введены некорректные данные", false);
            }
            else
                Common_functions.Show_message(Error_message, "Не
удается получить данные из БД.\nОбратитесь к адмиристратору",
false);
            connect.Close();
        }
    }
public partial class Messages : Window
    {
        public Messages()
        {
            InitializeComponent();
        }

        public int id;

        OleDbConnection connect = new
OleDbConnection(Common_functions.connection);

        private void From_mess_load(object sender, RoutedEventArgs
e)
        {
            if (connect.State != ConnectionState.Open)
                connect.Open();
            OleDbCommand cmd = new OleDbCommand("SELECT * FROM
Messages WHERE Id_To = " + id, connect); // создаём запрос
            OleDbDataReader read = cmd.ExecuteReader(); //
получаем данные
```

```
            List<About_message> result = new
List<About_message>();
            while (read.Read())
            {
                OleDbCommand second_cmd = new OleDbCommand("SELECT
* FROM Users WHERE Id_sotr = " + read.GetValue(1), connect);
                OleDbDataReader second_read =
second_cmd.ExecuteReader();
                second_read.Read();
                string FIO = second_read.GetString(6);
                int id_reg = (int)second_read.GetValue(5);
                second_read.Close();
                second_cmd = new OleDbCommand("SELECT * FROM
Regions WHERE Id_reg = " + id_reg, connect);
                second_read = second_cmd.ExecuteReader();
                second_read.Read();
                string str_reg = second_read.GetString(1);
                result.Add(new
About_message((int)read.GetValue(0), FIO, read.GetDateTime(3),
str_reg, read.GetBoolean(5)));
                second_read.Close();
            }
            read.Close();
            connect.Close();
            if (result.Count == 0)
            {
                Inc.Visibility = Visibility.Hidden;
                Inc_empty.Visibility = Visibility.Visible;
            }
            else
            {
                Inc.Visibility = Visibility.Visible;
                Inc_empty.Visibility = Visibility.Hidden;
```

```
                Inc.ItemsSource = result;
                Inc.Columns[0].Header = "id сообщения";
                Inc.Columns[1].Header = "ФИО";
                Inc.Columns[2].Header = "Дата отправления";
                (Inc.Columns[2] as
DataGridTextColumn).Binding.StringFormat = "dd.MM.yyyy HH:mm";
                Inc.Columns[3].Header = "Идентификатор региона";
                Inc.Columns[4].Header = "Статус";
            }
        }

        public void To_mess_load(object sender, RoutedEventArgs e)
        {
            if (connect.State != ConnectionState.Open)
                connect.Open();
            OleDbCommand cmd = new OleDbCommand("SELECT * FROM
Messages WHERE Id_From = " + id, connect); // создаём запрос
            OleDbDataReader read = cmd.ExecuteReader(); //
получаем данные
            List<About_message> result = new
List<About_message>();
            while (read.Read())
            {
                OleDbCommand second_cmd = new OleDbCommand("SELECT
* FROM Users WHERE Id_sotr = " + read.GetValue(2), connect);
                OleDbDataReader second_read =
second_cmd.ExecuteReader();
                second_read.Read();
                string FIO = second_read.GetString(6);
                int id_reg = (int)second_read.GetValue(5);
                second_read.Close();
                second_cmd = new OleDbCommand("SELECT * FROM
Regions WHERE Id_reg = " + id_reg, connect);
```

```
                second_read = second_cmd.ExecuteReader();
                second_read.Read();
                string str_reg = second_read.GetString(1);
                result.Add(new
About_message((int)read.GetValue(0), FIO, read.GetDateTime(3),
str_reg, read.GetBoolean(5)));
                second_read.Close();
            }
            read.Close();
            connect.Close();
            if (result.Count == 0)
            {
                Send.Visibility = Visibility.Hidden;
                Send_empty.Visibility = Visibility.Visible;
            }
            else
            {
                Send.Visibility = Visibility.Visible;
                Send_empty.Visibility = Visibility.Hidden;
                Send.ItemsSource = result;
                Send.Columns[0].Header = "id сообщения";
                Send.Columns[1].Header = "ФИО";
                Send.Columns[2].Header = "Дата отправления";
                (Send.Columns[2] as
DataGridTextColumn).Binding.StringFormat = "dd.MM.yyyy HH:mm";
                Send.Columns[3].Header = "Идентификатор региона";
                Send.Columns[4].Header = "Статус";
            }
        }

        private void Check_mess_click(object sender,
RoutedEventArgs e)
        {
```

```
            if (Inc_radio.IsChecked == true)
            {
                if (Inc.SelectedItems.Count != 1)
                    Common_functions.Show_message(Error_message4,
"Выберите одно принятое сообщение!", false);
                else
                {
                    if (connect.State != ConnectionState.Open)
                        connect.Open();
                    About_message item =
(About_message)Inc.SelectedItem;
                    OleDbCommand cmd = new OleDbCommand("SELECT *
FROM Messages WHERE id_message = " + item.id, connect); // создаём
запрос
                    OleDbDataReader read = cmd.ExecuteReader(); //
получаем данные
                    read.Read();
                    string text = read.GetString(4);
                    read.Close();
                    connect.Close();
                    MessBox box = new MessBox();
                    box.Title = "Принятое сообщение";
                    box.txt_message.Text = text;
                    box.ShowDialog();
                    if (item.Result != true)
                        Common_functions.SQL_question("UPDATE
Messages SET [Status] = 1 WHERE id_message = " + item.id);
                    From_mess_load(null, null);
                    Common_functions.Hide_message(Error_message4);
                }
            }
            else
            {
```

```
                if (Send.SelectedItems.Count != 1)
                    Common_functions.Show_message(Error_message4,
"Выберите одно отправленное сообщение!", false);
                else
                {
                    if (connect.State != ConnectionState.Open)
                        connect.Open();
                    About_message item =
(About_message)Send.SelectedItem;
                    OleDbCommand cmd = new OleDbCommand("SELECT *
FROM Messages WHERE id_message = " + item.id, connect); // создаём
запрос
                    OleDbDataReader read = cmd.ExecuteReader(); //
получаем данные
                    read.Read();
                    string text = read.GetString(4);
                    read.Close();
                    connect.Close();
                    MessBox box = new MessBox();
                    box.Title = "Отправленное сообщение";
                    box.txt_message.Text = text;
                    box.ShowDialog();
                    Common_functions.Hide_message(Error_message4);
                }
            }
        }

        private void Hide(object sender, RoutedEventArgs e)
        {
            if(Error_message4 != null)
                Common_functions.Hide_message(Error_message4);
        }
```

```
        private void Del_click(object sender, RoutedEventArgs e)
        {
            if (Inc_radio.IsChecked == true)
            {
                if (Inc.SelectedItems.Count == 0)
                    Common_functions.Show_message(Error_message4,
"Выберите принятые сообщения!", false);
                else
                {
                    foreach (About_message item in
Inc.SelectedItems)
                        Common_functions.SQL_question("DELETE *
FROM Messages WHERE id_message = " + item.id);
                    From_mess_load(null, null);
                    Common_functions.Show_message(Error_message4,
"Сообщени(е/я) удалены!", true);
                }
            }
            else
            {
                if (Send.SelectedItems.Count == 0)
                    Common_functions.Show_message(Error_message4,
"Выберите отправленные сообщения!", false);
                else
                {
                    foreach (About_message item in
Send.SelectedItems)
                        Common_functions.SQL_question("DELETE *
FROM Messages WHERE id_message = " + item.id);
                    To_mess_load(null, null);
                    Common_functions.Show_message(Error_message4,
"Сообщени(е/я) удалены!", true);
                }
```

```
            }
        }

        private void New_mess(object sender, RoutedEventArgs e)
        {
            New_message window = new New_message();
            window.Owner = this;
            window.ShowDialog();
        }
    }

    class About_message
    {
        public About_message(int id, string FIO, DateTime Date,
string Region , bool Result)
        {
            this.id = id;
            this.FIO = FIO;
            this.Date = Date;
            this.Region = Region;
            this.Result = Result;
        }
        public int id { get; set; }
        public string FIO { get; set; }
        public DateTime Date { get; set; }
        public string Region { get; set; }
        public bool Result { get; set; }
    }
public partial class MessBox : Window
    {
        public MessBox()
        {
            InitializeComponent();
```

```
        }

        private void Close_click(object sender, RoutedEventArgs e)
        {
            Close();
        }
    }
    public partial class New_message : Window
    {
        public New_message()
        {
            InitializeComponent();
        }

        OleDbConnection connect = new
OleDbConnection(Common_functions.connection);

        private void Send_click(object sender, RoutedEventArgs e)
        {
            string[] Log = Login.SelectedItem.ToString().Split(new
char[] { ' ' });
            if (connect.State != ConnectionState.Open)
                connect.Open();
            OleDbCommand cmd = new OleDbCommand("SELECT * FROM
Users Where [Login] = '" + Log[0] + "'", connect); // создаём
запрос
            OleDbDataReader read = cmd.ExecuteReader();
            if (read.Read())
            {
                if (Mess.Text == "")
                    Common_functions.Show_message(Error_message5,
"Введите сообщение!", false);
                else
```

```
                {
                    string text = Mess.Text;
                    text = text.Trim();
                    Common_functions.SQL_question("INSERT INTO
Messages (Id_From, Id_To, Date_message, Text_message, Status)
VALUES " +
                        "(" + (New.Owner as Messages).id + ", " +
(int)read.GetValue(0) + ", '" + DateTime.Now + "', '" + text + "',
0)");
                    Login.Text = "";
                    Mess.Text = "";
                    Common_functions.Show_message(Error_message5,
"Сообщение отправлено!", true);
                    (New.Owner as Messages).To_mess_load(null,
null);
                }
            }
            else
                Common_functions.Show_message(Error_message5,
"Неверный логин пользователя!", false);
            read.Close();
            connect.Close();
        }

        private void New_message_loaded(object sender,
RoutedEventArgs e)
        {
            if (connect.State != ConnectionState.Open)
                connect.Open();
            OleDbCommand cmd = new OleDbCommand("SELECT * FROM
Users", connect); // создаём запрос
            OleDbDataReader read = cmd.ExecuteReader();
            while (read.Read())
```

Окончание ПРИЛОЖЕНИЯ А

```
            {
                string item = read.GetString(1) + " - " +
read.GetString(6);
                Login.Items.Add(item);
            }
            Login.SelectedIndex = 0;
            read.Close();
            connect.Close();
        }
    }
}
```

ПРИЛОЖЕНИЕ Б

Описание класса Common_functions

```
using System;
using System.Collections.Generic;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;

namespace Check_Quality
{
    static class Common_functions
    {
        static public string connection =
@"Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=C:\Users\Vlad\Dropbox\diplom\database\diplom.accdb";

        static public Brush RGBConvert(string a)
        {
            var converter = new
System.Windows.Media.BrushConverter();
            var brush = (Brush)converter.ConvertFromString(a);
            return brush;
        }

        static public void Show_message(TextBlock item, string
txt, bool Color)
        {
            if (Color)
```

```
                item.Foreground = Brushes.Green;
            else
                item.Foreground = RGBConvert("#FFFF6600");
            item.Text = txt;
            item.Margin = new Thickness(0, 0, 0, 0);
            item.Visibility = Visibility.Visible;
        }

        static public void Hide_message(TextBlock item)
        {
            item.Visibility = Visibility.Hidden;
            item.Margin = new Thickness(0, -20, 0, 0);
        }

        static public void SQL_question(string selectSQL) //
функция подключения к базе данных и обработка запросов
        {
            string connection =
@"Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=C:\Users\Vlad\Dropbox\diplom\database\diplom.accdb";
            OleDbConnection connect = new
OleDbConnection(connection); // подключаемся к базе данных
            connect.Open(); // открываем базу данных
            OleDbCommand cmd = new OleDbCommand(selectSQL,
connect); // создаём запрос
            cmd.ExecuteNonQuery();
            connect.Close();
        }

        static public dynamic Regular(string title, string regex,
bool collection)
        {
            if (collection)
```

Окончание ПРИЛОЖЕНИЯ Б

```
            {
                Regex r = new Regex(regex);
                MatchCollection matches = r.Matches(title);
                return matches;
            }
            else
            {
                Regex regexp = new Regex(regex);
                Match match = regexp.Match(title);
                return match.Value;
            }
        }
    }
}
```

Заявление о самостоятельном характере выполнения выпускной квалификационной работы

Я, Казанцев Владислав Олегович, студент 4 курса, направления подготовки / специальности «Программная инженерия», заявляю, что в моей работе на тему «Разработка программного модуля для автоматизации работы отдела контроля качества call-центра», представленной в Государственную экзаменационную комиссию для публичной защиты, не содержится элементов неправомерных заимствований.

Все прямые заимствования из печатных и электронных источников, а также ранее защищенных письменных работ, кандидатских и докторских диссертаций имеют соответствующие ссылки.

Я ознакомлен с действующим в Университете Положением о проверке работ обучающихся ФГБОУ ВО «МГУ им. Н.П. Огарева» на наличие заимствований, в соответствии с которым обнаружение неправомерных заимствований является основанием для отрицательного отзыва руководителя работы.

ФИО Казанцев Владислав Олегович подпись Каз

дата 12.06.2020

ФОРМА ОТЧЕТА
о результатах проверки работы обучающегося
на наличие заимствований

Ф.И.О. автора работы Казанцев Владислав Олегович

Тема работы «Разработка программного модуля для автоматизации работы отдела контроля качества call-центра»

Руководитель канд. физ.-мат. наук, доц. С. А. Фирсова

Представленная работа прошла проверку на наличие заимствований в системе «Антиплагиат.ВУЗ» (или иной аналогичной системе, анализа текстов на наличие заимствований, выбранной Университетом).

Результат автоматической проверки: оригинальность 87,51%

цитирования 1,87%

заимствования 10,62%

Результаты анализа полного отчета на наличие заимствований:

правомерные заимствования: 10,62%

корректные цитирования: 1.87%

неправомерные заимствования: отсутствуют, все источники указаны в списке использованной литературы

признаки обхода системы: отсутствуют

Общее заключение об итоговой оригинальности работы и возможности ее допуска к защите:

Выпускная квалификационная работа имеет высокую степень оригинальности, не содержит элементов неправомерных заимствований. Автор работы может быть допущен к ее защите.

Руководитель
канд. физ.-мат. наук,
доцент  С. А. Фирсова

Новые Решения

430034, РФ, г. Саранск,
ул. Лодыгина, 3, офис 322 (Технопарк-Мордовия)
Сайт: www.neosar.ru, e-mail: info@neosar.ru
Телефоны: 8 (499) 579-88-34, 8 (8342) 34-04-03

ЗАКЛЮЧЕНИЕ
на выпускную квалификационную работу
студента Казанцева Владислава Олеговича,
обучающегося по направлению подготовки
09.03.04 «Программная инженерия» (прикладной бакалавриат)
на тему: «Разработка программного модуля для автоматизации работы отдела контроля качества call-центра»

На данный момент деятельность call-центра тесно связана с процессами, которые, не смотря на технический прогресс, все еще остаются не автоматизированными. Выпускная квалификационная работа Казанцева В. О. посвящена разработке программного модуля для автоматизации работы отдела контроля качества call-центра.

В работе проведен обзор существующих систем автоматизации, а именно: системы учета рабочего времени Dialer Express, эмулятора телефонного набора X_Lite и web-приложения SurveyStudio, в котором интервьюер заполняет социологическую анкету; а также показана логика их взаимодействия друг с другом.

Новые Решения

430034, РФ, г. Саранск,
ул. Лодыгина, 3, офис 322 (Технопарк-Мордовия)
Сайт: www.neosar.ru; e-mail: info@neosar.ru
Телефоны: 8 (499) 579-88-34, 8 (8342) 34-04-03

Студентом был осуществлен анализ предметной области выпускной квалификационной работы и проведен обзор сервисов транскрибации, таких как: Dragon Mobile SDK, Google Speech Recognition API, Microsoft Speech API. Это позволило определить наиболее подходящие инструменты для решения поставленной задачи.

В процессе разработки программного модуля, Казанцевым В. О. была спроектирована архитектура системы: построены диаграмма вариантов использования, диаграмма состояний, диаграмма базы данных, а также описаны схема интеграции и схема внутреннего взаимодействия компонентов.

В результате разработки был создан программный модуль, который обеспечивает автоматизацию работы отдела контроля качества call-центра, а именно, осуществляет проверку качества проведенных при социологическом исследовании анкет.

Для реализации программного модуля студентом были выбраны язык программирования C# и платформа WPF (Windows Presentation Foundation), включенные в среду разработки Microsoft Visual Studio 2019, а для создания и управления базой данных была выбрана СУБД Microsoft Access.

Настоящая выпускная квалификационная работа полностью соответствует заданию по своему содержанию и объему.

Новые Решения

430034, РФ, г. Саранск,
ул. Лодыгина, 3, офис 322 (Технопарк-Мордовия)
Сайт: www.neosar.ru, e-mail: info@neosar.ru
Телефоны: 8 (499) 579-88-34, 8 (8342) 34-04-03

Работа написана ясно и грамотно, отличается четкостью изложения, имеет хорошее качество оформления. Выпускная квалификационная работа студента Казанцева В.О. достаточно полно раскрывает рассматриваемую тему, полностью соответствует предъявленным требованиям и заслуживает оценки «отлично», а ее автор – присвоения соответствующей квалификации бакалавра по направлению подготовки 09.03.04 «Программная инженерия».

Директор
ООО «Новые Решения» 26.06.20 Иванов М. Ю.
(подпись, дата)

ОТЗЫВ

о выпускной квалификационной работе студента 4-го курса факультета математики и информационных технологий направления подготовки 09.03.04 «Программная инженерия» Мордовского государственного университета им. Н. П. Огарева Казанцева В. О. на тему: «Разработка программного модуля для автоматизации работы отдела контроля качества call-центра»

Выпускная квалификационная работа Казанцева В. О. посвящена разработке программного модуля для автоматизации работы отдела контроля качества call-центра.

Актуальность данного исследования обусловлена тем, что на данный момент качество проведенных при социологическом исследовании анкет контролируется людьми вместо автоматизированной системы.

Результатом разработки является приложение на платформе WPF, функционал которого позволяет программными средствами осуществлять проверку качества работы сотрудников отдела контроля качества call-центра, а именно проверять соответствие между ответами респондента в записанной аудиозаписи разговора и текстом анкеты, составленным интервьюером.

Разработанное приложение, в силу особенности своей архитектуры, имеет перспективы интеграции с существующими системами автоматизации процессов, тесно связанных с деятельностью call-центра.

При выполнении выпускной квалификационной работы Казанцев В. О. грамотно использовал базовые теоретические знания и практические навыки, полученные за годы обучения в университете. Он зарекомендовал себя как квалифицированный специалист, способный в дальнейшем самостоятельно решать научные и производственные задачи, показал высокий уровень владения методологиями, методами и инструментами программной инженерии при разработке программного обеспечения.

При проверке итоговая оценка оригинальности работы из отчета о проверке работы на наличие заимствований составляет 87,51%.

Исходя из вышеизложенного, считаю, что выпускная квалификационная работа Казанцева В. О. заслуживает оценки «отлично».

Руководитель
доцент кафедры САПР
ФГБОУ ВО «МГУ им. Н. П. Огарёва»
канд. физ.-мат. наук  С. А. Фирсова