

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

CRM-СИСТЕМА ПРЕДПРИЯТИЯ

Студент

31.05.2022

(подпись, дата)

Алпатова Алёна Андреевна

(ФИО полностью)

Руководитель

31.05.2022

(подпись, дата)

Н.С. Карамышева

(фамилия, инициалы)

Нормоконтролёр

31.05.2022

(подпись, дата)

А.С. Бычков

(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.2022 г. №

13)

Заведующий кафедрой

(подпись)

М.А. Митрохин

(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

07.06.2022 г. №

3)

Секретарь ГЭК

(подпись)

Е.И. Гурин

(фамилия, инициалы)

Пенза 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

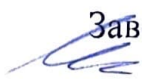
вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

 М.А. Митрохин

« 26 » апреля 20 22 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Алпатова Алёна Андреевна гр. 18BB2 факультета вычислительной техники

2. Тема работы «CRM-система предприятия»

Тема утверждена приказом ПГУ № 357/о от "26" апреля 2022 г.

3. Руководитель бакалаврской работы к.т.н., доцент каф. ВТ ПГУ Карамышева Н. С.

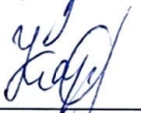
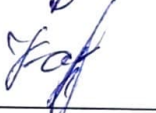
4. Задание на работу (назначение разработки, исходные данные и т.п)

Разработать приложение «CRM-система предприятия», выполняющее учет, обмен и структурирование информации по производству, обеспечивающее взаимодействие сотрудников, повышающую эффективность работы предприятия в целом. Приложение должно иметь клиент-серверную архитектуру. Клиентская часть должна обладать графическим интерфейсом.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области, изучить аналоги и шаблоны существующих CRM-систем, изучить способы хранения данных, в соответствии с этим выбрать архитектуру хранения данных и систему управления базой данных, изучить архитектуру клиент-серверных приложений, выбрать среду разработки, определить интерфейс приложения, спроектировать приложение, реализовать приложение, отладить и провести тестирование разработанного приложения.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	14	26.04.2022-28.04.2022	
2.	Анализ требований	10	29.04.2022-30.04.2022	
3.	Выбор технологий, применяемых при проектировании и разработки	10	01.05.2022	
4.	Разработка приложения «CRM-система предприятия»	247	01.05.2022-21.05.2022	
5.	Отладка и тестирование приложения	25	22.05.2022-26.05.2022	
6.	Оформление пояснительной записки	19	27.05.2022-31.05.2022	

Дата выдачи задания "26" апреля 2022 г.

Руководитель бакалаврской работы



26.04.2022

(подпись, дата)

Задание к исполнению принял студент



26.04.2022

(подпись, дата)

Работу к защите допустить
Декан факультета



(подпись, дата)

Реферат

Работа содержит 168 страниц формата А4, 30 рисунков, 7 таблиц, 8 источников, 3 приложения.

CRM-СИСТЕМА, *CRM*, БАЗА ДАННЫХ, ИНТЕРФЕЙС, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА.

Объект разработки - *CRM*-система предприятия.

Целью дипломной работы является разработка *CRM*-системы предприятия, предназначенной для повышения эффективности, оперативности, устойчивости работы предприятия и управленческих решений. Это достигается путем заполнения и сбора информации в хранилище данных, введенных сотрудниками предприятия. Необходимо решить задачу повышения производительности за счет оптимизации процесса обмена информации и оперативного взаимодействия сотрудников.

В результате проведенной работы создано приложение, протестированы программные средства, удовлетворяющие поставленной задаче.

В пояснительной записке приведен анализ предметной области, обзор технологии разработки и отладки программы.

Графическая часть содержит инфологическую и физическую модели БД и *UML*-диаграммы приложения.

					ПГУ 1.090301.01.001 ПЗ				
Изм.	Лист	№ докум	Подпись	Дата	CRM-система предприятия Пояснительная записка	Лит.	Лист	Листов	
Разраб.		Алпатова А.А.							
Провер.		Карамышева Н.С.					2	168	
Н. Контр.		Бычков А.С.				ФВТ, каф. ВТ гр. 18ВВ2			
Утверд.		Митрохин М.А.							

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1 CRM-система предприятия.....	7
1.1 Анализ предметной области	7
1.1.1 Анализ существующих CRM-систем.....	12
1.1.2 Постановка задачи	16
1.2 Архитектура приложения и выбор инструментов разработки.....	17
1.2.1 Выбор архитектуры приложения	17
1.2.2 Хранение данных.....	19
1.2.3 Взаимодействие клиента и сервера	23
1.2.4 Выбор языка программирования	24
1.2.5 Выбор СУБД	25
1.2.6 Выбор среды разработки	26
1.3 Процесс проектирования и разработки.....	28
1.3.1 Реализация интерфейса.....	30
1.3.2 Проектирование и разработка серверной части приложения.....	40
1.3.3 Проектирование и разработка БД.....	43
1.3.4 Проектирование и разработка клиентской части приложения....	53
1.4 Отладка и тестиование	57
1.4.1 Тестирование окна авторизации	58
1.4.2 Тестирование рабочего пространства	63
1.4.3 Тестирование сообщений	68
2 Безопасность жизнедеятельности.....	69
2.1 Потенциально-опасные и вредные факторы	69
2.2 Пожарная безопасность	70
2.3 Электробезопасность	73
2.4 Определение класса условий труда.....	74
Заключение	78
Список используемых источников.....	79
Приложение А – Листинг программы.....	80

Приложение Б – *UML*-диаграммы..... 155

Приложение В – Презентация..... 160

Перечень сокращений и обозначений

API (Application Programming Interface) – Программный интерфейс приложения

CRM (Customer Relationship Management) – система управления взаимоотношениями с клиентами

CRUD (Create, Read, Update, Delete) – функции создания, чтения, модификации, удаления при работе с базами данных

ID (data name, identifier) – опознаватель

IP (Internet Protocol) – межсетевой протокол

SQL (Structured query language) – язык структурированных запросов;

TCP (Transmission Control Protocol) – протокол управления передачей

UML (Unified Modeling Language) – унифицированный язык моделирования;

БД – База данных

СУБД – система управления базами данных

ПО – программное обеспечение

					ПГУ 1.090301.01.001 ПЗ	Лист
						5 5
Изм.	Лист	№ докум.№	Подпись	Дата		

Введение

Основой современной рыночной экономики, подверженной постоянному и всеохватывающему развитию, являются предприятия. В целях поддержания конкурентоспособности сегодняшнее предприятие вынуждено непрерывно повышать эффективность, обеспечивать устойчивость и оперативность управленческих решений, а также обладать мощными инструментами планирования работы. Решением данной задачи является использование автоматизированной среды управления предприятием, а именно внедрение *CRM*-системы. При этом большое значение будет иметь *CRM*-система, учитывающая нужды и специфику текущего предприятия.

Актуальность выбранной темы обусловлена потребностью в автоматизации взаимодействия с заказчиком и между сотрудниками, в управлении информацией о процессах производства и клиентах в целях улучшения и оптимизации управления предприятием в условиях рыночной экономики.

Объектом исследования является *CRM*-система предприятия.

Цель работы – разработка информационного обеспечения для структуризации и оптимизации деятельности предприятия.

1 CRM-система предприятия

1.1 Анализ предметной области

На сегодняшний день существует огромное количество разного рода предприятий, фактором успеха которых является грамотная автоматизация каждого процесса предприятия.

В условиях рыночной экономики управление современным предприятием — это сложный и глубокий процесс, включающий выбор и реализацию определенного набора управленческих воздействий на текущих временных отрезках. Для данных воздействий в качестве основной цели является решение стратегической задачи обеспечения устойчивого финансового и социально - экономического развития предприятия.

Современная экономика создает ситуацию, в которой появляется необходимость постоянного повышения эффективности производства, максимально быстрого реагирования на изменения любой степени важности, улучшения качества предоставляемых услуг, снижения разного рода потерь, а также строительства эффективных прогнозов на будущее [1].

И сегодня все больше руководителей компаний начинают понимать, что проблему выживания нынешнего предприятия решает не только оптимизация производства.

В сложившейся ситуации особенно актуальны как точное планирование, так и активная работа с имеющимися клиентами, а также их удержание.

Предприятие может потерять существенные средства из-за недостаточно эффективной работы в условиях отсутствия единой выверенной информационной базы со значимыми данными.

Для этого руководителю необходимо обладать достоверной информацией при проведении полного анализа деятельности учреждения в

режиме реального времени. Это может быть информация о состоянии производственного цикла, об учете товарных запасов и их перемещений, о деятельности торговых предприятий. Также нередко возникает необходимость получения данных о взаимоотношениях с поставщиками или с филиалами, решать задачи управления персоналом или вести финансовый учет.

Рано или поздно возникают задачи автоматизации, которые могут отличаться только средствами и путями, ведущими к достижению конечного результата.

Достигшие в последнее десятилетие нового качественного уровня информационные технологии в ощутимой мере расширяют возможности эффективного управления, поскольку предоставляют новейшие методы как обработки, так и анализа информации, необходимой для принятия решений, в распоряжение различного рода сотрудников: управленцев, менеджеров, финансистов, маркетологов, руководителей производства всех рангов [2].

Помимо этого, посредством информационных систем специалисты расширяют свои профессиональные возможности. А также использование данных систем позволяет осуществлять деятельность хозяйствующего субъекта более рационально, целенаправленно и экономно, то есть более эффективно.

Выходом из данной ситуации является автоматизация и стандартизация управления отношений с клиентами. Иными словами, внедрение CRM-системы.

Дословно термин CRM-системы (англ. *Customer Relationship Management*) можно интерпретировать как «управление взаимоотношениями с клиентами».

Фактически, CRM – это программное обеспечение прикладного характера с множеством функций, предназначенное для автоматизации взаимоотношений между клиентами и предприятием.

CRM представляет собой грамотную стратегию, нацеленную на создание долговременных, прибыльных взаимоотношений с клиентами посредством понимания их индивидуальных потребностей [2].

CRM – это намного большее, чем просто интеллектуальная технология, решающая проблему наработки клиентской базы и ее расширения.

Данная система включает в себя глубокий синтез стратегического прогнозирования, корпоративного понимания природы ценности потребителя в условиях, когда в распоряжении большинства дистрибьюторских рынков имеется множество каналов сбыта [1].

Кроме того, система задействует средства управления информацией, разработанных под *CRM* приложения, вбирает в себя качественные операции, высокоэффективную исполнительность и сервис.

Обеспечение перехода от стратегии масс-маркетинга и массовых продаж к индивидуальным продажам, а также услугам с учетом индивидуальных требований отдельного потребителя входит в идеологию *CRM*.

По сути, *CRM* можно представить в виде пирамиды, которая состоит из трех пластов: вершина – стратегия компании по отношению к клиентам, средний пласт – люди и процессы, а фундаментом являются технологичные решения, задействованные в работе предприятия.

Программа *CRM* предоставляет возможность обеспечения результативного взаимодействия со следующими субъектами: поставщиками, посредниками, структурными подразделениями предприятия и, конечно же, с клиентами.

CRM может содержать информацию обо всех клиентах предприятия, которые хоть раз совершали с ним целевое действие: заказ, звонок, общение в чате и др. Специфика *CRM* позволяет автоматизировать значительную часть работы, которую сотрудники прежде делали вручную. Важен и тот факт, что система позволяет отслеживать и анализировать уже имеющиеся

отношения с клиентом, поскольку затраты на привлечение нового клиента на порядок выше затрат, которые будут нацелены на удержание имеющегося клиента.

При этом, управление взаимоотношениями с клиентами имеет сложную структуру, являясь многоплановым и продолжительным процессом, отражая, помимо прочего, реакцию на быстро меняющуюся бизнес-среду в условиях непрерывной модификации современной экономики.

В момент настройки системы руководитель компании принимает решение о специфике и направлении автоматизации. Различные отрасли системы приходят в эксплуатацию с учетом нужд предприятия. Владелец или руководитель компании, в свою очередь, от *CRM* получает контроль за оперативностью работы, инструмент планирования и прогнозирования, качественную и глубокую аналитику, систематизированную базу клиентов, дополнительные меры информационной безопасности.

Как было упомянуто выше, *CRM* имеет возможность проведения сбора, хранения и анализа информации о существующем или потенциальном клиенте.

Таким образом, данной системой может быть собрана следующая информация:

- реакции клиента на деловое предложение;
- мера удовлетворенности обслуживания;
- изменение предпочтений клиента;
- исполнение договоренностей;
- получаемый от клиента доход.

Ниже более подробно описаны задачи, которые обычно решает *CRM*-система.

В первую очередь – это формирование базы клиентов. С ее помощью менеджерам больше не требуется использовать ручной труд, занося клиентов в базу или пополняя их историю обращений и покупок. В карточке клиента

CRM формирует оповещение в момент звонка, визита сайта или обращения в мессенджер. Затем *CRM* добавляет информацию о просмотрах клиента, о конкретной сумме покупки, о возможных претензиях клиента. К карточке может прикладываться запись телефонного звонка, указываться имя и фамилия ответственного менеджера.

Также с помощью данной системы осуществляются звонки и заявки с сайта. При объединении *CRM* с сайтом и каналами коммуникаций обращения клиентов приходят в единую систему. Так менеджерам значительно проще уследить за заявками, обработать их, не упуская ничего.

CRM позволяет изменять статус заказов: автоматически переводить заказы в оплаченные – при совершении покупки клиентом или же в отмененные – при истечении срока оплаты. Программа отображает как отгруженные со складов заказы, так и полученные клиентами.

CRM-система способна автоматизировать документооборот. Если менеджер работает с *CRM*, необходимость тратить время на создание договоров пропадает. В *CRM*-программе они создаются посредством использования готовых шаблонов, автоматически подтягивая из карточек клиентов платежную и контактную информацию.

Кроме этого, *CRM* повышает рост продаж, лояльность клиентов, создавая единую клиентскую базу, позволяет грамотно управлять временем персонала.

Таким образом, управление взаимоотношениями с клиентами располагает некоторыми преимуществами, становясь прозрачным, планируемым и управляемым.

Таблица 1.1, представленная ниже, наглядно демонстрирует решение *CRM*-системами некоторых проблем, возникающих на предприятии.

					ПГУ 1.090301.01.001 ПЗ	Лист
Изм.	Лист	№ докум.№	Подпись	Дата		11

Таблица 1.1 – Роль CRM-систем

Описание проблемы	Роль CRM-системы
Потеря входящих заявок	Учет создаваемых контактов для сравнения с протоколом
Сотрудники забывают связаться с клиентом	Экономия времени сотрудника на планирование и руководителя на контроль «хвостов»
Эффективность работы сотрудника снижается	Экономия времени руководителя на определение/изъятие «бесперспективных» сотрудников
Уменьшение выпуска продукции с предприятия	Экономия времени на отчетность и упрощение работы над ошибками руководителя

Опираясь на вышеизложенное, можно заключить, что внедрение CRM-системы действительно приносит компании значительную пользу. Это происходит за счет:

- автоматизации работы персонала и предприятия в целом;
- избавления сотрудников от рутинных задач;
- быстрого формирования отчетов;
- соединения в единую систему множество отделов компании.

1.1.1 Анализ существующих CRM-систем

Необходимо провести обзор систем класса CRM для выявления преимуществ и необходимого функционала при разработке CRM-системы предприятия.

«Bitrix24» представляет собой мощный инструментарий, который способен организовывать работу компаний.

«Bitrix24» отличается комплексностью, с его помощью удобно управлять продажами и маркетинговыми кампаниями, а также данная система располагает возможностью настройки службы *HR*-поддержки. Включает возможность создавать бесконечное число групп пользователей и рабочих групп. Упрощает документооборот – позволяет централизовать хранение важных файлов.

Однако сервис работает довольно медленно, действия порой не вступают в силу сразу, так как после обновления страницы изменения на ней не отражаются.

«A2B» – продукт от отечественных разработчиков с простым, но разнообразным функционалом.

Система имеет следующие преимущества: быстрое внедрение, широкие возможности для доработок, а также самостоятельное администрирование.

Минусы: почтовый раздел имеет мало функций, не подойдет для использования в сложных штатах.

Рассмотрим «CRM Простой Бизнес». Сервис устроен для управления бухгалтерской и складской документацией.

Имеет следующие плюсы: легкость в управлении заказами, формировании отчетов; обладает телефонией, *SMS*-рассылками. «CRM Простой бизнес» является хорошим вариантом для розничных торговых точек, общепита, салонов красоты, фитнес – центров и прочего. Благодаря грамотной системе оповещений помогает персоналу своевременно закрывать поставленные задачи. В рамках сервиса удобно проводить и конференции, включая конференции для клиентов.

Но в «CRM Простой бизнес» отсутствует возможность интегрироваться с иными *SMS*-сервисами, а также для работы с системой требуется обучение.

«*atoCRM*» представляет собой простую и надежную *web* – программу. Отличается не перегруженным, простым интерфейсом, легка в освоении,

имеет удобный модуль для связи сотрудников. В данной системе прекрасно реализована телефония, присутствует возможность интеграции и внесения доработок через *API*.

Из минусов можно выделить некую избыточность, то есть данная система не подойдет компаниям, не ориентирующимся на массовые продажи. «*атоCRM*» не подходит и для работы с сотрудниками, управлением производством или разработкой продуктов, так как она рассчитана только на работу с клиентами и продажи.

«РосБизнесСофт» - российская *CRM*-система, предназначенная для средних и малых предприятий. Продукт нацелен на управление отношениями с клиентами, поставщиками, автоматизации производства, сервисного обслуживания, маркетинга и другого.

Основные возможности: работа с клиентами, управление продажами (счет сделок, выставление и отправка счетов, создание договоров и коммерческих предложений), управление складом (совершение необходимых операций по складу, отслеживание остатков товара через *CRM*, создание документации), управление сервисом, учет финансов, модуль маркетинга, интеграция в 1С.

Из недостатков можно выделить то, что в «РосБизнесСофт» необходимо подключать отдельно некоторые опции, система достаточно сложна в использовании. Иными словами, система отличается долгим внедрением. Функционал устроен таким образом, что фокус стоит на продажи и учет клиентов и в гораздо меньшей степени – на внутренние бизнес-процессы.

Достоинства и недостатки вышерассмотренных систем наглядно вынесены в таблицу 1.2.

					ПГУ 1.090301.01.001 ПЗ	Лист
						14
Изм.	Лист	№ докум.№	Подпись	Дата		

Таблица 1.2 – Обзор существующих CRM-систем

Название системы	Достоинства	Недостатки
<i>Bitrix24</i>	Комплексность. Возможность настройки HR-поддержки. Возможность создания бесконечного числа групп. Упрощенный документооборот.	Низкая скорость работы.
<i>A2B</i>	Простой и разнообразный функционал. Быстрое внедрение. Широкие возможности для доработок и администрирования.	Малое количество функций в почтовом разделе. Не подходит для использования в сложноустроенных компаниях.
<i>CRM</i> Простой Бизнес	Легкость в управлении заказами, формировании отчетов Наличие телефонии. Грамотная система оповещения. Удобство при проведении конференций.	Отсутствие возможности интеграции с другими сервисами. Требуется обучение пользователя.
<i>amoCRM</i>	Простой интерфейс, легкость в освоении. Наличие модуля для связи сотрудников. Удачно реализованная телефония. Присутствует возможность интеграции и внесения доработок через API.	Избыточность. Рассчитана только на работы с клиентами и продажи.

Продолжение таблицы 1.2

Название системы	Достоинства	Недостатки
РосБизнес Софт	Гибкая настройка прав доступа. Присутствует возможность интеграции через <i>API</i> . Наличие нескольких уровней настройки.	Сложность в использовании. Необходимость подключать отдельные опции. Долгое внедрение.

1.1.2 Постановка задачи

Учитывая достоинства и недостатки вышеописанных систем, а также особенности проблемы для создания *CRM*-системы предприятия были поставлены следующие задачи:

- просмотреть аналоги, шаблоны *CRM*-систем, провести сравнение и продумать наиболее удобные дизайн и функционал разрабатываемого приложения;
- изучить архитектуру клиент-серверных приложений, их типовые примеры и инструментарий;
- изучить сетевые модели передачи данных (*TCP/IP*);
- овладеть навыками создания архитектуры хранения данных. Проанализировать *MySQL*, *Microsoft SQL Server*, *Oracle*, *SQLite* и *PostgreSQL*, сравнить и выявить их преимущества;
- разработать функционал для взаимодействия сервера с клиентами;
- изучить основы разработки удобного и функционального интерфейса с помощью возможностей графической библиотеки;
- разработать интерфейс на графической форме содержимого *xml*-файлов;

- просмотреть наиболее популярные среды разработки. В качестве IDE использовать *Qt Creator* под операционной системой *Windows 10*;
- использовать язык программирования – *C++* с применением фреймворка *Qt5*;
- провести анализ выбранных инструментов разработки и сравнить их с существующими аналогами.

1.2 Архитектура приложения и выбор инструментов разработки

1.2.1 Выбор архитектуры приложения

С учетом требований к задаче была выбрана многоуровневая архитектура, разделяющая комплекс ПО на уровни по принципу «клиент-сервер».

Такие архитектуры обладают высокой масштабируемостью. Реализация *n*-уровневой системы, как правило, обходится дороже, однако обеспечивает высокую производительность. Именно поэтому такой вид архитектуры обычно применяется в крупных и комплексных программных решениях.

Данный подход сочетаем с современной сервис-ориентированной архитектурой для создания сложнейших моделей. Рекомендуется использовать его для сложных ПО, требующих производительности и масштабируемости, поскольку реализация чаще всего может оказаться дорогостоящей с точки зрения времени и ресурсов

Понятие «клиент-серверная архитектура» – сборная концепция, состоящая из двух комплементарных компонентов, которыми являются сервер и клиент.

Фактически, клиент и сервер – это программное обеспечение (ПО). На практике данные программы располагают на разных вычислительных машинах, которые взаимодействуют между собой через вычислительную

сеть, используя сетевые протоколы. Не исключено, что они могут быть расположены также и на одной машине.

В качестве клиента представлен локальный компьютер на стороне виртуального пользователя. С этой машины выполняется отправка запроса серверу для выполнения определенных системных действий или для возможности предоставления собственных данных.

Сервер, как правило, – это мощный компьютер или специальное системное оборудование. Он предназначен для разрешения определенного круга задач, возникающих в процессе выполнения программных кодов. Сервер может иметь в своем распоряжении различные функции: работать в качестве сервисного обслуживания по клиентским запросам, предоставлять пользователям доступ к определенным системным ресурсам, сохранять данные, используя БД. Одна программа-сервер имеет способность выполнять запросы от множества программ-клиентов, по этой причине часто её размещают на специально выделенной вычислительной машине, которую настраивают особым образом: обычно вместе с другими программами-серверами. Тогда производительность такой машины, как правило, высокая.

У клиент-серверной архитектуры можно выделить следующие преимущества:

- снижение технических требований к клиентским компьютерам снижаются, так как все вычисления выполняются на сервере;
- отсутствие дублирования кода программы-сервера программами-клиентами;
- упрощенная организация контроля полномочий на сервере, заключающаяся в разрешении доступа к данным только клиентам с соответствующими правами доступа, то есть все данные хранятся на сервере, который, как правило, защищен гораздо лучше большинства клиентов.

Однако выход из строя сервера может сделать неработоспособной всю вычислительную сеть. Также поддержка работы данной системы требует

отдельного специалиста, а именно системного администратора. Кроме этого, серверное оборудование обладает высокой стоимостью.

Необходимо учесть все минусы выбранной архитектуры для наиболее эффективных этапов проектирования и разработки.

1.2.2 Хранение данных

База данных (БД) — это упорядоченный набор структурированной информации или данных, хранящиеся обычно в электронном виде в компьютерной системе. База данных, как правило, управляется системой управления базами данных (СУБД).

Нормализация БД - преобразования отношений базы данных к виду, отвечающему нормальным формам. Цель нормализации: исключить избыточное дублирование данных, которое является причиной аномалий, возникших при добавлении, редактировании и удалении строк таблицы [3].

Проектирование БД осуществляется на основе трехуровневой архитектуры. Наглядно это представлено на рисунке 1.1.



Рисунок 1.1 - Трехуровневая архитектура БД

Самый верхний уровень – уровень внешних моделей данных. На нем каждая модель имеет свое «видение» данных. Точка зрения отдельных приложений на БД определяет именно данный уровень. Приложением же обрабатываются лишь необходимые ему данные.

Концептуальный уровень является центральным управляющим звеном. Именно на данном уровне база данных представляется в наиболее общем виде. Здесь происходит объединение данных, используемых всеми приложениями, работающими с данной БД. По сути, концептуальный уровень отражает обобщенную модель предметной области (объектов реального мира), для которой и создавалась база данных. Концептуальная модель также отражает лишь существенные особенности объектов реального мира, как и любая другая существующая модель [4].

Физический уровень – это расположенные в страничных структурах или файлах данные. При этом находятся они на внешних носителях информации. Эта архитектура обеспечивает два вида независимости при работе с данными: как логическую (между первыми двумя уровнями), так и физическую (между вторым и третьим уровнями).

Логическая независимость подразумевает под собой специфику изменения данных, а именно – возможность изменения логической структуры данных без корректировки прикладных программ.

Физическая независимость же обусловлена возможностью переноса информации между носителями с сохранением работоспособности программ, работающих с текущей базой данных.

Это именно то, чего не хватает в процессе использования файловых систем.

В нормализованной реляционной базе данных можно хранить факты без избыточного дублирования. При этом такая БД вследствие нормализации будет представлять собой огромное множество связанных таблиц и колонок. В таком случае будет наименований колонок и таблиц может быть

недостаточно для понимания содержащейся в них информации. Поэтому собранная в нормализованной реляционной базе данных информация, как правило, доступна к анализу только с участием специалистов-программистов.

Получение агрегированной информации из нормализованной базы данных часто требует выполнения операций соединения по многим таблицам, содержащим большое число записей. Такие запросы могут выполняться часами, нарушая тем самым нормальную работу подразделений организации [3].

Таким образом, характеристики нормализованной БД не позволяют использовать их для оперативного анализа непосредственно лицами, принимающими решения.

Следствием этого стало активное применение в информационно-аналитических системах организаций хранилищ данных. Хранилище данных – это предметно – ориентированный интегрированный, привязанный ко времени и неизменяемый набор данных, предназначенный для поддержки принятия стратегических решений. Хранилище обеспечивает поддержку бизнес – аналитики [3].

Рассмотрим основные характеристики хранилища данных, отличающие его от нормализованной реляционной базы данных.

Первое – это предметная ориентированность, предполагающая, что данные в хранилище объединяются в категории. Хранение данных зависит от областей, которые они описывают.

Интегрированность означает, что исходные данные для анализа не берутся напрямую из источников. Они извлекаются, проверяются, очищаются, унифицируются, чтобы удовлетворять требованиям аналитика, исследующего деятельность всего предприятия. Этот процесс позволяет повысить как качество данных аналитической системы, так и скорость выполнения аналитических запросов к хранилищу данных.

Следующая характеристика – это привязка ко времени. Она подразумевает «жесткую» привязку данных в хранилище к определенному периоду времени. Это позволяет анализировать тенденции в развитии бизнеса.

И последнее – это неизменяемость, означающая, что данные никогда не будут изменены при их попадании в определенный исторический слой хранилища. Стабильность данных способствует достоверности результатов аналитических запросов.

Основываясь на поставленных задачах, в качестве архитектуры хранения данных было выбрано хранилище данных логической модели «Звезда».

Схема «Звезда» имеет одну таблицу фактов и несколько таблиц измерений. Служит для поддержки многомерного представления содержащихся в ней данных. БД данной схемы включает таблицу фактов и ряд связанных с нею при помощи внешних ключей таблиц измерений. Таблицы измерений могут иметь составные первичные ключи и являются денормализованными. Это позволяет упростить восприятия информации пользователем: саму структуру данных и формулировку запросов, поскольку при обработке запросов уменьшается количество операций соединения таблиц. Таблицы измерений являются денормализованными.

Денормализация — это намеренное приведение структуры базы данных в состояние, не удовлетворяющее требованиям нормализации. Обычно денормализация проводится с целью ускорения операций чтения из базы за счет добавления избыточных данных. Основным ее назначением является ограничение межтабличных соединений в запросах [5].

Универсальную денормализованную структуру данных, которая бы обеспечивала высокую производительность при любом запросе, создать невозможно. По этой причине выбранная схема строится так, чтобы

обеспечить наивысшую производительность при выполнении одного самого важного запроса, либо для группы похожих запросов.

1.2.3 Взаимодействие клиента и сервера

При разработке клиент-серверного приложения, передача информации между клиентом и сервером должна осуществляться на основе протокола *TCP/IP*.

TCP/IP – это сетевая модель передачи данных, которые представлены в цифровом виде. Она описывает способ передачи данных от источника информации к получателю описано в источнике [6].

В этой модели предполагается прохождение информации через четыре уровня: прикладной уровень, транспортный уровень, межсетевой уровень и канальный уровень. Каждый уровень представлен правилом или протоколом передачи.

Наименование *TCP/IP* происходит от двух важнейших протоколов семейства — *Transmission Control Protocol (TCP)*, а также *Internet Protocol (IP)*.

Протокол *TCP* гарантирует эффективную передачу информации от источника к приемнику. Для этого необходимо организовать взаимодействие через данный протокол, чтобы обеспечить должную степень надежности. Для исполнения данной задачи взаимодействие будет осуществляться за счет такой абстракции как сокет.

Сокет (англ. *socket* – разъём) – название программного интерфейса для обеспечения обмена данными между процессами. Процессы при данном обмене могут исполняться на одной или различных ЭВМ, которые должны быть связаны между собой сетью. Сокет представляет собой абстрактный объект, воплощающий конечную точку клиент-серверного соединения.

Следует различать клиентские и серверные сокеты. В *TCP* как клиент, так и сервер могут отправлять и получать. Таким образом, различие

закljučается не в отправке и получении. Все дело в том, как создается соединение.

Сокет сервера создается для привязки к порту и прослушивания подключения от клиента. Таким образом, сервер просто ждет подключения.

Клиентский сокет создается для подключения к серверу прослушивания. Клиент инициирует соединение. Клиентские сокеты можно сравнить с конечными устройствами телефонной сети, тогда как серверные – с коммутаторами. В клиентском приложении могут использоваться только клиентские сокеты, в серверном же – как клиентские, так и серверные сокеты.

1.2.4 Выбор языка программирования

При написании клиентской и серверной программ предпочтительнее использовать один язык программирования для увеличения скорости и эффективности разработки.

Серверная и клиентская части должны обладать графическим интерфейсом для удобного взаимодействия с пользователями и администратором, иметь высокое быстродействие, а также разрабатываемые компоненты ПО должны быть относительно независимыми, особенно это важно для серверной части, взаимодействующей с базой данных. Необходима поддержка классов для повышения читабельности исходного кода. Для данных целей подходят языки, основанные на объектно-ориентированном подходе. Рассмотрим наиболее популярные и доступные из них.

Язык *Python* имеет открытый исходный код и большое сообщество программистов, читабельность и удобство кода, гибкость и масштабируемость языка, обладает простотой. Однако данный язык имеет низкую скорость, поскольку является интерпретируемым.

Java — язык высокого уровня, имеющий простой синтаксис. Поддерживает множество библиотек, имеет стандарт для корпоративных

вычислительных систем. Данный язык является многопоточным и не зависит от платформ. *Java* имеет следующие недостатки: платное коммерческое использование, сложность кода и низкую производительность.

C# включает множество уже готовых конструкций языка, шаблонов и библиотек, а также строгую типизацию. При этом некоторые готовые конструкции языка могут работать неоднозначно, что является немалым недостатком при разработке многоклассового клиент-серверного приложения. *C#* легко дизассемблируется, это означает, что код может быть получен и изучен сторонними программистами.

Таким образом, в качестве базового языка был выбран *C++*, для которого существует большой выбор компиляторов и опций сборки. *C++* имеет относительно чистую реализацию ООП, высокую скорость, универсальность (написанные на этом языке программы могут без проблем исполняться на любой платформе), широкую сферу применения, большое сообщество – язык постоянно обновляется, дополняется шаблонами и библиотеками. Помимо этого, по данному языку написано множество книг и самоучителей.

1.2.5 Выбор СУБД

Рассмотрим наиболее популярные СУБД.

MySQL — СУБД с открытым исходным кодом. Ее главными плюсами являются скорость, гибкость (поддерживается большое количество различных типов таблиц). Также эта СУБД бесплатна, включает простой интерфейс и может синхронизироваться с другими базами данных. Но она имеет низкую скорость разработки и недостаточную надежность.

Microsoft SQL Server – фирменная СУБД, разработанная *Microsoft*, отсюда и следует ее основной недостаток – зависимость от операционной системы. Однако *SQL Server* обеспечивает надежную защиту данных и простой интерфейс. Также из недостатков можно отметить высокую стоимость лицензии и повышенное потребление ресурсов.

PostgreSQL предоставляет немало возможностей, достаточно производительна и надежна, при этом данные характеристики улучшаются в каждой новой версии. Работает практически на всех ОС, включая *Windows Server*. Но данная СУБД имеет высокую стоимость и сложный процесс отладки ошибок, потребляет большой объем оперативной памяти.

Oracle предлагает лучшую производительность, надежность, масштабируемость и безопасность. Однако *Oracle* также является платной системой управления базами данных.

Для хранения данных была выбрана *SQLite* — встраиваемая кроссплатформенная БД, которая поддерживает наиболее полный набор команд *SQL*. Также *SQLite* доступна в исходных кодах.

К ее основным преимуществам можно отнести:

- открытость;
- бесплатность;
- *SQL*-совместимость;
- отсутствие необходимости в администрировании;
- простота подключения;
- при достаточном объеме ОЗУ – высокая производительность;
- возможность шифрования данных.

Кроме того, у *SQLite* имеются некоторые ограничения. Здесь отсутствуют отдельные элементы: триггеры, хранимые процедуры и т.д. Не предусмотрена совместная работа с данными, а также имеется лимит по объему хранимых данных.

Если понимать все ограничения, подходя разумно к оценке сложности задач, использование СУБД *SQLite* является отличным решением для операций с данными.

1.2.6 Выбор среды разработки

Рассмотрим наиболее популярные среды разработки: *Visual Studio*, *Eclipse* и *Qt Creator*.

Visual Studio – официальная, бесплатная, функциональная *IDE*, имеющая облачные хранилища. Однако данная среда довольно сложна в освоении, при переходе на платную версию нередко теряются настройки и нарушается работа корпоративного сервера.

Eclipse ориентирована преимущественно на разработку *Java*-приложений, но полезна и для кодов на *C#*. Имеет хороший компилятор, высокую функциональность, а также данная среда разработки бесплатна. Из недостатков: ненадежность и сложность.

Учитывая достоинства и недостатки рассмотренных сред, выбор пал на *Qt Creator*. *Qt Creator* — кроссплатформенная свободная *IDE* для разработки графических пользовательских интерфейсов (*GUI*) на *C*, *C++*, *JavaScript* и *QML*. Она включает в себя графический интерфейс отладчика, а также визуальные средства разработки интерфейса с использованием *QtWidgets* и *QML*. Основной задачей *Qt Creator* является упрощение разработки графических приложений, поддерживающихся на разных платформах, с помощью фреймворка *Qt*.

Данная библиотека является полностью объектно-ориентированной, что обеспечивает легкое расширение возможностей и создание новых компонентов. Она поддерживается на различных платформах: от *Windows 95/98/NT/2000* и *Linux* до *IBM AIX* и *SGI IRIX*. Существуют бесплатная и платная версии библиотеки *Qt*. Бесплатная версия (*Qt Free Edition*) предназначена только для *Unix/X11* и разработки программ с открытым исходным текстом. Для разработки коммерческих программ применяется *Qt Professional Edition*, которая и будет использоваться при разработке *CRM*-системы предприятия. В библиотеке *Qt* имеются несколько важных возможностей, которые реализованы в виде специальных классов *C++*:

- способность коммуникации между объектами с помощью сигналов и слотов;

- механизм создания свойств объектов, которые не поддерживает компилятор C++:

- поддержка событий и фильтров событий;

- перевод строк для поддержки интернационализации;

- поддержка внутренних таймеров (позволяют производить интеграцию многих задач для событийных *GUI*);

- организация внутреннего взаимодействия путем использования иерархических деревьев объектов, сопоставимых с генеалогическими деревьями;

- наличие "охраняемых" указателей (автоматически устанавливаются в *NULL* при удалении объекта, на который ведет ссылка) [7].

Язык программирования сам по себе не способен модифицироваться, вбирая в себя создающиеся нестандартные механизмы, которые возникают за счет появления новых современных требований к пользовательским интерфейсам. Решением данной проблемы как раз и служит реализации специальных классов с помощью *IDE*.

1.3 Процесс проектирования и разработки

Разработанное приложение состоит из серверной и клиентской частей, создающих единую систему, основанную на сети с использованием *TCP/IP* протокола. Для каждой части предусмотрен свой графический интерфейс. Полный листинг программы продемонстрирован в приложении А.

Серверная часть приложения должна функционировать на отдельном ПК. Клиентские приложения располагаются на одном или более устройствах. Пользователи предприятия осуществляют взаимодействие только с клиентской частью. Тогда как сервер имеет согласование с базой данных. В приложении Б продемонстрирована *UML*-диаграмма развертывания, отражающая архитектуру исполнения программы.

Работа данного приложения построена на основе запросов, которыми обмениваются клиент и сервер. Отправка запросов происходит после срабатывания определенных сигналов по действию пользователя. Статус данных сигналов отслеживается в клиентской части, где и формируются необходимые для передачи и дальнейшей корректной работы приложения данные. Далее на сервер поступают запросы с клиентской части. Только после этого клиенту обратно отправляются данные об успешном получении запроса или запрос вместе с основными данными для вывода в интерфейсе. В качестве служебных запросов выступают передаваемые данные типа *QString*. Эти данные также отражаются и в окне отладки клиентской и серверной частей. В приложении выделены следующие запросы: *getListen*, *saveLetter*, *deleteLetter*, *updateLetter*, *fileTransfer*, *vvLogPar*, *createLogPar*, *getlogParFull*, *getGyrnal*, *addGyrnal*, *addMesChat*, *getMesChat*, *getClientOnline*. Соответствие запросам действий, происходящих в программе, показано в таблице 1.3.

Перед началом работы клиентов необходимо произвести запуск сервера, данное действие осуществляется разработчиком или администратором. Далее пользователь должен совершить авторизацию. Затем начинается работа непосредственно в CRM-системе. На серверной части, в свою очередь, отслеживаются полученные данные и статус соединения.

1.3.1 Реализация интерфейса

Графический интерфейс сервера предназначен исключительно для удобства разработчика. Он располагается в функции *createMain()* (ее содержимое показано ниже) класса *SystemTray*, представляет собой форму, приведенную к табличному виду с помощью виджета *QGridLayout*.

В графическом окне сервера располагаются две кнопки (*QPushButton*) и поле для вывода текста (*QTextEdit*) – информации о взаимодействиях клиентов.

Таблица 1.3 – Выполняемые запросы по действию пользователя

Запрос	Описание действия
<i>getListen</i>	Получение информации
<i>saveLetter</i>	Сохранение документа
<i>deleteLetter</i>	Удаление документа
<i>updateLetter</i>	Обновление документа
<i>fileTransfer</i>	Загрузка файла
<i>vvLogPar</i>	Обработка введенных логина и пароля
<i>createLogPar</i>	Регистрация
<i>getlogParFull</i>	Получение всех пар логинов и паролей
<i>getGyrnal</i>	Получение журнала
<i>addGyrnal</i>	Добавление записи в журнал
<i>addMesChat</i>	Добавление сообщения
<i>getMesChat</i>	Получение всех сообщений
<i>getClientOnline</i>	Получение клиентов онлайн

Серверный интерфейс продемонстрирован на рисунке 1.2.

По первой кнопке осуществляется демонстрация всех зарегистрированных пользователей, то есть хранящихся в базе данных клиентов. Это происходит в слоте *slotOpenClient*. В нем создается виджет с кнопкой «Заккрыть», а также с инструментом для компоновки (*QGridLayout*) и с *QListWidget*, где демонстрируется список клиентов.

Вторая кнопка стартового окна серверного интерфейса просто отключает сервер, прекращая его работу.

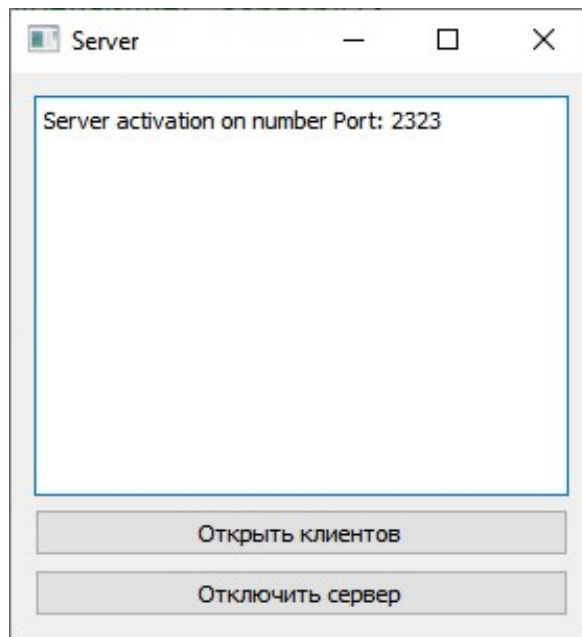


Рисунок 1.2 – Интерфейс серверной части

Работа клиента начинается с окна входа (рисунок 1.3).

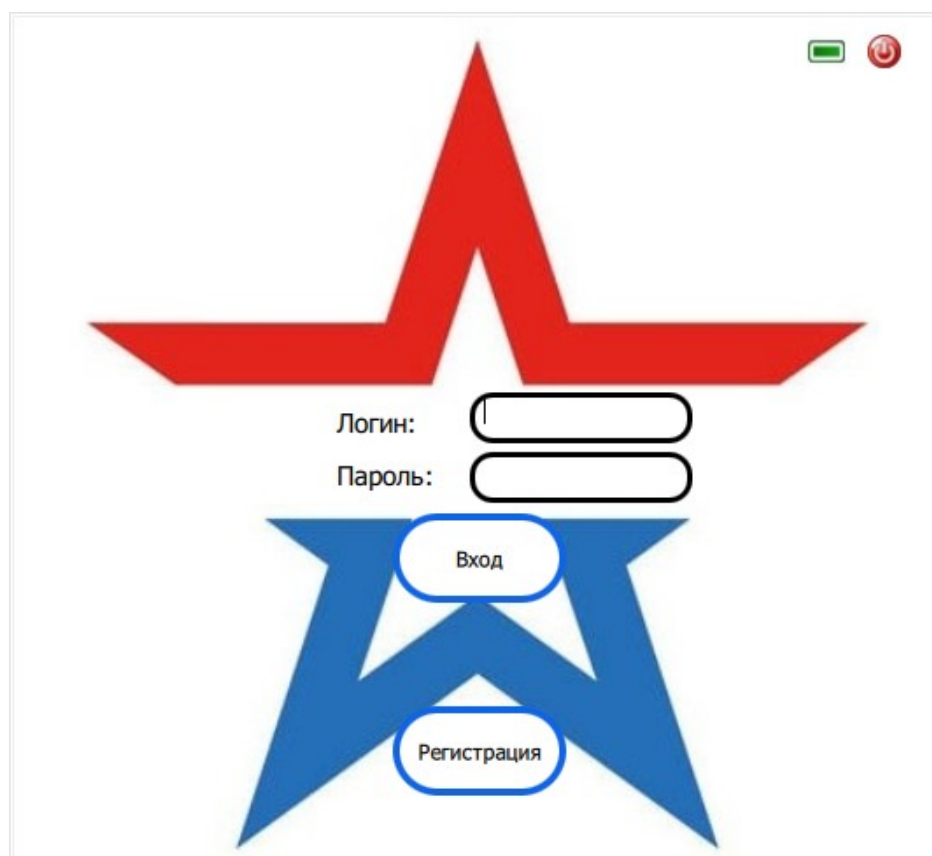


Рисунок 1.3 – Окно входа клиентской части

Реализация данного окна осуществляется посредством *QML* файла (*main.qml*), для создания которого использовался готовый *Qt* шаблон.

QML содержит в себе основное окно (*Window*), два поля для ввода логина и пароля пользователя (*TextInput*), кнопку «Вход» (*MouseArea*), а также поля *Rectangle* и импортируемые картинки (*Image*) для визуальной составляющей.

QML файл внедряется в клиентский класс *Login*, в который передаются поля ввода текста и кнопка. Для этого в конструкторе *Login* следующим образом создаются объекты класса *QObject*:

```
QDir dir(qApp->applicationDirPath() + "qrc:/main.qml");
QDebug() << dir.absolutePath();
QQmlApplicationEngine *eng = new
QQmlApplicationEngine(this);
QQmlComponent *comp = new QQmlComponent(eng, QUrl
("main.qml"));
QObject* pobj = comp->create();
```

Сначала, в первой строчке, происходит поиск необходимого файла, в котором хранится графическое описание окна входа. Затем данный файл вместе с его необходимыми компонентами загружаются в приложение.

При неверных действиях пользователя происходит вывод предупреждающий, информационных окон или окон ошибки, которые с помощью диалоговых окон *QMessageBox* или *QWidget*.

Окно регистрации нового пользователя (*QWidget*) продемонстрировано на рисунке 1.4.

Оно включает кнопку (*QPushButton*) «Создать пользователя», два виджета *QLabel*, являющимися «подсказками» пользователю, два поля *QLineEdit* для ввода логина и пароля, а также *QGridLayout* для упорядочивания.

Данные виджеты создаются с помощью кода, содержащегося в классе *Login*, без использования форм.

После авторизации у пользователя появляется возможность переключения между двумя вкладками. Переключение между ними реализовано в классе *Меню*.

Основные поля этого класса располагаются в пространстве *QHBoxLayout*, которое объявлено в коде, в отличие от других виджетов, реализованных с помощью *ui*-формы. *QHBoxLayout* обновляется по нажатию на соответствующие пункты *QListWidget*.

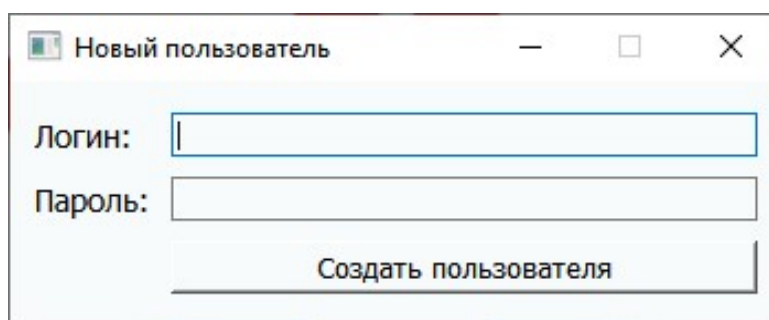


Рисунок 1.4 – Окно регистрации клиентской части

По нажатию на «Рабочее пространство» происходит вызов класса *Mail*, после клика на «Сообщения» вызывается класс *Chat*. Эти классы по нажатию на соответствующий пункт *QListWidgetItem* добавляются в *QHBoxLayout*.

Для начала рассмотрим вкладку «Рабочее пространство». Здесь пользователю предоставляется возможность создать и заполнить документ, а также совершить сопутствующие действия. Данное окно является стартовым, отображаясь по умолчанию после авторизации пользователя.

«Рабочее пространство» реализовано с помощью *ui*-формы (класс *Mail*), содержащей следующие элементы: *QHBoxLayout*, *QVBoxLayout*, 3 кнопки *QPushButton* (создание документа, удаление документа, открытие журнала), *QComboBox* для сортировки документов по алфавиту или дате изменения, *QListWidget* (для хранения списка документов), *QTableWidget* (здесь располагается собственно документ) и *QLineEdit* (сюда вводится название документа).

В названии формы класса *Menu* находится логин авторизованного пользователя, который на данный момент работает с программой.

QHBoxLayout используется для создания горизонтального ряда объектов, для которых применяется выравнивание. *QVBoxLayout* – для создания вертикального ряда. В данной форме присутствует пять таких компоновочных элементов в целях создания более гармоничного графического представления.

Рассмотрим подробнее тело документа с элементами, используемыми для его заполнения. Основа реализована с помощью *QTableWidget*, состоящей из трех столбцов: наименования пункта документа, пояснения к заполняемой ячейке и собственно вводимой информации, поступающей из базы данных посредством запросов. При этом заполняемые пользователем ячейки отображаются в 3 разных форматах: дата, номер (строковый тип данных) и кнопки (для открытия формы новых таблиц или загрузки файла). Тело документа (содержимое *QTableWidget*) реализовано в коде программы в отличие от других вышеописанных виджетов класса *Mail*. Часть таблицы данного класса, вызванного из класса *Menu*, представлена на рисунке 1.5.

Ниже представлен фрагмент кода, отражающий заполнения таблицы данными, поступившими от сервера:

```
columnsTable table = listG.at(num);  
ui->lineEdit->setText(table.name);  
QLineEdit *l = (QLineEdit *)ui->tableWidget->  
>cellWidget(0,2);  
l->setText(table.date01);  
ui->tableWidget->item(1,2)->setText(table.number01);
```

На поля для ввода дат наложена маска (*setInputMask("00-00-0000")*), добавляя удобство при заполнении документа. В случае ввода пользователем некорректных данных с последующей попыткой сохранения документа предусмотрен вывод об ошибке, а также очищение полей с некорректными

данными. Сохранение в таком случае невозможно. Это предохраняет БД от внесения в нее некорректных форматов.

Документы		Название документа
Сортировка: Дата изменения		Отчет Сидорова Н.Г.
Накладная за январь	Дата и номер составления заявки от кадровой службы	Дата: 01-07-2021 Номер: 1365
Отчет Сидорова Н.Г.	Дата получения заявки от кадровой службы	Дата: 12-07-2021
Конец апреля 2021	Номер резюме и дата его составления	Дата: 15-07-2021 Номер: 445
Результат проверки НТЦ	Номер заключения трудового договора и дата его заключения	Номер: 678 Дата: 23-08-2021
Новые сотрудники в мае 2022	Дата первого выхода сотрудника	Дата: 24-08-2021
Прибытие оборудования за 2020	Дата и номер заявки по выдаче технических средств	Дата: 01-09-2021 Номер: 32
Получение ПО июль 2021	Ответ сотрудника о получении технических средств	Номер: 121 Дата: 01-09-2021
Запланированные мероприятия на 2022		
Накладная за март 2022		
Получение оборудования за 2020		

Рисунок 1.5 – Содержимое документа

После нажатия на кнопку «Периферия ввода» происходит открытие нового окна, представленного на рисунке 1.6. Код слота, реагирующего на сигнал по данной кнопке:

```
int numb = ui->listWidget->currentRow();
columnsTable table = listG.at(numb);
fs = new FullSpisok(0, numb, ui->tableWidget->item(24,2)->text());
connect(fs, &FullSpisok::signSave, this,
&Mail::saveSignSlots);
```

Как видно по последней строчке, программа отслеживает сигнал *signSave* из класса *Fullspisok*

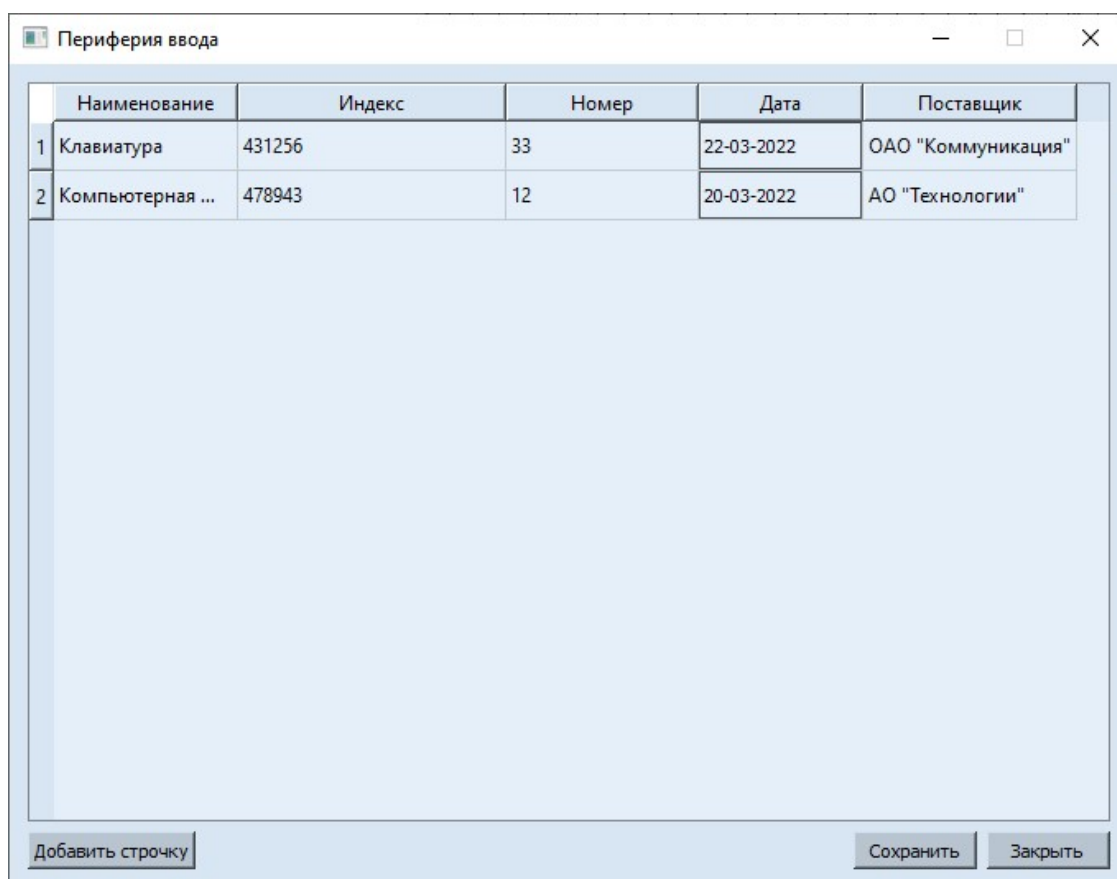


Рисунок 1.6 – Таблица «Периферия ввода»

По срабатыванию данного сигнала происходит добавление записи в таблицу окна «Периферия ввода» следующим образом:

```
ui->tableWidget->item(24,2)->setText(words)
```

Рассматриваемое окно «Периферия ввода» реализовано с помощью виджета *QWidget*, содержащего внутри себя таблицу и три кнопки («Добавить строку», «Сохранить», «Заккрыть» с одноименными возможностями). Они также связаны с соответствующими слотами сигналами нажатия.

Таблица класса *QTableWidget*, составляющая основу рассматриваемого окна, содержит пять столбцов, четыре из которых имеют строковый тип данных и один – *QDate*, преобразованный в дальнейшем в строковый тип данных для проверки корректного ввода даты. Аналогичным образом осуществлен интерфейс таблицы «Периферия вывода». Виджет,

появляющийся после нажатия на кнопку «Мероприятия», эквивалентен предыдущим двум с единственным отличием: таблица имеет только один столбец строковых типов данных. Реализация этих трех таблиц расположена в классе *FullSpisok*.

Вернемся к основному окну заполнения документа. В данной форме также присутствует возможность просмотреть журнал изменений по кнопке «Журнал» (рисунок 1.7).

	Дата и Время	Логин	ID документа	Операция
17	15:40 27-04-2022	Starodubova D.I.	7	Изменил
18	15:41 27-04-2022	Starodubova D.I.	9	Создание
19	15:44 27-04-2022	Gulyakova N.D.	10	Изменение
20	15:45 27-04-2022	Gulyakova N.D.	10	Создание
21	15:45 27-04-2022	Gulyakova N.D.	11	Создание
22	15:45 27-04-2022	Gulyakova N.D.	11	Изменение
23	15:45 27-04-2022	Gulyakova N.D.	11	Удаление
24	15:45 27-04-2022	Alpatova A.A.	12	Изменение
25	15:45 27-04-2022	Alpatova A.A.	12	Удаление
26	15:45 27-04-2022	Alpatova A.A.	5	Изменение
27	15:46 27-04-2022	Speranskiy D.D.	8	Изменение
28	15:46 27-04-2022	Speranskiy D.D.	10	Создание
29	15:46 27-04-2022	Speranskiy D.D.	11	Удаление
30	15:46 27-04-2022	Goryinin A.M.	10	Создание
31	15:47 27-04-2022	Alpatova A.A.	11	Создание
32	15:47 27-04-2022	Nazarova U.A.	11	Удаление
33	15:47 27-04-2022	Nazarova U.A.	12	Удаление
34	15:47 27-04-2022	Nazarova U.A.	6	Изменение
35	15:48 27-04-2022	Astafiev V.I.	4	Изменение
36	15:48 27-04-2022	Astafiev V.I.	9	Изменение

Рисунок 1.7 – Табличное окно «Журнал»

Данное окно в своей основе имеет отдельную *ui*-форму, содержащую одну кнопку «Заккрыть» и таблицу. Таблица включает в себя четыре столбца: три строковых типа («Дата и Время», «Логин», «Операция») и один целочисленный («ID документа»).

Вставка и заполнение этой таблицы происходит следующим образом:

```
ui->tableWidget->setRowCount(list.size());
    if(list.size() != 0) {
        for(int i = 0; i < ui->tableWidget->rowCount();
i++)
            for(int j = 0; j < ui->tableWidget-
>columnCount(); j++)
                {
                    QTableWidgetItem *view = new
QTableWidgetItem();
                    if(j != 2) {
                        view->setFlags(view->
flags() & ~Qt::ItemIsEditable);
                        view-
>setTextAlignment(Qt::AlignJustify);
                        view-
>setTextAlignment(Qt::AlignCenter);
                    }
                    ui->tableWidget->setItem(i,j,view);
                }
        for(int i = 0 ; i < ui->tableWidget->rowCount();
i++)
            {
                ui->tableWidget->item(i,0)->
setText(list.at(i).dateTime);
                ui->tableWidget->item(i,1)->
setText(list.at(i).login);
                ui->tableWidget->item(i,2)->
setText(QString::number(list.at(i).uidRecord));
```

```

        ui->tableWidget->item(i, 3)->
setText(list.at(i).operation); } }

```

Информация с данных столбцов отправляется в класс *Client* посредством сигналов и слотов.

Теперь рассмотрим вкладку «Сообщения» (рисунок 1.8). Как было упомянуто выше, реализация данной формы находится в классе *Chat*. В правой части предоставленного окна находится список клиентов (*QListWidget*). Логины авторизированных пользователей и статус их соединения с сервером передаются из класса *Client*, отвечающего за клиент-серверное взаимодействие. Пользователи «в сети» отмечаются белым.

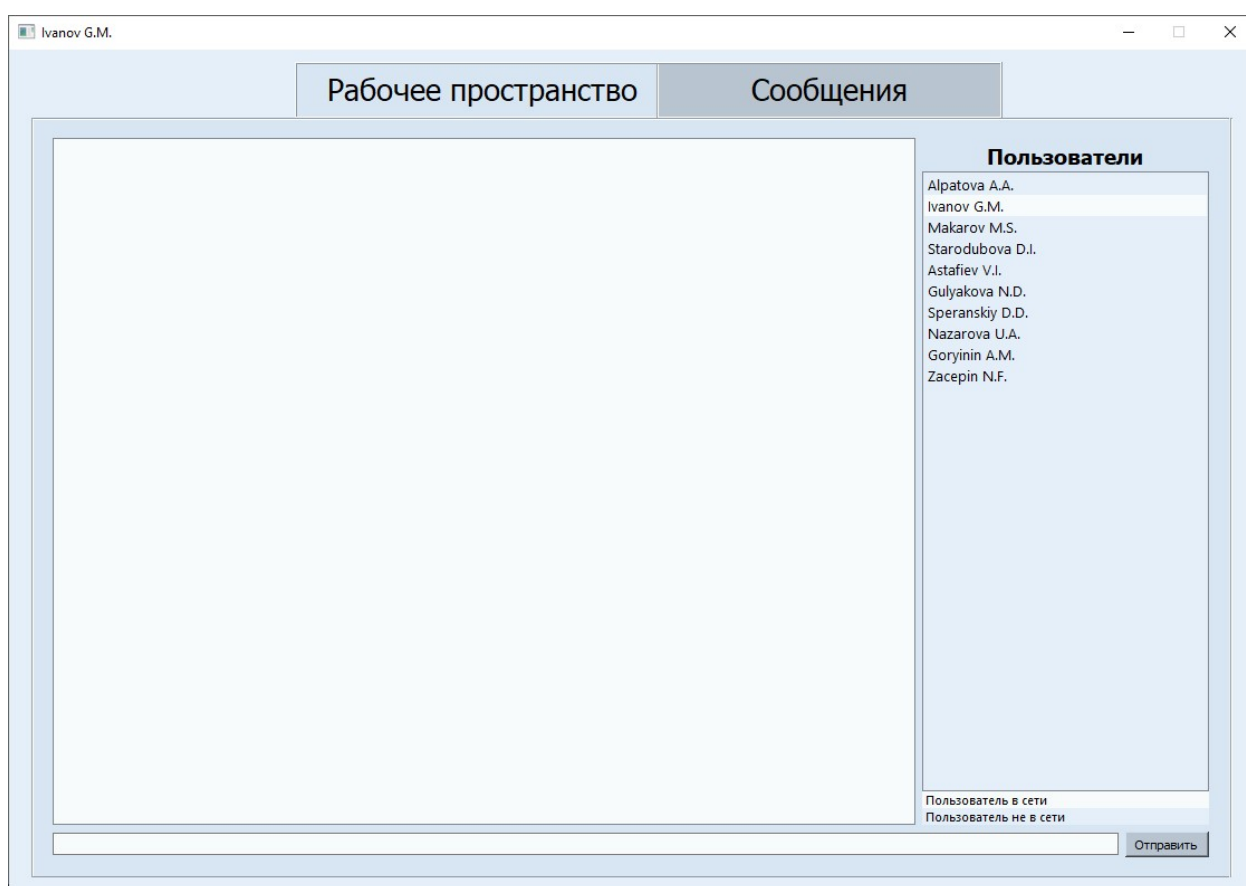


Рисунок 1.8 – Вкладка «Сообщения»

В левом основном виджете (*QListWidget*) располагаются сообщения с логином их отправителя. После ввода текста в *QLineEdit* посредством

сигнально-слотовой связи появляется сообщение в формате «логин пользователя: сообщение».

Таким образом, графический интерфейс клиентской части расположен в классах *FullSpisok*, *Gyrnal*, *Login*, *Mail*, *Menu*. Серверной – в *SystemTray*.

1.3.2 Проектирование и разработка серверной части приложения

Для реализации сервера используется класс *QTcpServer*, предназначенный для управления входящими *TCP*-соединениями.

В первую очередь, создается объект сервера (объект класса *SystemTray*) в основном классе *Main*. Здесь в объект класса в качестве параметра передается номер порта, по которому будет происходить соединение. Объект встроенного класса (*QTcpServer*) *m_server* является основой управления сервером.

Запуск сервера начинается по вызову метода *listen()*, в который передаются номер хоста (*QHostAddress::Any*) и порта. При возникновении ошибки работы данного метода сервер прекращает свою работу. В обратном случае происходит вызов слота *slotNewConnection()* по сигналу *newConnection()*, который отправляется при каждом соединении нового клиента. Для подтверждения соединения с клиентом вызывается метод *nextPendingConnection()*, возвращающий сокет, посредством которого можно осуществить дальнейшую связь с клиентом [7].

Работа сокета обеспечивается сигнально-слотовыми связями. Сигнал *disconnected()*, отправляемый сокетом при отсоединении клиента, соединяется со слотом *deleteLater()*, предназначенный для его последующего уничтожения. При отправлении клиентских запросов вызывается слот *slotReadClient()*, реагирующий на сигнал *readyRead()*.

Рассмотрим подробнее слот *slotReadClient()*. В его функционал входит чтение служебных запросов от клиента и необходимых данных. Сначала в данном слоте выполняется преобразование указателя, возвращаемого методом *sender()*, к типу *QTcpSocket*:

```

QTcpSocket* pClientSocket = (QTcpSocket*)sender();
QDataStream in(pClientSocket);
in.setByteOrder(QDataStream::BigEndian);

```

Если размер доступных для чтения данных больше или равен размеру блока, тогда из потока считываются необходимые данные. Это достигается путем передачи целого числа (размера блока) в начале каждого сообщения.

В методе *sendToClient()* формируются данные, которые отсылаются клиенту (класс *SystemTray*). Он используется для отправки клиенту сообщений о результате работы сервера (для формирования и обработки вышеупомянутых подразделов запросов) и отправки данных из БД или от другого клиента.

При помощи структур класса *DBClaims* сервер взаимодействует с базой данных. Потоки каждой таблицы базы данных считываются и записываются посредством операторов потока, которые объявлены как члены класса.

Рассмотрим данную реализацию в программе (запись полей основного документа):

```

QDataStream &operator>>(QDataStream &s,
tagcolumnsTable &c) {
    s >> c.uid >> c.name >> c.date01 >> c.number01 >>
c.date02 >> c.date03 >> c.number03 >> c.date04 >>
c.number04 >> c.date05 >> c.date06 >> c.number06 >>
c.date07 >> c.number07 >> c.number08 >> c.tz09 >>
c.date10 >> c.number10 >> c.riperf10 >> c.rireason10 >>
c.date11 >> c.number12 >> c.date13 >> c.number13 >>
c.whommade13 >> c.whomshow13 >> c.product14 >>
c.detail15 >> c.date16 >> c.date17 >> c.defectreason18
>> c.date19 >> c.riperf20 >> c.rireason20 >> c.date20 >>
c.number20 >> c.date201 >> c.number201 >> c.actreason21
>> c.date22 >> c.number22;
    return s; }

```

Чтение и запись в другие таблицы происходит аналогичным образом.

Ниже, на рисунке 1.9, представлена схема деятельности серверной части в рамках сетевого соединения.

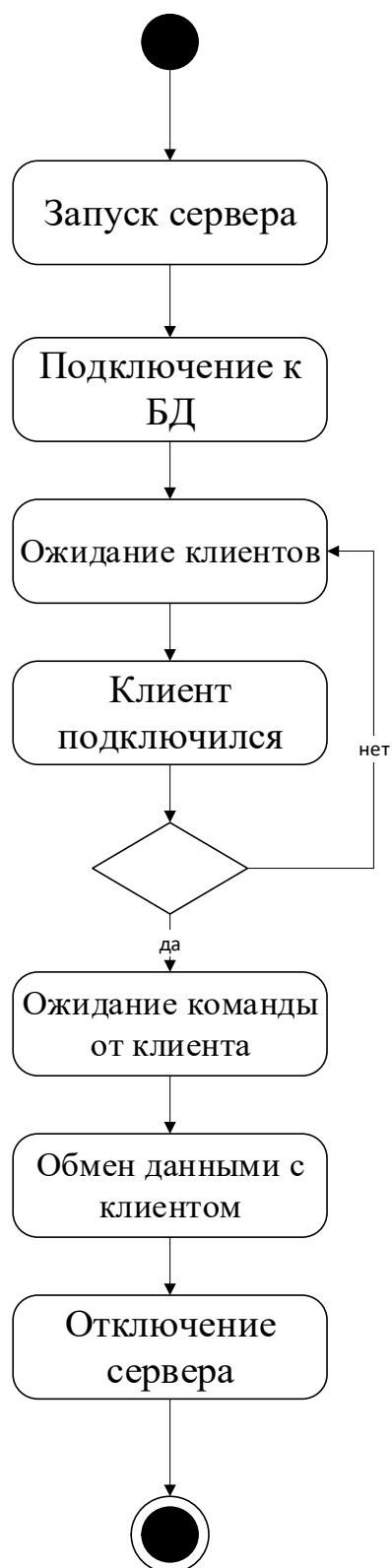


Рисунок 1.9 – UML-диаграмма деятельности сервера

Помимо графической части слот *slotOpenClient* содержит в себе и создание объектов структур класса базы данных (*Dbclaims*), использование которых необходимо для получения логинов клиентов из БД и отслеживания клиентов «в сети»:

```
QListWidget *listUsers = new QListWidget();
QList<columnsTableLogPass> listCTLP;
columnsTableLogPass cTLP;
listCTLP = db.selectAllRecordsInLogPass();
listUsers->clear();
for (int i=0; i < listCTLP.size(); i++){
    cTLP = listCTLP.at(i);
    listUsers->addItem(cTLP.login);
}
```

Также в данной части важно упомянуть функции *transferFile* и вызываемую в нем функцию *getName*.

Они используются для поиска по каталогам, открытия файла и получения его имени при прикреплении файла к документу и сохранении его в базе данных.

В приложении Б находится *UML*-диаграмма классов сервера.

1.3.3 Проектирование и разработка БД

База данных состоит из трех сущностей, логическая модель которых показана на рисунке 1.10.

Здесь выделены следующие сущности:

- Пользователь;
- Журнал
- Документ.

Таблица Документ связана с таблицей Журнал по коду документа.

Таблица Пользователь связана с таблицей Журнал внешним ключом – кодом пользователя.

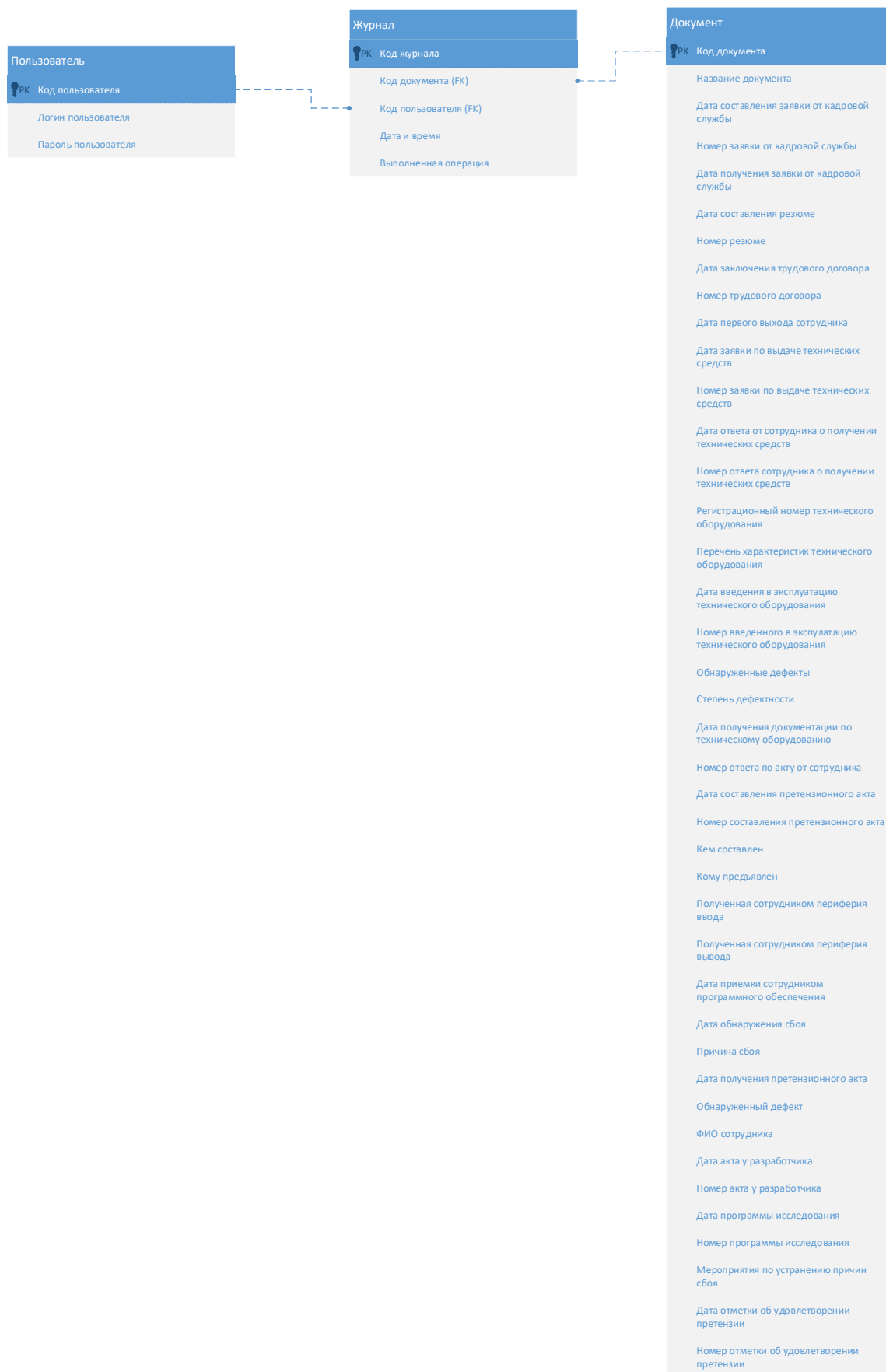


Рисунок 1.10 – Логическая модель данных

Название таблиц БД и соответствующие им атрибуты представлены в таблице 1.4. Таблица Документ занесена не полностью в силу присутствия в ней множества однотипных атрибутов. Их полный перечень продемонстрирован на рисунках 1.10 и 1.11.

Таблица 1.4 – Сущности и их атрибуты

Название таблицы	Описание	Атрибуты	Первичный ключ
Пользователь	Код пользователя	<i>uid</i>	<i>uid</i>
	Логин пользователя	<i>login</i>	
	Пароль пользователя	<i>password</i>	
Журнал	Код журнала	<i>uid</i>	<i>uid</i>
	Дата и время	<i>datetime</i>	
	Код пользователя	<i>uidUser</i>	
	Код документа	<i>uidRecords</i>	
	Выполненная операция	<i>operation</i>	
Документ	Код документа	<i>uid</i>	<i>uid</i>
	Название документа	<i>name</i>	
	Дата получения заявки от кадровой службы	<i>date01</i>	
	Номер заявки от кадровой службы	<i>number01</i>	
	Дата получения заявки от кадровой службы	<i>date02</i>	

Таким образом, БД состоит из трех денормализованных таблиц, которые образуют недоработанную схему «Звезда» с центральной таблицей фактов – Журнал.

Реализация базы данных располагается в серверной программе в классе *DBClaims*.

Для подключения базы данных в проектный файл была добавлена строка *QT += sql*.

Для работы с классами этого модуля используется библиотека *QtSql* и последние два уровня классов этого модуля: программный уровень (классы *QSqlDatabase*, *QSqlQuery*, *QSqlError*, *QSqlField*, *QSqlIndex* и *QSqlRecord*) и уровень пользовательского интерфейса (классы *QSqlQueryModel*, *QSqlTableModel* и *QSqlRelationalTableModel*).

В классе *DBClaims* используются следующие публичные слоты для работы с базой данных:

- *addRecordInTable()* – добавление записи в БД;
- *dbIsOpen()* – проверка на открытие БД;
- *removeRecordInTable()* – удаление записи из БД;
- *selectIdAllRecordsInTables()* – чтение идентификационных номеров всех записей из БД;
- *selectAllRecordsInTable()* – чтение всех записей из БД;
- *selectRecordInTables()* – чтение записи из БД по идентификационному номеру;
- *updateRecordInTable()* – обновление записи в БД;
- *addRecordInLogPass()* – добавление записи логин – пароль;
- *selectIdAllRecordsInTableLogPass()* – чтение идентификационных номеров всех записей из таблицы *logpass*;
- *removeRecordInLogPass(columnsTableLogPass)* – удаление записи в таблице *logpass*;
- *updateRecordInLogPass()* – обновление записи в таблице *logpass*;
- *selectAllRecordsInLogPass()* – чтение всех записей в таблице *logpass*;
- *selectRecordInLogPass()* – чтение записи из таблицы *logpass* по идентификационному номеру;

- *addRecordInMess()* – добавление записи в таблицу *message*;
- *selectIdAllRecordsIntableMess()* – чтение идентификационных номеров всех записей из таблицы *message*;
- *selectAllRecordsInMess()* – чтение идентификационных номеров всех записей из таблицы *message*.

Приватные слоты:

- *createDB()* – создание БД;
- *createTables()* – создание таблиц;
- *slotClearStructTables()* – очистка структуры БД;
- *dataToStringInsert()* и *dataToStringUpdate()* используются для преобразования дат в строковый тип данных;
- *stringlistToStruct*, *stringlistToStructLogPass*, *stringlistToStructMess* – перевод таблиц из строкового типа данных в структуры.
- *structToStringlist*, *structToStringlistLogPass*, *structToStringlistMess* – перевод таблиц из структур в строковый тип данных.

Передача и получение данных от сервера происходит с помощью трех структур:

- *tagcolumnsTableLogPass* – взаимодействие с таблицей «Пользователь»;
- *tagcolumnsTable* – взаимодействие с таблицей «Документ»;
- *tagcolumnsTableMess* – взаимодействие с таблицей «Журанл».

Структура *tagcolumnsTableMes*, показанная ниже, предназначена для передачи данных о пользователе: идентификационный уникальный номер пользователя, его логин и пароль. В базе данных таблица, связанная с данной структурой, носит название *LogPass* (Пользователь в логической модели).

```
typedef struct tagcolumnsTableLogPass
{
    int      uid;
    QString login;
```

```

        QString password;
    } columnsTableLogPass;

```

Структура *tagcolumnsTableMes* предназначена для взаимодействия сервера с таблицей БД Журнал или *Message* в физической модели данных. Состоит из следующих полей:: идентификационный номер (UID), дата взаимодействия с документом, UID пользователя и собственно операция, совершенная с документом.

```

typedef struct tagcolumnsTableMess
{
    int      uid;
    QString  dateTime;
    QString  login;
    int      uidRecord;
    QString  operation;
} columnsTableMess;

```

tagcolumnsTable – структура для обмена данными документа, заполняемого пользователем. Все члены данной структуры, кроме *UID* записи (типа *int primary key*), имеют тип данных *varchar(255)*. Ниже представлен фрагмент *tagcolumnsTable*. В базе данных – это таблица *ListClaims* (Документ в логической модели).

```

typedef struct tagcolumnsTable
{
    int uid;
    QString name;
    QString date01;
    QString number01;
    QString date02;
    QString date03;
    QString number03;
    QString date04;
    ...
    QString tz09;
}

```

```

QString date10;
QString number10;
QString riperf10;
QString rireason10;
...
} columnsTable;

```

Создание таблицы происходит в функции *createDB()*. Для этого, в первую очередь, активизируется драйвер при помощи вызова статического метода *QSqlDatabase::addDatabase()*. В него передается идентификатор драйвера используемой СУБД – *QSQLITE*.

Далее в метод *setDatabaseName()* передается имя базы данных, являющееся названием файла, в котором хранятся данные.

Соединение осуществляет посредством вызова метода *open()*. Затем происходит проверка на существование файла и его открытие.

При ошибочном варианте присоединения базы данных происходит вывод информации в окно отладки. Отслеживание происходит методом *lastError()*.

На этом проверка работы БД завершается, также завершается проверка *CRUD sql*-операций.

Исполнение *SQL* команд реализовано в функции *createRecordTables()*. Здесь используется класс *QSqlQuery*. Запросы в виде строки передаются в метод *exec()*.

Рассмотрим пример запроса по добавлению записи в базу данных, а именно таблицу *ListClaims* посредством использования структуры *tagcolumnsTable* (фрагмент метода *addRecordInTable()*).

```

columnsTable ct;

ct = _ct;
if(li.contains(ct.uid) || (ct.uid <= 0))
{
    if(li.isEmpty())

```

```

        {
            ct.uid = 1;
        }
        else
        {
            ct.uid = li.last() + 1;
        }
    }

    QString stringTempNames;
    stringTempNames.clear();
    stringTempNames += "(" + constNamesColumnsTable[0];
    for(int i = 1; i < constCountColumnsTable; i++)
        stringTempNames += " , " +
constNamesColumnsTable[i];
    stringTempNames += ")";
    QString stringTempData;
    stringTempData.clear();
    QStringList sl = structToStringlist(ct);
    stringTempData += tr("(%1").arg(sl.at(0));
    for(int i = 1; i < sl.size(); i++)
        stringTempData += tr(", '%1'").arg(sl.at(i));
    stringTempData += tr(")");
    QString queryRequestString = "";
    queryRequestString.clear();
    queryRequestString =
        tr("INSERT INTO %1 %2 VALUES %3")
        .arg(constNameTable1)
        .arg(stringTempNames)
        .arg(stringTempData);
    QSqlQuery query(db);
    rez = query.exec(tr("(%1").arg(queryRequestString));

```

Здесь увеличивается номер новой записи перед занесением таблицы.
 Далее происходит добавление новой записи в формате: имя таблицы (в

данном случае *listclaims*), хранящееся в строке *stringTempNames*; имя записи, хранящееся в *stringTempNames*, а именно название документа, введенное пользователем, время добавления записи и номер (эти данные предварительно преобразовываются в строковый тип данных); и собственно сама запись, преобразованная предварительно в строковый тип данных для передачи данных из строки *stringTempData*.

Помимо выше рассмотренного запроса (на добавление записи в таблицу) в программе используются запросы на удаление записи, чтение записей, и их обновление. Эти запросы осуществляются для всех трех таблицы базы данных.

Таблицы, составляющие базу данных, и их взаимодействие показаны на рисунке 1.11.

Таким образом, база данных состоит из трех таблиц: *LogPas*, *Message*, *ListClaims*.

Таблица *LogPas* содержит четыре поля: *uid* – уникальный код пользователя (типа *int primary key*); *login* – логин пользователя и *password* – пароль пользователя (типа *varchar(255)*).

Таблица *Message* состоит из пяти полей: *uid* – уникальный код журнальной записи (типа *int primary key*); *datetime* – дата и время целевого действия с документом (типа *varchar(255)*); *uidUser* – идентификационный номер пользователя (типа *int primary key*); *uidRecord* (типа *int*) – идентификационный номер документа; *operation* – выполненная операция с документом (типа *varchar(255)*). В данной таблице два внешних ключа *uidUser* и *uidRecord* для связей с таблицами *LogPas* и *ListClaims* соответственно.

Таблица *ListClaims* включает в себя 41 поле, где первое поле *uid* – уникальный идентификационный номер документа (типа *int primary key*), остальные сорок полей типа *varchar(255)* – данные по заполнению документа.

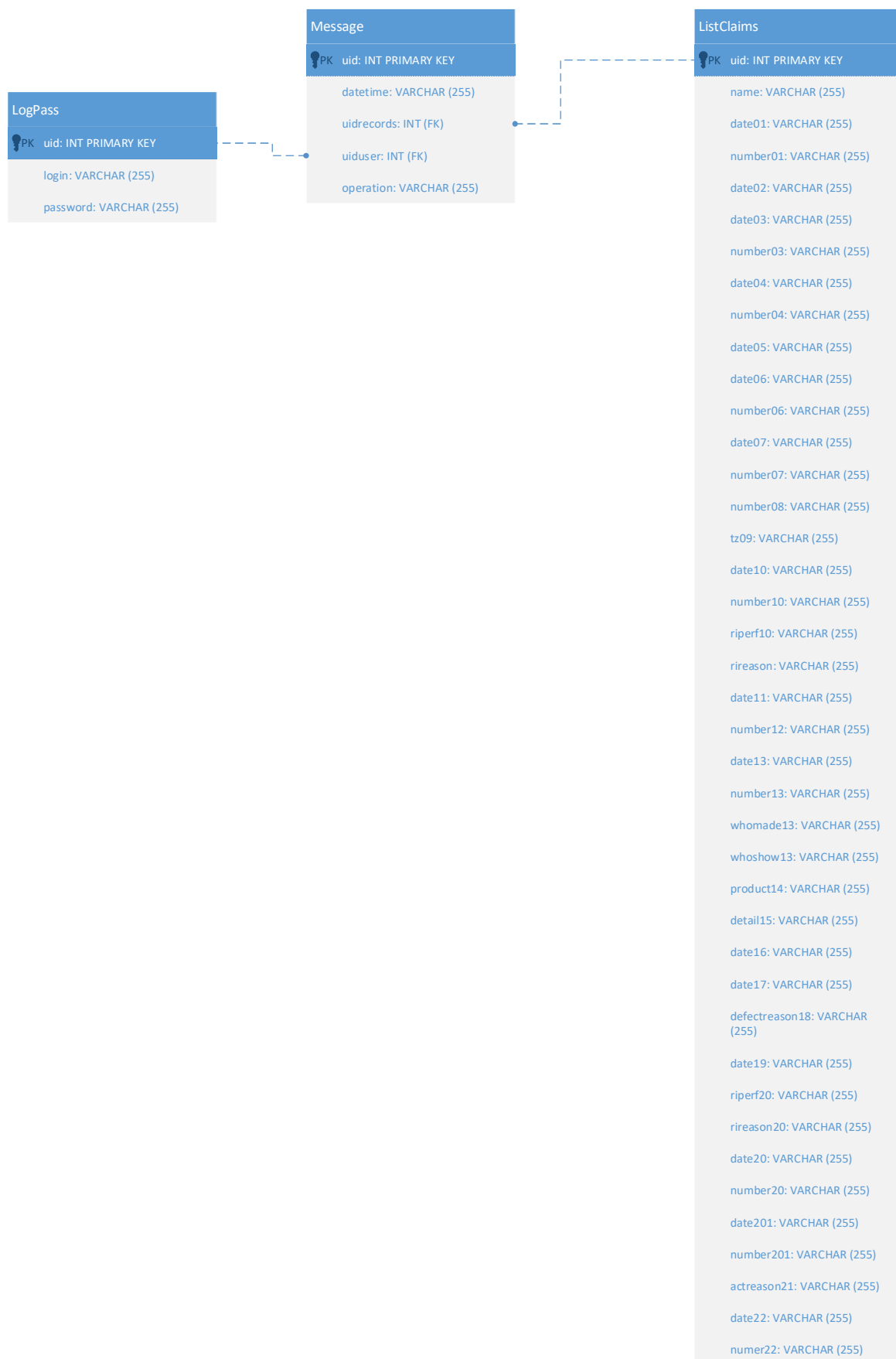


Рисунок 1.11– Физическая модель данных

Некоторые из них: *name* – название документа; *date01*, *date02...date22* – даты по некоторым целевым действиям в рамках предприятия; *number01*, *number02... number22* – соответствующие данным датам номера фиксируемых документов; *tz09* - перечень характеристик технического оборудования в виде загружаемого файла; *riperfl0* – обнаруженные дефекты, *rireason10* – интенсивность дефектности.

ListClaims связана своим первичным ключом *uid* с таблицей *Message* полем *uidrecords*, являющимся внешним ключом данной таблицы. Также *Message* полем *uidUser* (типа *int*) связана с таблицей *LogPass*.

В используемой недоработанной схеме «Звезда» *Message* – это таблица фактов, оставшиеся *LogPass* и *ListClaims* – таблицы измерений.

В клиентской части предусмотрена проверка на ввод пользователем соответствующих данных, что исключает избыточности и неоднозначности при широком использовании типа *varchar(255)* в базе данных.

К тому же, в интерфейсе предусмотрен интуитивно понятный формат для ввода дат. При вводе пользователем данных неверного формата пользователю сообщается об этом, а сохранение в таком случае невозможно. Это наглядно продемонстрировано в разделе 1.3.1, реализация же рассмотрена в разделе 1.3.4.

Все данные БД расположены в файле *dbclaims.db*, находящейся в папке сборки серверного проекта. Каждое поле хранится в исходном виде за исключением паролей – они проходят процедуру хеширования, описанную подробнее в разделе 3.4. При регистрации в базу данных заносится хешированный пароль, далее при каждом входе пароль также проходит процесс хеширования и сравнивается с имеющимися паролями базы данных.

1.3.4 Проектирование и разработка клиентской части приложения

Основная логика клиентской части взаимодействия с сервером и передачи данных расположена в классе *Client*.

Для реализации клиента создан объект класса *QTcpSocket* в функции *initClient*, которая вызывается из конструктора. После этого происходит вызов метода *connectToHost()*, в который передаются IP-адрес компьютера и порт сервера. Данный метод устанавливает связь с сервером [7].

Далее с помощью сигналов происходит отслеживание статуса сокета. Предусмотрено четыре варианта поведения сокета после создания. Соответствие сигнала и слота, описанного в клиентском классе, рассмотрено в таблице 1.5.

Таблица 1.5 – Сигнально-слотовая связь клиентского сокета

Сигнал	Слот
<i>connected()</i> (соединение успешно установлено)	<i>slotConnected()</i>
<i>readyRead()</i> (готовность к чтению новых данных)	<i>slotReadyRead()</i>
<i>disconnected()</i> (произошло отсоединение от сервера)	<i>slotDisconnected()</i>
<i>error()</i> (произошла ошибка)	<i>slotError()</i>

Сокет отправляет сигнал *connected()* при создании соединения. И сигнал *readyRead()* – при готовности предоставить данные для чтения. В случае возникновения ошибок отправляется сигнал *error()*.

Если клиент отсоединяется от сервера, то вызывается сигнал *disconnected()*. Состояние подключения в этих слотах отслеживается с помощью булевой переменной *connected*, а также информация о сокете отражается в окне отладки.

slotError вызывается при возникновении ошибок. В нем происходит преобразование кода ошибки в текст, отображаемый в окне отладки.

slotReadyRead() вызывается при поступлении данных от сервера для считывания информации.

Чтение и запись данных из сокета осуществляется при помощи объекта потока данных *QDataStream*. При этом, в начале каждого блока сообщений в поток отправляется переменная типа *quint64*, хранящая размер передаваемых данных. В качестве отсылаемых и принимаемых данных может быть как служебный запрос, так и запрос и получаемые от сервера данные для отображения пользователю.

Аналогично методу *sendToClient()*, находящемуся в серверной части, используется слот *slotSendToServer()*. Он осуществляет запрос к серверу. Согласно сигнально-слотовым связям, данный метод вызывается по нажатию кнопок или при переходе из изменяемой ячейки.

Данные документа, отправляемые серверу, записываются в структуры *tagcolumnsTable*, а также *tagcolumnsTableMess* (журнал). Эти поля заполняются пользователем и отправляются на сервер для хранения в базе данных. К структуре *tagcolumnsTableLogPass* из основного клиентского класса *Client* происходит обращение функции *getLogParFull* класса *Chat* для вывода уже имеющихся сообщений пользователя (вкладка «Сообщения») другим клиентам и получения клиентов, находящихся в сети. Логин текущего пользователя передается в конструктор класса *Chat* из класса для авторизации *Login*, что необходимо для вывода клиентов.

Ниже, на рисунке 1.12, представлена UML-диаграмма деятельности клиентской части в рамках сетевого соединения.

Также в классе *Client* предусмотрено хеширование паролей посредством встроенного в *Qt* класса *QCryptographicHash*. Хеширование происходит в методах *createLogPar()* (отправка пользовательских логина и пароля при регистрации) и *vvLogPar()* (отправка логина и пароля при входе).

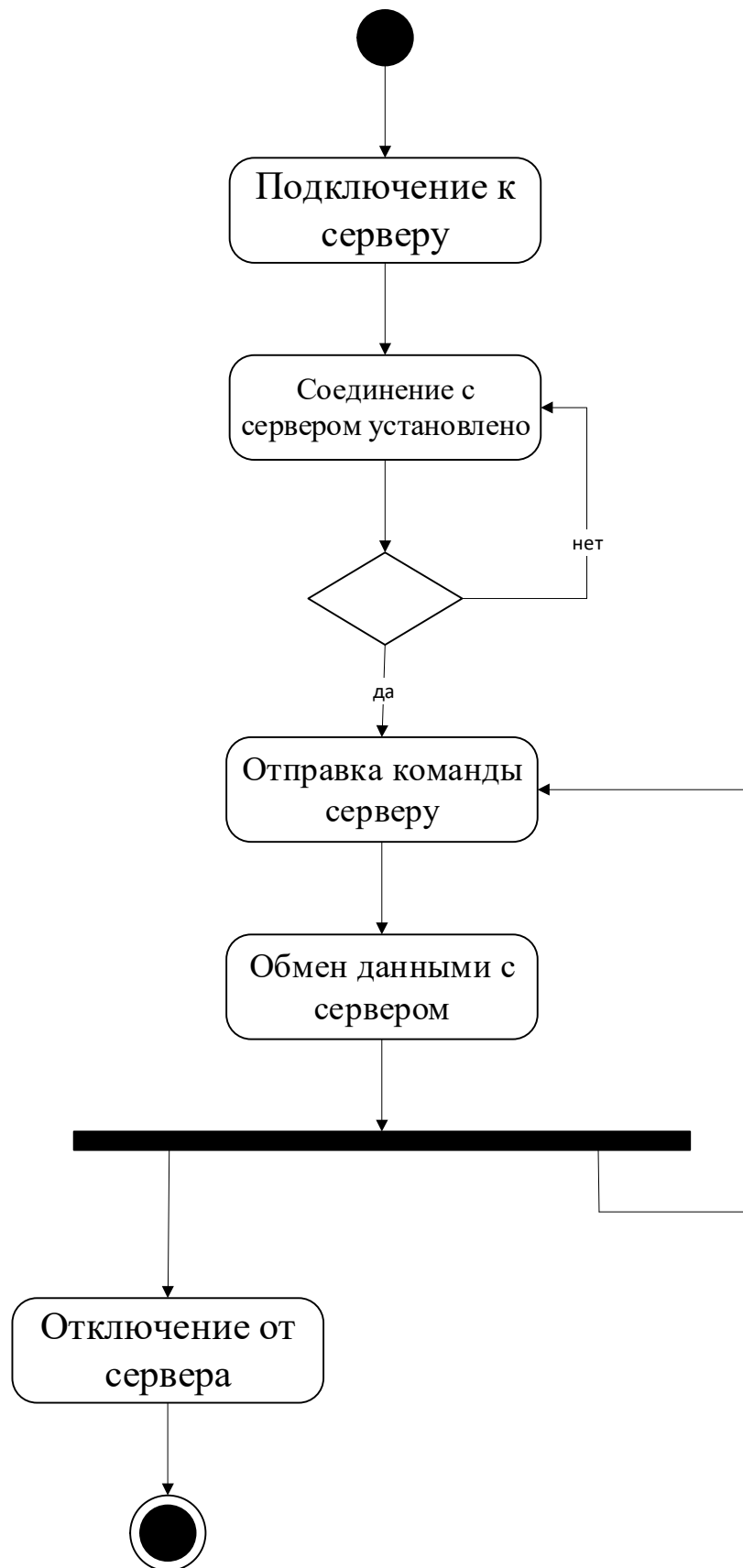


Рисунок 1.12 – UML-диаграмма деятельности клиента

Фрагмент кода, реализующий хеширование и отправку на сервер пароля:

```
QByteArray parolByte = parol.toUtf8();
parol =
QCryptographicHash::hash(parolByte,
QCryptographicHash::Sha256);
out << quint64(0) << QString::fromUtf8("createLogPar") <<
login << parol;
```

Здесь *parol* – переменная типа *QString*, прочтенная из текстового поля. Для проведения хеширования используется метод *hash()*, являющийся экземпляром класса *QCryptographicHash*. Перед этим значение из строковой переменной преобразуется в значение типа *QByteArray*, для отправки же пароля на сервер происходит обратное действие.

Помимо взаимодействия с сервером, в данном разделе стоит упомянуть о реализации проверки на некорректный ввод пользователя в основной документ. После нажатия на кнопку «Сохранить» программой отслеживается сигнал *on_pushButton_5_clicked()*. Номера должны состоять только из цифр, остальные символы исключаются. Данная проверка осуществляется путем использования встроенной функции *isDigit()*. Если в строке обнаруживается посторонний символ, появляется окно *QMessageBox*, а некорректные поля очищаются следующим образом:

```
ui->tableWidget->item(i, 2)->setText("")
```

Аналогичным образом происходит и проверка формата поля с датой с помощью наложения маски «*dd-MM-yyuu*» с последующим использованием встроенной функции *isValid()*. В приложении Б находится *UML*-диаграмма классов клиента.

1.4 Отладка и тестирование

Приложение отлажено с использованием среды программирования *Qt Creator* и *SQLiteStudio*.

Отладка проводилась с использованием построчного прохождения программы, точек останова, просмотра и анализа содержимого стеков, переменных, регистров. Также в процессе написания кода был использован инструмент *qDebug()*, позволяющий фиксировать данные на выводе приложений.

Тестирование осуществлялось в рабочем порядке, в процессе разработки, а также после завершения написания программ.

В ходе тестирования были выявлены и устранены проблемы, касающиеся синтаксиса *SQL*-запросов, передачи данных между клиентом и сервером, взаимодействия сервера с базой данных.

Логически приложение можно разделить на три части: авторизация, взаимодействия пользователя со вкладкой «Рабочее пространство», взаимодействие со вкладкой «Сообщения».

Наглядно полный набор возможностей пользователя продемонстрирован в приложении Б (*UML*-диаграмма вариантов использования).

Также на рисунке 1.13 показана диаграмма последовательности действий в приложении сотрудника предприятия. Здесь синий цвет – это взаимодействие клиента с сервером; фиолетовый – этап авторизации; серый и голубой – разноуровневые действия с журналом; желтый – обращение с вкладками перехода по двум областям приложения; коричневый относится к работе на вкладке «Сообщения»; красный – действия с документом; зеленый – редактирование документа; желтый – работа с таблицами, входящими в документ.

1.4.1 Тестирование окна авторизации

Пользователь начинает работу с приложением с окна входа (рисунок 1.3). При вводе некорректных данных с последующим нажатием на кнопку «Вход» в программе предусмотрена система оповещений об ошибках.

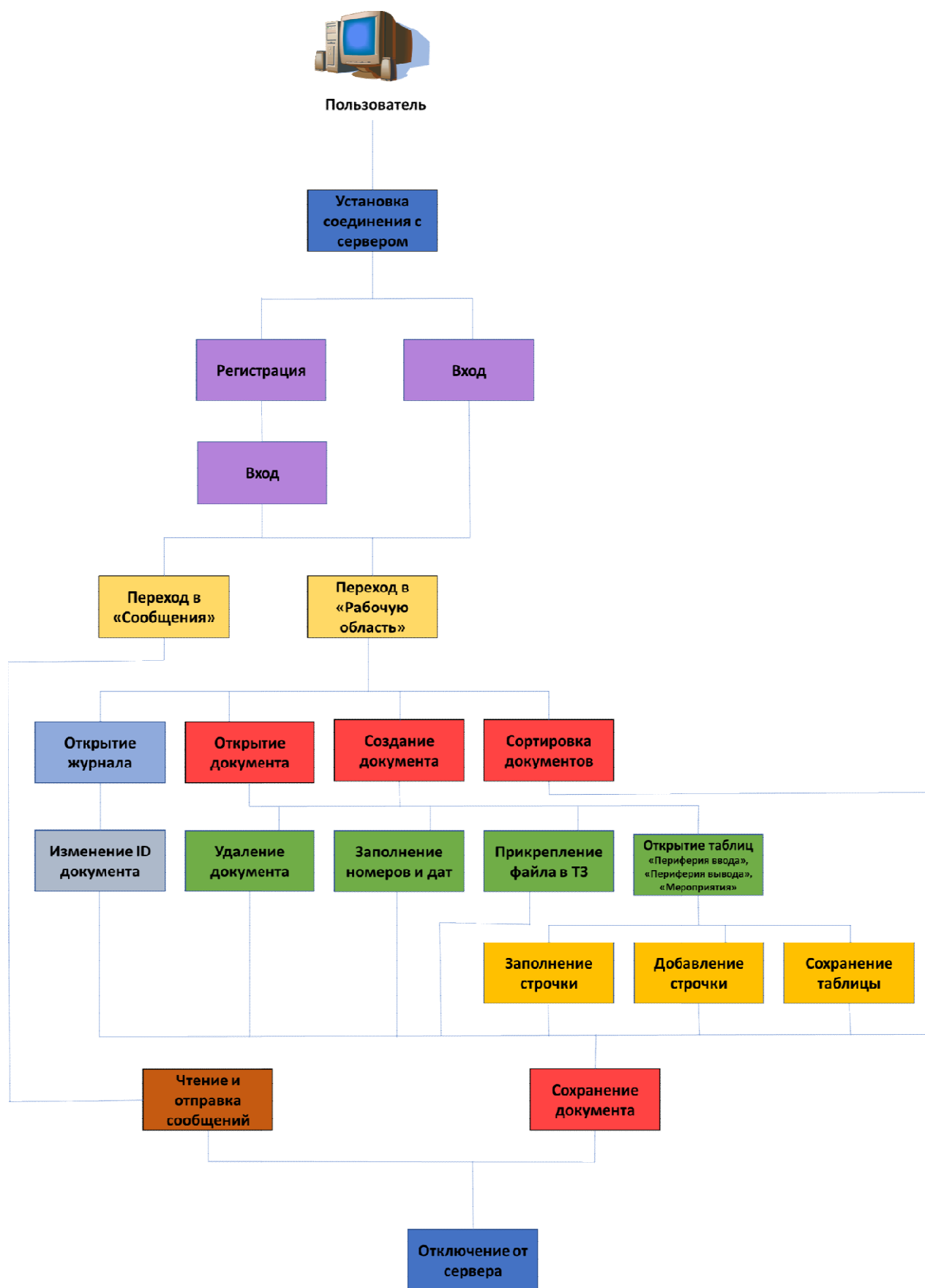


Рисунок 1.13 – Диаграмма последовательности действий в приложении

В таблице 1.6 представлены реакции окна входа на определенные действия пользователя с учетом полученного результата на соответствующих рисунках.

Таблица 1.6 – Реакция программы на действия пользователя при входе

Действие пользователя	Ожидаемый результат	Полученный результат на рисунках
Вход с пустыми полями логин - пароль	Оповещение об ошибке	Рисунок 1.14
Вход с пустым полем для логина	Оповещение об ошибке	Рисунок 1.15
Вход с пустым полем для пароля	Оповещение об ошибке	Рисунок 1.16
Вход с неверным паролем	Оповещение об ошибке	Рисунок 1.17
Вход с несуществующим логином	Оповещение предупреждение	– Рисунок 1.18
Вход существующей пары логин - пароль	Успешная авторизация, переход в основное окно программы	Рисунок 1.19

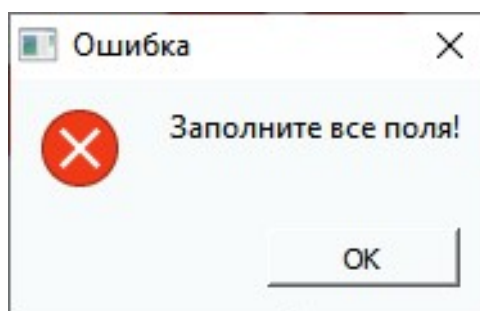


Рисунок 1.14 – Вход с пустыми полями логин-пароль

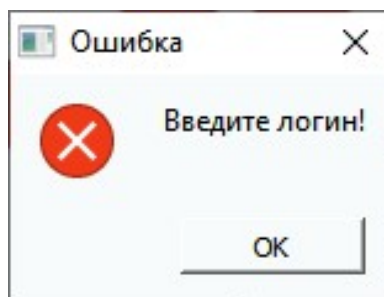


Рисунок 1.15 – Вход с пустым полем для логина

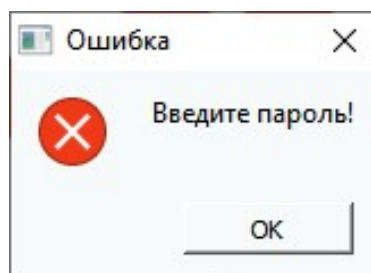


Рисунок 1.16 – Вход с пустым полем для пароля

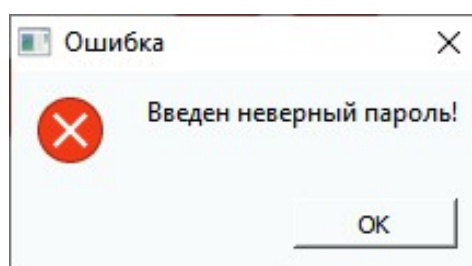


Рисунок 1.17 – Вход с неверным паролем

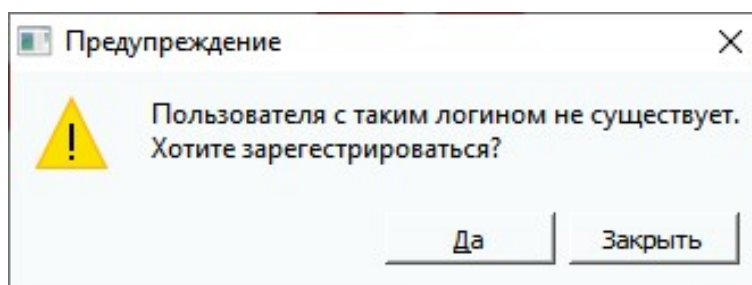


Рисунок 1.18 – Вход с несуществующим логином

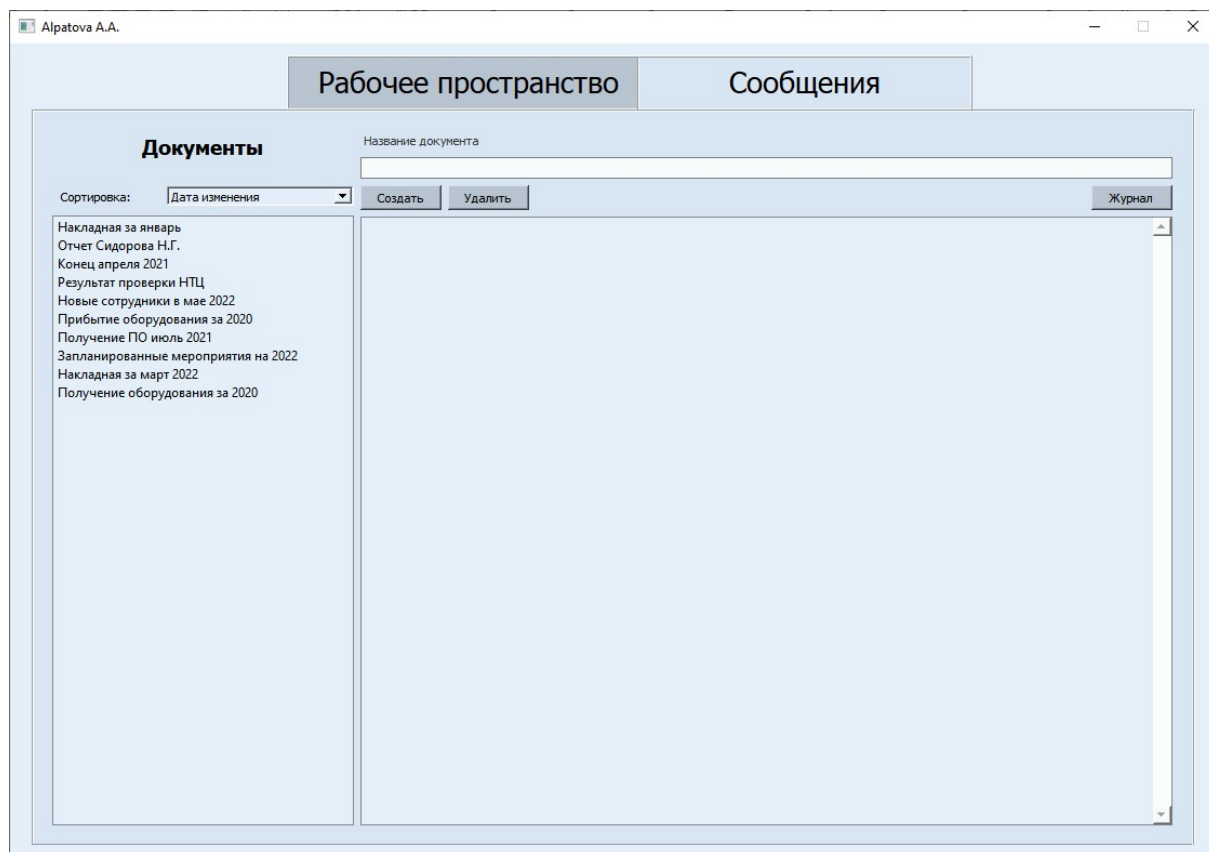


Рисунок 1.19 – Переход в основное окно программы

Далее после нажатия на кнопку «Регистрация» или нажатии на кнопку «Да» в окне предупреждения (рисунок 1.18) пользователь переходит на окно регистрации (рисунок 1.20).

Рисунок 1.20 – Окно регистрации

Здесь также при оставлении пустых полей пользователь будет оповещен об ошибках (рисунки 1.14 – 1.15). При вводе существующего в

базе данных логина также выходит окно – оповещение об ошибке (рисунок 1.21).

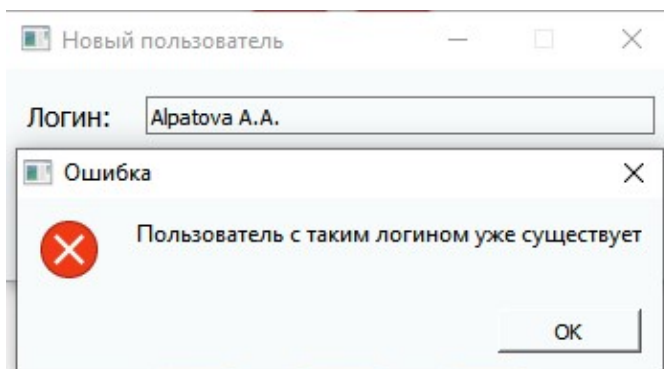


Рисунок 1.21 – Ошибка при регистрации

Когда регистрация завершена успешно, об этом сообщается пользователю (рисунок 1.22).

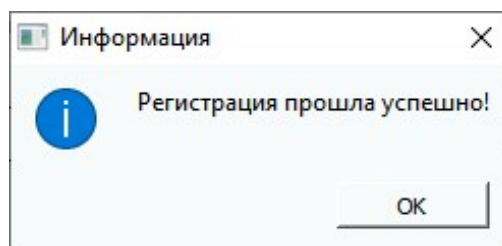


Рисунок 1.22 – Успешная регистрации

Таким образом, полученный результат тестирования совпал с ожидаемым.

После успешной авторизации начинается работа с основным окном приложения, которое, как было упомянуто выше, состоит из вкладок «Рабочее пространство» и «Сообщения».

1.4.2 Тестирование рабочего пространства

Интерфейс приложения устроен таким образом, что работа начинается именно с вкладки «Рабочее пространство». Заголовок формы – логин текущего пользователя. В левой части находятся документы, которые можно отсортировать как по дате изменения, так и по алфавиту. Каждый

пользователь имеет возможность изменить уже существующий документ, удалить его или создать новый. Справа – само тело документа. Создание и процесс заполнения документа продемонстрированы на рисунке 1.23.

Рисунок 1.23 – Тело нового документа

После создания документ необходимо сохранить, нажав соответствующую кнопку. После сохранения документ (его название) отобразится слева в поле «Документы».

В случае, когда пользователь переходит к другому документу или сортирует документы без сохранения текущего документа, программа предупреждает об этом и предоставляет возможность сохранить или выйти без сохранений (рисунок 1.24).

По кнопке «Журнал» можно просмотреть действия каждого пользователя над документами (рисунок 1.25).

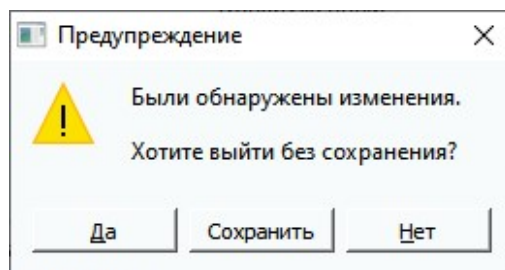


Рисунок 1.24 – Выход без сохранения

Журнал				
	Дата и Время	Логин	ID документа	Операция
19	15:44 27-04-2022	Gulyakova N.D.	10	Изменение
20	15:45 27-04-2022	Gulyakova N.D.	10	Создание
21	15:45 27-04-2022	Gulyakova N.D.	11	Создание
22	15:45 27-04-2022	Gulyakova N.D.	11	Изменение
23	15:45 27-04-2022	Gulyakova N.D.	11	Удаление
24	15:45 27-04-2022	Alpatova A.A.	12	Изменение
25	15:45 27-04-2022	Alpatova A.A.	12	Удаление
26	15:45 27-04-2022	Alpatova A.A.	5	Изменение
27	15:46 27-04-2022	Speranskiy D.D.	8	Изменение
28	15:46 27-04-2022	Speranskiy D.D.	10	Создание
29	15:46 27-04-2022	Speranskiy D.D.	11	Удаление
30	15:46 27-04-2022	Goryinin A.M.	10	Создание
31	15:47 27-04-2022	Alpatova A.A.	11	Создание
32	15:47 27-04-2022	Nazarova U.A.	11	Удаление
33	15:47 27-04-2022	Nazarova U.A.	12	Удаление
34	15:47 27-04-2022	Nazarova U.A.	6	Изменение
35	15:48 27-04-2022	Astafiev V.I.	4	Изменение
36	15:48 27-04-2022	Astafiev V.I.	9	Изменение
37	15:48 27-04-2022	Astafiev V.I.	1	Изменение
38	09:26 05-05-2022	Alpatova A.A.	9	Изменение

Рисунок 1.25 – Журнал

Рассмотрим подробнее возможные ошибки пользователя при работе с документом и реакцию программы на них.

В полях «Номер» должны вводиться только цифры, в иных случаях после попытки сохранения пользователь оповещается о некорректных данных (рисунок 1.26), поле очищается. Помимо этого, программа так же реагирует и на внесение дат неверного формата.

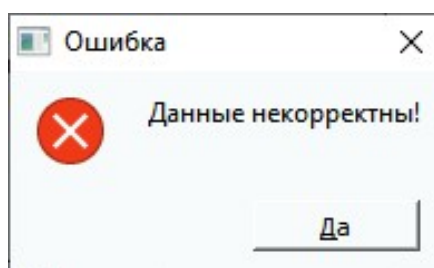


Рисунок 1.26 – Ошибка при вводе некорректных данных

К каждому документу можно прикрепить файл с перечнем характеристик оборудования (рисунок 1.27), а также заполнить три табличных окна внутри документа: «Периферия ввода» (рисунок 1.28), «Периферия вывода» (полностью аналогичная таблице «Периферия ввода») и «Мероприятия» (рисунок 1.29). В данных окошках добавляются и заполняются необходимые строки. Здесь также предусмотрены система контроля некорректных значений и предохранения от выхода без сохранения по аналогии с основной таблицей.

Регистрационный номер технического оборудования	Номер	
Перечень характеристик технического оборудования	Характеристики	Открыть файл

Рисунок 1.27 – Поле с открытием файла

Итак, в данном подразделе были рассмотрены и протестированы возможности пользователя на основной вкладке «Рабочее пространство». Ожидаемые результаты совпали с реальными.

Периферия ввода

	Наименование	Индекс	Номер	Дата	Поставщик
1	Клавиатура	431256	33	22-03-2022	ОАО "Коммуникация"
2	Компьютерная ...	478943	12	20-03-2022	АО "Технологии"

Добавить строčku Сохранить Закрýть

Рисунок 1.28 – Табличное окно «Периферия ввода»

Мероприятия

	Мероприятия
1	Выявление области сбоя
2	Выявление условий проявления сбоя
3	Анализ ПО на совместимость

Добавить строčku Сохранить Закрýть

Рисунок 1.29 – Табличное окно «Мероприятия»

1.4.3 Тестирование сообщений

После нажатия на вкладку «Сообщения» пользователь переходит в область чата сотрудников.

Здесь справа показаны зарегистрированные в программе пользователи, при этом пользователи онлайн выделены светлым фоном, о чем также сообщается на интерфейсе снизу от списка пользователей.

Основная часть – область ранее написанных сообщений. Чтобы отправить сообщение в общий чат, пользователь должен ввести текст в поле в самом низу формы и нажать кнопку «Отправить».

На рисунке 1.30 продемонстрирована данная вкладка после отправки сообщения.

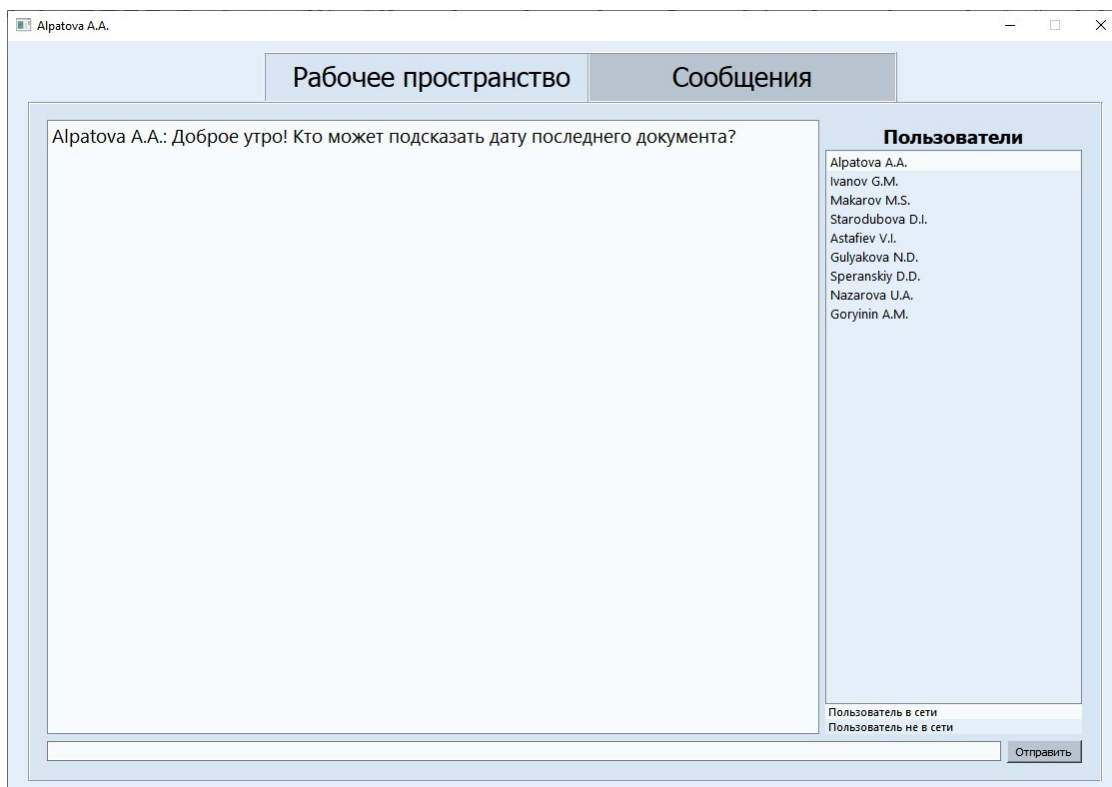


Рисунок 1.30 – Вкладка «Сообщения» после отправки сообщения

Также здесь предусмотрена возможность быстрого обращения к другим пользователям. Так после нажатия на логин пользователя в области «Пользователи» данный логин отобразится в поле ввода текста сообщения.

2 Безопасность жизнедеятельности

Прогрессирующее научно-техническое развитие современного общества создает задачу поддержания безопасности исполнения людьми своих трудовых обязанностей. Осуществлением данной задачи занимается отдельная наука о безопасности труда и жизнедеятельности человека.

Безопасность жизнедеятельности (БЖД) - это комплекс мероприятий, направленных на обеспечение безопасности человека в среде обитания, сохранение его здоровья, разработку методов и средств защиты путем снижения влияния вредных и опасных факторов до допустимых значений, выработку мер по ограничению ущерба в ликвидации последствий чрезвычайных ситуаций мирного и военного времени.

Охрана здоровья трудящихся, обеспечение безопасности условий труда, ликвидация профессиональных заболеваний и производственного травматизма составляет одну из главных забот человеческого общества. Обращается внимание на необходимость широкого применения прогрессивных форм научной организации труда, сведения к минимуму ручного, малоквалифицированного труда, создания обстановки, исключающей профессиональные заболевания и производственный травматизм [8].

2.1 Потенциально-опасные и вредные факторы

При эксплуатации ПО необходимо обращать внимание на воздействие вредных факторов на сотрудников. Вредный фактор рабочей среды - производственный фактор, воздействие которого на работника в определенных условиях может привести к заболеванию или отрицательному влиянию на здоровье потомства. Вредные и опасные факторы делятся на четыре категории: физические факторы, химические факторы, биологические факторы и факторы тяжести трудового процесса [8].

К физическим факторам относят микроклимат помещения, неионизирующие электромагнитные поля и излучения, электростатическое и постоянное магнитное, геомагнитное поля, источники ионизирующего излучения, производственный шум, ультразвук, инфразвук, вибрацию, аэрозоли преимущественно фиброгенного действия – пыли, световую среду, аэроионный состав воздуха.

Химические факторы – это химические вещества, смеси, получаемые химическим синтезом и/или для контроля которых используют методы химического анализа [8].

К биологическим факторам относятся микроорганизмы-продуценты, живые клетки и споры, патогенные микроорганизмы.

Факторы тяжести трудового процесса оценивают по тяжести труда. Тяжесть труда - характеристика трудового процесса, отражающая преимущественную нагрузку на опорно-двигательный аппарат и функциональные системы организма (сердечно-сосудистую, дыхательную и др.), обеспечивающие его деятельность [8].

2.2 Пожарная безопасность

Требования по присвоению категории пожарной безопасности касаются помещений, которые имеют производственное, техническое или складское значение.

Категорирование производится на основе следующих факторов:

- наличие горючих веществ и материалов и их количество;
- проектирование помещения и проводимые в них технологические процессы.

Категории, установленные для помещений рассмотрены в таблице 2.1.

Офис, который рассчитан на использование разработанного приложения, относится к низшей категории по взрывопожарной и пожарной

безопасности (Д), поскольку в помещении находятся негорючие вещества и материалы.

Таблица 2.1 – Категорирование помещений по взрывопожарной и пожарной опасности

Категория помещения	Характеристики веществ и материалов, находящихся (обращающихся) в помещении
А (повышенная взрывопожароопасность)	Горючий газ, легковоспламеняющиеся жидкости с температурой вспышки не более 28 градусов по Цельсию
Б (взрывопожароопасность)	Горючие пыли или волокна, легковоспламеняющиеся жидкости с температурой вспышки более 28 градусов по Цельсию
В1-В4 (пожароопасность)	Горючие и трудногорючие жидкости, твердые горючие и трудногорючие вещества и материалы
Г (умеренная пожароопасность)	Негорючие вещества и материалы в горячем, раскаленном или расплавленном состоянии
Д (пониженная пожароопасность)	Негорючие вещества и материалы в холодном состоянии

Наиболее частые причины пожаров – нарушение правил пожарной безопасности или технологических процессов, неправильная эксплуатация оборудования. Как на предприятии в целом, так и в отдельных помещениях необходимо проводить противопожарную профилактику.

Мероприятия по пожарной безопасности делятся на:

- организационные (правильная эксплуатация и содержание машин, транспорта, зданий, территорий, противопожарный инструктаж работников);
- технические (соблюдение противопожарных правил, норм при проектировании зданий, при устройстве и размещении оборудования);
- мероприятия режимного характера (запрещение курения в неустановленных местах, производства огневых работ в пожароопасных помещениях);
- эксплуатационные (своевременные профилактические осмотры, ремонты, испытания технологического оборудования) [8].

Также необходимо обеспечить предприятие средствами тушения пожаров и сигнализации.

Основные огнегасительные вещества – вода, пена, песок, инертные газы, сухие (твердые) огнегасительные вещества и т.д. Так на основе данных веществ используются следующие первичные средства пожаротушения: переносимые и передвижные огнетушители, пожарные краны и средства обеспечения их использования, пожарный инвентарь (лопаты, молотки, конусы и т.д.), одеяла для изоляции очага возгорания [8].

Территория предприятия с офисом, рассчитанной под эксплуатацию разработанного ПО, окружена металлическим забором, въезд на территорию осуществляется через центральные ворота, что важно для эвакуации при пожаре. На каждом этаже здания имеются планы эвакуации. В помещении присутствуют средства пожаротушения, установлен извещатель пожарный ручной. Применяются углекислотные огнетушители.

Таким образом, на предприятии соблюдаются противопожарные правила и нормы, имеется свободный доступ к средствам пожаротушения и запасному выходу. Расположение рабочих мест оптимально.

2.3 Электробезопасность

Помещения, опасные с точки зрения поражения людей током подразделяются на:

- особо опасные (особо сырые помещения, с химически активной или органической средой);
- с повышенной опасностью (помещения с наличием токопроводящей пыли, с высокой температурой, помещения, имеющие токопроводящие полы);
- без повышенной опасности (в помещении отсутствуют условия, создающие повышенную или особую опасность поражения людей электрическим током).

По опасности поражения электротоком данное помещение относится к помещениям без повышенной опасности, поскольку текущее помещение характеризуется отсутствием условий, создающих повышенную или особую опасность. К данным условиям относятся: сырость и токопроводящая пыль, токопроводящие полы, высокие температуры, химически активная или органическая среды, а также возможность одновременного прикосновения человека к металлоконструкциям зданий, имеющим соединение с землей, технологическими аппаратами, механизмам и т.п. – с одной стороны, и к металлическим корпусам электрооборудования – с другой [8].

С точки зрения эксплуатации электрооборудования помещения, считающиеся безопасными, должны соответствовать следующим критериям:

- температура в помещении ниже 30 градусов Цельсия;
- отсутствие токопроводящей пыли;
- влажность помещения не более 60 процентов;
- помещение хорошо проветривается;
- полы выполнены из непроводящего ток материала;

- заполнение площади помещения предметами из металла менее 20 процентов.

В свою очередь, персонал, эксплуатирующие электропроводящее оборудование, должен знать правила и требования технической эксплуатации, а также правила оказания первой помощи пострадавшему от действия электрического тока.

В число системы электробезопасности текущего предприятия входят: защита от прямого и косвенного прикосновения к токоведущим и токопроводящим частям оборудования, применение защитного заземления, использование защитного автоматического отключения питания. Работники проходят обучение безопасным методам и приемам выполнения работ с токопроводящим оборудованием, а также проходят проверку знаний и требований, предъявляемых к организации и выполнению работ.

2.4 Определение класса условий труда

На здоровье и работоспособность человека в процессе труда оказывает влияние совокупность факторов производственной (рабочей) среды и трудового процесса. Гигиенические критерии оценки условий труда позволяют оценить степень отклонений параметров производственной среды и трудового процесса от действующих гигиенических нормативов [8].

В частности, при организации рабочего места программиста должны быть соблюдены следующие основные условия:

- оптимальное размещение оборудования, входящего в состав рабочего места;
- достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения;
- необходимо естественное и искусственное освещение для выполнения поставленных задач;

					ПГУ 1.090301.01.001 ПЗ	Лист
Изм.	Лист	№ докум. №	Подпись	Дата		74

- уровень акустического шума не должен превышать допустимого значения».

Напряженность труда показывает степень воздействия нагрузок, связанных с рабочей деятельностью, на эмоциональные процессы, центральную нервную систему и органы чувств. Уровень напряженности оценивают по интеллектуальным, сенсорным и эмоциональным нагрузкам, а также их монотонности и режиму работы.

В ходе анализа трудовой деятельности учитываются следующие критерии: нагрузки интеллектуального характера, сенсорные нагрузки, эмоциональные нагрузки, режим работы.

По данным параметрам оценивается общая напряженность трудового процесса.

Для работника, эксплуатирующего текущее ПО, представленные нагрузки интеллектуального характера были отнесены к следующим классам:

- «Содержание работы» - 2 класс;
- «Восприятие сигналов (информации) и их оценка – 2 класс;
- «Распределение функций по степени сложности задания» – 2 класс;
- «Характер выполняемой работы» – 2 класс.

Сенсорные нагрузки:

- «Длительность сосредоточенного наблюдения (в процентах от времени смены)» - 1 класс (15%);
- «Плотность сигналов (световых, звуковых) и сообщений в среднем за 1 час работы» - 1 класс (1-3 сигнала);
- «Число производственных объектов одновременного наблюдения» - 1 класс (1-2 объекта);
- «Размер объекта различения при длительности сосредоточенного внимания (% от времени) – 1 класс (0 объектов);
- «Работа с оптическими приборами (микроскоп, лупа и т.п.) при длительности сосредоточенного наблюдения (% от времени)» - 1 класс;

- «Наблюдение за экраном видеотерминала (ч в смену)» - 1 класс;
- «Нагрузка на слуховой анализатор» - 1 класс;
- «Нагрузка на голосовой аппарат (суммарное количество часов, наговариваемых в неделю)» - 1 класс.

Эмоциональные нагрузки:

- «Степень ответственности за результат собственной деятельности. Значимость ошибки» - 2 класс;
- «Степень риска для собственной жизни» - 1 класс;
- «Ответственность за безопасность других лиц» - 1 класс;
- «Количество конфликтных производственных ситуаций за смену» - 1 класс.

Монотонность нагрузок:

- «Число элементов (приемов), необходимых для реализации простого задания или многократно повторяющихся операций» - 2 класс;
- «Продолжительность выполнения простых производственных заданий или повторяющихся операций» - 2 класс;
- «Время активных действий (в процентах к продолжительности смены)» - 2 класс (от 95%);
- «Монотонность производственной обстановки (время пассивного наблюдения за ходом техпроцесса, в процентах от времени смены)» - 1 класс (до 5%).

Режим работы:

- «Фактическая продолжительность рабочего дня» - 1 класс;
- «Сменность работы» - 1 класс;
- «Наличие регламентированных перерывов и их продолжительность (без учета обеденного перерыва)» - 3.1 класс.

Итак, по результатам данных показателей можно дать общую оценку напряженности трудового процесса. Ко 2 классу отнесены 7 показателей из

33, 1 показатель – к классу 3.1, остальные 25 – к 1 классу. Значит, общий трудовой процесс является «Допустимым» (2 класс).

Полученные показатели отражают степень опасности условий труда на организм. На основании рассмотренных характеристик разрабатывается комплекс мероприятий, направленных на повышение эффективности деятельности в рамках предприятия, а также на минимизацию вредного воздействия на здоровье человека.

Таким образом, благоприятные условия труда имеют большое значение как для облегчения труда, так и для повышения положительного влияния на производительность труда.

					ПГУ 1.090301.01.001 ПЗ	Лист
						77
Изм.	Лист	№ докум.№	Подпись	Дата		

Заключение

В ходе выполнения данной работы был проведен анализ имеющихся на современном рынке *CRM*-систем, выявлены их особенности и основные характеристики, удовлетворяющие потребности планирования работы и организации предприятия. Получены навыки в проектировании удобного и функционального интерфейса с помощью возможностей графической библиотеки.

Изучены архитектуры приложений, их типовые примеры и инструментарий. Получены навыки по разработке клиент-серверной архитектуры, а также по построению сетевых моделей передачи данных (*TCP/IP*).

Приобретены знания по созданию архитектуры хранения данных.

Построены основные *UML*-диаграммы: диаграммы классов, диаграммы деятельности, диаграмма развертывания и диаграмма вариантов использования.

В результате данной работы спроектирована, разработана и отлажена *CRM*-система предприятия. Приложение создано в *IDE Qt Creator* на основе объектно-ориентированного языка программирования *C++*. Используемый фреймворк *Qt5* обусловил разработанное приложение понятным графическим интерфейсом.

Данное приложение работает на основе клиент-серверного соединения с использованием *TCP/IP*. Каждая программа характеризуется многомодульностью. Обладает возможностью функционального расширения в связи с нуждами предприятия.

Список используемых источников

1. Научная электронная библиотека [Электронный ресурс] / Режим доступа <https://www.elibrary.ru> свободный (дата обращения 25.04.2022).
2. Голышева, Е. М. *CRM. Российская практика эффективного бизнеса* / Е. М. Голышева, А. Т. Кудинов. – М.: 1С-Паблишинг, 2009. – 463 с.
3. Точилкина, Т. Е. *Хранилища данных и средства бизнес-аналитики* / Т. Т. Точилкина, А. А. Громова. – М.: Финансовый университет, 2017. – 161 с.
4. Татаренко, В. И. *Безопасность труда в техносфере* / В. И. Татаренко, В. Л. Ромейко, О. П. Ляпина – Новосибирск: СГГА, 2012. – 469 с.
5. Боресков, А. В., Предисл.: Садовничий В. А. *Параллельные вычисления на GPU. Архитектура и программная модель CUDA: Учебное пособие* / А. В. Борсевков, В. А. Садовничий – М.: – Издательство Московского университета, 2012. – 336 стр.
6. Крейг, Х. *TCP/IP. Сетевое администрирование* / Х. Крейг. – М.: Символ-Плюс. 2006. – 816 с.
7. Шлее, М. *Qt 5.10. Профессиональное программирование на C++* / М. Шлее. – М.: БХВ-Петербург, 2018. – 1072 с.
8. Фролов, А. В. *Безопасности жизнедеятельности. Охрана труда: Учебное пособие для вузов* / А. В. Фролов. – Ростов н/Дю: Феникс, 2008. — 750 с.

Приложение А

(Обязательное)

CRM-система предприятия

Листинг программы

					ПГУ 1.090301.01.001 ПЗ	Лист
						80
Изм.	Лист	№ докум. №	Подпись	Дата		

Файл «chat.cpp»

```
#include "chat.h"
#include "ui_chat.h"

Chat::Chat(QWidget *parent, Client *cl, QString login) :
    QWidget(parent),
    ui(new Ui::Chat)
{
    ui->setupUi(this);
    setWindowTitle("Чат");
    ui->listUsers->setStyleSheet("background: #e4effa");
    ui->listChat->setStyleSheet("background: #f8fbfc");
    ui->textLine->setStyleSheet("background: #f8fbfc");
    ui->sendButton->setStyleSheet("background: #b8c4cf");
    ui->listWidget->setStyleSheet("background: #e4effa");
    ui->listWidget->item(0)->setBackgroundColor("#f8fbfc");
    ui->listWidget->setEditTriggers(QAbstractItemView::NoEditTriggers);
    ui->listWidget->setFocusPolicy(Qt::NoFocus);
    ui->listWidget->setSelectionMode(QAbstractItemView::NoSelection);
    ui->textEdit->setAlignment(Qt::AlignCenter);
    this->cl = cl;
    this->loginPub = login;
    init();
}

void Chat::init() {
    cl->getLogParFull();
    cl->getMesChat();
    cl->getClientOnline();
    connect(cl, &Client::signGetLogParFull, this, &Chat::getLogParFull);
    connect(cl, &Client::signGetMesChat, this, &Chat::setChatMes);
    connect(cl, &Client::signGetClientOnline, this, &Chat::getClientOnline);
}

Chat::~Chat()
{
    delete ui;
}

void Chat::getLogParFull(QList<columnsTableLogPass> list)
{
    ui->listUsers->clear();
    foreach(columnsTableLogPass c, list)
        ui->listUsers->addItem(c.login);
}

void Chat::on_sendButton_clicked()
{
    QDataStream ds;
    QString message;
    message = loginPub + ": " + ui->textLine->text();
    ui->listChat->addItem(message);
    ui->textLine->clear();
    cl->addMesChat(message);
}

void Chat::setChatMes(QStringList list)
{
    ui->listChat->clear();
}
```

```

        ui->listChat->addItem(list);
    }

void Chat::on_listUsers_currentTextChanged(const QString &currentText)
{
    if(ui->textLine->text().isEmpty())
        ui->textLine->setText(currentText);
    else {
        ui->textLine->setText(ui->textLine->text() + " " + currentText);
    }
}

void Chat::getClientOnline(QList<QString> list)
{
    socG = list;
    for (int i = 0; i < ui->listUsers->count(); i++){
        QString str = ui->listUsers->item(i)->text();
        qDebug() << str;
        if(loginPub == str)
            ui->listUsers->item(i)->setBackgroundColor("#f8fbfc");
        else
            for(int j = 0; j < socG.size(); j++)
                if(socG.at(j) == str)
                    ui->listUsers->item(i)->setBackgroundColor(Qt::blue);
    }
}

```

Файл «client.cpp»

```

#include "client.h"
#include <QDir>
#include <QApplication>
#include <QMessageBox>
#include <QIcon>
#include <QCryptographicHash>

Client::Client(const QString& strHost, int nPort, QWidget *parent) :
    QWidget(parent), m_nNextBlockSize(0)
{
    host = strHost;
    port = nPort;
    styleBack = "background: #f8fbfc";
    createMain();
    initClient(strHost, nPort);
}

void Client::initClient(const QString& strHost, int nPort){
    m_pTcpSocket = new QTcpSocket(this);
    m_pTcpSocket->connectToHost(strHost, nPort);
    connected = false;
    connect(m_pTcpSocket, SIGNAL(connected()), SLOT(slotConnected()));
    connect(m_pTcpSocket, SIGNAL(readyRead() ), SLOT(slotReadyRead()));
    connect(m_pTcpSocket, SIGNAL(disconnected()), SLOT(slotDisconnected()));
    connect(m_pTcpSocket, SIGNAL(error(QAbstractSocket::SocketError)), SLOT
(slotError(QAbstractSocket::SocketError)));
}

bool Client::prConnected(){
    connected = (m_pTcpSocket->state() == QTcpSocket::ConnectedState);
    qDebug() << "prConnected: " << connected;
    m_ptxt->append("prConnected: " + QString::number(connected));
}

```

```

        if(!connected)
        {
            QMessageBox msgBox;
            msgBox.setStyleSheet(styleBack);
            msgBox.setWindowTitle("Информация");
            msgBox.setText("Связь с сервером разорвана");
            msgBox.setIcon(QMessageBox::Information);
            msgBox.exec();
            timer = new QTimer(this);
            connect(timer, SIGNAL(timeout()), this, SLOT(slotReconnect()));
            timer->start(3000);
        }
        return connected;
    }

void Client::slotReconnect()
{
    qDebug() << "Client: slotReconnect";
    m_ptxt->append("Client: slotReconnect");
    connected = (m_pTcpSocket->state() == QTcpSocket::ConnectedState);
    if(connected)
    {
        QMessageBox msgBox;
        msgBox.setStyleSheet(styleBack);
        msgBox.setWindowTitle("Информация");
        msgBox.setText("Связь с сервером восстановлена");
        msgBox.setIcon(QMessageBox::Information);
        msgBox.exec();
        timer->stop();
    }
    else
    {
        m_ptxt->append("Server no found");
        m_pTcpSocket->connectToHost(host, port);
    }
}

void Client::createMain(){
    m_ptxt = new QTextEdit();
    m_ptxt->setReadOnly(true);
    but1 = new QPushButton("Отправить");
    QGridLayout *layout = new QGridLayout();
    layout->addWidget(m_ptxt);
    layout->addWidget(but1);
    setLayout(layout);
    connect(but1, SIGNAL(clicked()), SLOT(slotSendToServer()));
}

void Client::slotReadyRead()
{
    m_ptxt->append("Server: slotReadyRead - Пришло сообщение");
    QDataStream in(m_pTcpSocket);
    in.setByteOrder(QDataStream::BigEndian);
    for (;;) {
        if (!m_nNextBlockSize) {
            if (m_pTcpSocket->bytesAvailable() < sizeof(quint64)) {
                break;
            }
            in >> m_nNextBlockSize;
        }
        if (m_pTcpSocket->bytesAvailable() < m_nNextBlockSize)
            break;
    }
}

```

```

QString str;
in >> str;
qDebug() << "Server command: " << str;
if(str == "getListen")
{
    QList<columnsTable> list;
    in >> list;
    qDebug() << list.size();

    emit signGetListen(list);
}
else if(str == "vvLogPar")//vvLogPar - все ок. vvLogPar1 - нет такого
логина. vvLogPar2 - не правильный пароль
{
    emit signVVLogPar(true);
}
else if(str == "vvLogPar1")
{
    emit signVVErrror(0);
}
else if(str == "vvLogPar2")
{
    emit signVVErrror(1);
}
else if(str == "createLogPar0")//createLogPar0 - все ок, 1 - не ок, 2
- ошибка
{
    emit signSuccCr();
}
else if(str == "createLogPar1")
{
    emit signVVErrror(2);
}
else if(str == "createLogPar2")
{
    emit signVVErrror(3);
}
else if(str == "getGyrnal")
{
    QList<columnsTableMess> list;
    in >> list;
    qDebug() << list.size();

    emit signGetGyrnal(list);
}
else if(str == "getlogParFull")
{
    QList<columnsTableLogPass> list;
    in >> list;
    emit signGetlogParFull(list);
}
else if(str == "getMesChat")
{
    QStringList list;
    in >> list;
    emit signGetMesChat(list);
}
else if(str == "getClientOnline")
{
    QList<QString> socG;

```

```

        in >> socG;
        emit signGetClientOnline(socG);
    }

    m_ptxt->append("Server command: " + str);
    m_nNextBlockSize = 0;
}

QDataStream &operator>>(QDataStream &s, columnsTableLogPass &c)
{
    s >> c.uid >> c.login >> c.password;
    return s;
}

QDataStream &operator<<(QDataStream &s, const columnsTableLogPass &c)
{
    s << c.uid << c.login << c.password;
    return s;
}

//занесение в таблицу
QDataStream &operator>>(QDataStream &s, columnsTableMess &c)
{
    s >> c.uid >> c.dateTime >> c.login >> c.uidRecord >> c.operation;
    return s;
}

QDataStream &operator<<(QDataStream &s, const columnsTableMess &c)
{
    s << c.uid << c.dateTime << c.login << c.uidRecord << c.operation;
    return s;
}

QDataStream &operator>>(QDataStream &s, tagcolumnsTable &c)
{
    s >> c.uid >> c.name >> c.date01 >> c.number01 >> c.date02 >> c.date03 >>
c.number03
    >> c.date04 >> c.number04 >> c.date05 >> c.date06 >> c.number06
>> c.date07
    >> c.number07 >> c.number08 >> c.tz09 >> c.date10 >> c.number10
>> c.riperf10 >> c.rireason10
    >> c.date11 >> c.number12 >> c.date13 >> c.number13 >>
c.whommade13
    >> c.whomshow13 >> c.product14 >> c.detail15 >> c.date16 >>
c.date17 >> c.defectreason18
    >> c.date19 >> c.riperf20 >> c.rireason20 >> c.date20 >>
c.number20 >> c.date201 >> c.number201 >> c.actreason21
    >> c.date22 >> c.number22;
    return s;
}

QDataStream &operator<<(QDataStream &s, const tagcolumnsTable &c) //
{
    s << c.uid << c.name << c.date01 << c.number01 << c.date02 << c.date03 <<
c.number03
    << c.date04 << c.number04 << c.date05 << c.date06 << c.number06 <<
c.date07
    << c.number07 << c.number08 << c.tz09 << c.date10 << c.number10 <<
c.riperf10 << c.rireason10
    << c.date11 << c.number12 << c.date13 << c.number13 << c.whommade13

```

```

        << c.whomshow13 << c.product14 << c.detail15 << c.date16 << c.date17 <<
c.defectreason18
        << c.date19 << c.riperf20 << c.rireason20 << c.date20 << c.number20 <<
c.date201 << c.number201 << c.actreason21
        << c.date22 << c.number22;
        return s;
    }

void Client::slotError(QAbstractSocket::SocketError err) //если ошибки
{
    QString strError =
        "Error: " + (err == QAbstractSocket::HostNotFoundError?
                    "The host was not found." :
                    err == QAbstractSocket::RemoteHostClosedError?
                    "The remote host is closed." :
                    err ==
QAbstractSocket::ConnectionRefusedError?
                    "The connection was refused." :
                    QString(m_pTcpSocket->errorString()));
    m_ptxt->append(strError);
}

void Client::slotSendToServer()
{
    if(prConnected())
    {
        m_ptxt->append("Client: slotSendToServer");

        QByteArray arrBlock;
        QDataStream out(&arrBlock, QIODevice::WriteOnly);
        out.setByteOrder(QDataStream::BigEndian);
        QString str;
        out << quint64(0) << QTime::currentTime() <<
QString::fromUtf8("asd");

        out.device()->seek(0);
        out << quint64(arrBlock.size() - sizeof(quint64));
        m_pTcpSocket->write(arrBlock);
    }
}

void Client::slotConnected()
{
    connected = true;
    m_ptxt->append("Client: slotConnect");
    qDebug() << "Client: slotConnect: " << (m_pTcpSocket->state() ==
QTcpSocket::ConnectedState);
}

void Client::slotDisconnected(){
    connected = false;
    m_ptxt->append("Client: slotDisconnect");
    qDebug() << "disconnected: " << (m_pTcpSocket->state() ==
QTcpSocket::ConnectedState);
}

void Client::getListen(){
    qDebug() << "Client command: getListen";
    m_ptxt->append("Client: getListen?");

    if(prConnected())

```

```

    {
        qDebug() << "Client: getListen";
        m_ptxt->append("Client: getListen!");
        QByteArray arrBlock;
        QDataStream out(&arrBlock, QIODevice::WriteOnly);
        out.setByteOrder(QDataStream::BigEndian);
        out << quint64(0) << QString::fromUtf8("getListen");
        out.device()->seek(0);
        out << quint64(arrBlock.size() - sizeof(quint64));
        m_pTcpSocket->write(arrBlock);
    }
}

void Client::saveLetter(columnsTable col){ //сохранение записи
    qDebug() << "SAVELETTER";
    if(prConnected())
    {
        m_ptxt->append("Client: saveLetter");
        QByteArray arrBlock;
        QDataStream out(&arrBlock, QIODevice::WriteOnly);
        out.setByteOrder(QDataStream::BigEndian);

        //out.setVersion(QDataStream::Qt_5_12);
        QList<columnsTable> list;
        list.push_back(col);
        out << quint64(0) << QString::fromUtf8("saveLetter") << list;
        out.device()->seek(0);
        out << quint64(arrBlock.size() - sizeof(quint64));
        m_pTcpSocket->write(arrBlock);
        if(col.tz09 != "")
            fileTransfer(col.tz09);
    }
}

void Client::fileTransfer(QString nameF){ //передача файла
    qDebug() << "File Transfer";
    if(prConnected())
    {
        m_ptxt->append("Client: file transfer");
        QByteArray arrBlock;
        QDataStream out(&arrBlock, QIODevice::WriteOnly);
        out.setByteOrder(QDataStream::BigEndian);
        QFile file(nameF);
        file.open((QIODevice::ReadOnly));
        QByteArray buf = file.readAll();
        qDebug() << "file size:" << file.size() << buf.size();
        out << quint64(file.size()) << QString::fromUtf8("fileTransfer") <<
        getName(file.fileName()) << buf;
        m_pTcpSocket->write(arrBlock);
        m_pTcpSocket->flush();
    }
}

void Client::updateLetter(columnsTable col){ //создание записи
    qDebug() << "updateLetter";
    if(prConnected())
    {
        m_ptxt->append("Client: updateLetter");

        QByteArray arrBlock;
        QDataStream out(&arrBlock, QIODevice::WriteOnly);

```

```

        out.setByteOrder(QDataStream::BigEndian);
        QList<columnsTable> list;
        list.push_back(col);
        out << quint64(0) << QString::fromUtf8("updateLetter") << list;
        out.device()->seek(0);
        out << quint64(arrBlock.size() - sizeof(quint64));
        m_pTcpSocket->write(arrBlock);
    }
}

void Client::deleteLetter(int numb) //удаление записи
{
    qDebug() << "deleteLetter";
    if(prConnected())
    {
        m_ptxt->append("slotSendToServer");
        QByteArray arrBlock;
        QDataStream out(&arrBlock, QIODevice::WriteOnly);
        out.setByteOrder(QDataStream::BigEndian);
        QString str;
        out << quint64(0) << QString::fromUtf8("deleteLetter") << numb;
        out.device()->seek(0);
        out << quint64(arrBlock.size() - sizeof(quint64));
        m_pTcpSocket->write(arrBlock);
    }
}

QString Client::getName(QString nameDoc) //получение имени файла
{
    qDebug() << nameDoc;
    if(nameDoc.contains("/"))
    {
        int numb = 0;
        for(int i = 0; i < nameDoc.size(); i++)
            if(nameDoc.at(i) == "/")
                numb = i;
        QString nameDoc2 = "";
        for(int i = numb+1; i < nameDoc.size(); i++)
            nameDoc2 += nameDoc.at(i);
        nameDoc = nameDoc2;
    }
    return nameDoc;
}

void Client::vvLogPar(QString login, QString parol) //поучение введенных
логина и пароля при входе
{
    qDebug() << "vvLogPar";
    if(prConnected())
    {
        m_ptxt->append("Client: vvLogPar");
        QByteArray arrBlock;
        QDataStream out(&arrBlock, QIODevice::WriteOnly);
        out.setByteOrder(QDataStream::BigEndian);
        QByteArray parolByte = parol.toUtf8();
        parol = QCryptographicHash::hash(parolByte,
        QCryptographicHash::Sha256);
        out << quint64(0) << QString::fromUtf8("vvLogPar") << login << parol;
        out.device()->seek(0);
        out << quint64(arrBlock.size() - sizeof(quint64));
        m_pTcpSocket->write(arrBlock);
    }
}

```

```

    }
}

void Client::createLogPar(QString login, QString parol) //получение введенных
логина и пароля при регистрации
{
    qDebug() << "createLogPar";
    if(prConnected())
    {
        m_ptxt->append("Client: createLogPar");
        QByteArray arrBlock;
        QDataStream out(&arrBlock, QIODevice::WriteOnly);
        out.setByteOrder(QDataStream::BigEndian);
        QByteArray parolByte = parol.toUtf8();
        parol = QCryptographicHash::hash(parolByte,
        QCryptographicHash::Sha256);
        out << quint64(0) << QString::fromUtf8("createLogPar") << login <<
        parol;
        out.device()->seek(0);
        out << quint64(arrBlock.size() - sizeof(quint64));
        m_pTcpSocket->write(arrBlock);
    }
}

void Client::getGyrnal()
{
    qDebug() << "Client command: getGyrnal";
    if(prConnected())
    {
        qDebug() << "Client: getGyrnal";
        m_ptxt->append("Client: getGyrnal");
        QByteArray arrBlock;
        QDataStream out(&arrBlock, QIODevice::WriteOnly);
        out.setByteOrder(QDataStream::BigEndian);
        out << quint64(0) << QString::fromUtf8("getGyrnal");
        out.device()->seek(0);
        out << quint64(arrBlock.size() - sizeof(quint64));
        m_pTcpSocket->write(arrBlock);
    }
}

void Client::getlogParFull()
{
    qDebug() << "Client command: getlogParFull";

    if(prConnected())
    {
        qDebug() << "Client: getlogParFull";
        m_ptxt->append("Client: getlogParFull");
        QByteArray arrBlock;
        QDataStream out(&arrBlock, QIODevice::WriteOnly);
        out.setByteOrder(QDataStream::BigEndian);
        out << quint64(0) << QString::fromUtf8("getlogParFull");
        out.device()->seek(0);
        out << quint64(arrBlock.size() - sizeof(quint64));
        m_pTcpSocket->write(arrBlock);
    }
}

void Client::getMesChat()
{

```

```

qDebug() << "Client command: getChatMes";
if(prConnected())
{
    qDebug() << "Client: getChatMes";
    m_ptxt->append("Client: getChatMes");
    QByteArray arrBlock;
    QDataStream out(&arrBlock, QIODevice::WriteOnly);
    out.setByteOrder(QDataStream::BigEndian);
    out << quint64(0) << QString::fromUtf8("getMesChat");
    out.device()->seek(0);
    out << quint64(arrBlock.size() - sizeof(quint64));
    m_pTcpSocket->write(arrBlock);
}
}

void Client::getClientOnline()
{
    qDebug() << "Client command: getClientOnline";

    if(prConnected())
    {
        qDebug() << "Client: getClientOnline";
        m_ptxt->append("Client: getClientOnline");
        QByteArray arrBlock;
        QDataStream out(&arrBlock, QIODevice::WriteOnly);
        out.setByteOrder(QDataStream::BigEndian);
        out << quint64(0) << QString::fromUtf8("getClientOnline");
        out.device()->seek(0);
        out << quint64(arrBlock.size() - sizeof(quint64));
        m_pTcpSocket->write(arrBlock);
    }
}

void Client::addGyrnal(columnsTableMess tm)
{
    qDebug() << "Client command: addGyrnal";

    if(prConnected())
    {
        qDebug() << "Client: addGyrnal";
        m_ptxt->append("Client: addGyrnal");
        QByteArray arrBlock;
        QDataStream out(&arrBlock, QIODevice::WriteOnly);
        out.setByteOrder(QDataStream::BigEndian);
        QList<columnsTableMess> list;
        list.clear();
        list.append(tm);
        out << quint64(0) << QString::fromUtf8("addGyrnal") << list;
        out.device()->seek(0);
        out << quint64(arrBlock.size() - sizeof(quint64));
        m_pTcpSocket->write(arrBlock);
    }
}

void Client::addMesChat(QString message)
{
    qDebug() << "Client command: addMesChat";
    if(prConnected())
    {
        qDebug() << "Client: addMesChat";
        m_ptxt->append("Client: addMesChat");
    }
}

```

```

        QByteArray arrBlock;
        QDataStream out(&arrBlock, QIODevice::WriteOnly);
        out.setByteOrder(QDataStream::BigEndian);
        out << quint64(0) << QString::fromUtf8("addMesChat") << message;
        out.device()->seek(0);
        out << quint64(arrBlock.size() - sizeof(quint64));
        m_pTcpSocket->write(arrBlock);
    }
}

```

Файл «fullspisok.cpp»

```

#include "fullspisok.h"
#include <QDesktopWidget>
#include <QMessageBox>
#include <QDate>
#include <QLineEdit>

FullSpisok::FullSpisok(int reg, int numb, QString spisok)
{
    qDebug() << reg << numb;
    this->reg = reg;
    this->numb = numb;
    this->spisok = spisok;
    styleBack = "background: #f8fbfc";
    createWindow();
    table->setStyleSheet(" QHeaderView::section {background: #D8E6F3;}
    QTableWidget{background: #e4effa;} ");
    if(spisok != "")
        insertTable(this->spisok);
}

void FullSpisok::createWindow()
{
    w = new QWidget();
    qDebug() << "reg: " << reg;
    switch (reg) {
    case 0:
        w->setWindowTitle("Периферия ввода");
        createTable();
        w->setMinimumSize(680,500);
        w->setMaximumSize(680,500);
        break;
    case 1:
        w->setWindowTitle("Периферия вывода");
        createTable();
        w->setMinimumSize(680,500);
        w->setMaximumSize(680,500);
        break;
    case 2:
        w->setWindowTitle("Мероприятия");
        createTable();
        w->setMinimumSize(460,500);
        w->setMaximumSize(460,500);

        break;
    }

    QGridLayout *layout = new QGridLayout();
    layout->addWidget(table,0,0);
    //Основные поля с кнопками

```

```

        but1 = new QPushButton("Сохранить");
        but2 = new QPushButton("Закрыть");
        but3 = new QPushButton("Добавить строчку");
        but1->setMaximumWidth(100);
        but2->setMaximumWidth(100);
        but3->setMaximumWidth(300);
        but1->setStyleSheet("background: #b8c4cf");
        but2->setStyleSheet("background: #b8c4cf");
        but3->setStyleSheet("background: #b8c4cf");
        connect(but1, &QPushButton::clicked, this, &FullSpisok::saveSlots);
//сохранение
        connect(but2, &QPushButton::clicked, this, &FullSpisok::closeSlots);
        connect(but3, &QPushButton::clicked, this, &FullSpisok::addStr);
        QHBoxLayout *box = new QHBoxLayout();
        QDesktopWidget *d = QApplication::desktop();
        int width = d->geometry().width();
        QSpacerItem *sp = new QSpacerItem(width,10);
        box->addWidget(but3);
        box->addItem(sp);
        box->addWidget(but1);
        box->addWidget(but2);
        layout->addLayout(box,1,0);
        w->setGeometry(500,200,500,500);
        w->setLayout(layout);
        w->setStyleSheet("background: #D8E6F3");
        w->show();

    }

void FullSpisok::createTable() //создание таблицы при нажатии
{
    table = new QTableWidgetItem();

    if(reg != 2)
    {
        table->setColumnCount(5);
        table->setRowCount(1);
        table->setColumnWidth(2,200);
    }
    else {
        table->setColumnCount(1);
        table->setRowCount(1);
        table->setColumnWidth(0,400);
    }
    QStringList list;
    switch (reg) {
        case 0: //проверено
            list << "Наименование" << "Индекс" << "Номер" << "Дата" <<
"Поставщик";
            table->setHorizontalHeaderLabels(list);
            if(spisok.count("|") > 1)
                table->setRowCount(spisok.count("|"));
            if(spisok.count("|") > 1)
                table->setRowCount(spisok.count("|"));
            break;
        case 1:
            list << "Наименование" << "Индекс" << "Номер" << "Дата" <<
"Поставщик";
            table->setHorizontalHeaderLabels(list);
            if(spisok.count("|") > 1)
                table->setRowCount(spisok.count("|"));

```

```

        break;
    case 2:
        list << "Мероприятия";
        table->setHorizontalHeaderLabels(list);
        if(spisok.count("|") > 1)
            table->setRowCount(spisok.count("|"));
        break;
    }
    viewTable();
}

void FullSpisok::viewTable() //установка вида таблицы
{
    QTableWidgetItem* view;
    for(int i = 0; i < table->rowCount(); i++)
        for(int j = 0; j < table->columnCount(); j++)
        {
            view = new QTableWidgetItem();
            table->setItem(i,j,view);
            if(j == 3)
            {
                table->setCellWidget(i,j,new QLineEdit());
                QLineEdit *l = (QLineEdit *)table->cellWidget(i,j);
                l->setInputMask("00-00-0000");
                l->setStyleSheet("background: #e4effa");
            }
        }
}

void FullSpisok::insertTable(QString spisok) //вставка таблицы
{
    QStringList list;
    QStringList listD;
    QString slovo = "";
    switch (reg) {
    case 0:
        for(int i = 0; i < spisok.size(); i++)
        {
            QString str = spisok.at(i);
            if(str != "|")
            {
                slovo += str;
            }
            else {
                listD.push_back(slovo);
                slovo.clear();
            }
        }
        foreach (QString str, listD) {
            qDebug() <<"insert" << str;
        }
        slovo = "";
        for(int i = 0; i < listD.size(); i++)
        {
            spisok = listD.at(i);
            for(int j= 0; j < spisok.size(); j++)
            {
                QString str = spisok.at(j);
                if(str != ";")
                {
                    slovo += str;
                }
            }
        }
    }
}

```

```

        else {
            list.push_back(slovo);
            slovo.clear();
        }
    }
    for(int j = 0; j < table->columnCount(); j++) {

        if(j != 3)
            table->item(i,j)->setText(list.at(j));
        else {
            QLineEdit *l = (QLineEdit *)table->cellWidget(i,j);
            l->setText(list.at(j));
        }
    }
    list.clear();
}
break;

case 1:
    for(int i = 0; i < spisok.size(); i++)
    {
        QString str = spisok.at(i);
        if(str != "|")
        {
            slovo += str;
        }
        else {
            listD.push_back(slovo);
            slovo.clear();
        }
    }
    foreach (QString str, listD) {
        qDebug() <<"insert" << str;
    }
    slovo = "";

    for(int i = 0; i < listD.size(); i++)
    {
        spisok = listD.at(i);
        for(int j= 0; j < spisok.size(); j++)
        {
            QString str = spisok.at(j);
            if(str != ";")
            {
                slovo += str;
            }
            else {
                list.push_back(slovo);
                slovo.clear();
            }
        }

        for(int j = 0; j < table->columnCount(); j++) {
            if(j != 3)
                table->item(i,j)->setText(list.at(j));
            else {
                QLineEdit *l = (QLineEdit *)table->cellWidget(i,j);
                l->setText(list.at(j));
            }
        }
        list.clear();
    }

```

```

    }
    break;

case 2:
    for(int i = 0; i < spisok.size(); i++)
    {
        QString str = spisok.at(i);
        if(str != "|")
        {
            slovo += str;
        }
        else {
            slovo += str;

            if(slovo.startsWith(" "))
                slovo.remove(0,1);
            if(slovo.endsWith("|"))
                slovo.remove(slovo.length()-1,1);
            list.push_back(slovo);
            slovo.clear();
        }
    }

    for(int i = 0; i < list.size(); i++) {
        table->item(i,0)->setText(list.at(i));
    }
    break;
}

}

QString FullSpisok::getWords(int reg) //проверка на правильность ввода в
основную таблицу
{
    qDebug() << "getWords";
    QString words = "";
    QString str;
    QStringList list;
    QDate date;
    bool pr = true;
    bool signE = true;
    QLineEdit *l;

    switch (reg) { //проверено
    case 0:
        for(int j = 0; j < table->rowCount(); j++){
            for(int i = 0; i < 2; i++)
                words.append(table->item(j,i)->text() + ";");
            str = table->item(j,2)->text();
            for(int j = 0; j < str.length(); j++)
                if(!str[j].isDigit())
                    pr = false;

            if(pr)
                words.append(str+";");
            else {
                table->item(j,2)->setText("неправильный номер");
                words.append(";");
                signE = false;
            }
        }
        l = (QLineEdit*)table->cellWidget(j,3);
        str = l->text();

```

```

        if(str != "--")
        {
            date = QDate::fromString(str, "dd-MM-yyyy");

            if(!date.isValid() && str != "--")
            {
                words.append(";");
                signE = false;
            }
            else {
                words.append(date.toString("dd-MM-yyyy") + ";");
            }
        }
        else
            words.append(";");
        words.append(table->item(j,4)->text() + ";");

        words.append("|");
    }
    qDebug() << words;
    break;

case 1:
    for(int j = 0; j < table->rowCount(); j++){
        for(int i = 0; i < 2; i++){
            words.append(table->item(j,i)->text() + ";");

            str = table->item(j,2)->text();
            for(int j = 0; j < str.length(); j++)
                if(!str[j].isDigit())
                    pr = false;

            if(pr)
                words.append(str+";");
            else {
                table->item(j,2)->setText("неправильный номер");
                words.append(";");
                signE = false;
            }
            l = (QLineEdit*)table->cellWidget(j,3);
            str = l->text();
            if(str != "--")
            {
                date = QDate::fromString(str, "dd-MM-yyyy");

                if(!date.isValid() && str != "--")
                {
                    words.append(";");
                    signE = false;
                }
                else {
                    words.append(date.toString("dd-MM-yyyy") + ";");
                }
            }
            else
                words.append(";");

            words.append(table->item(j,4)->text() + ";");

            words.append("|");
        }
    }
}

```

```

        qDebug() << words;
        break;

    case 2:
        for(int i = 0; i < table->rowCount(); i++)
        {
            QString str = table->item(i,0)->text();
            if(!str.endsWith("|"))
                str += "|";
            words.append(str);
        }

        qDebug() << words;
        break;
    }

    if(!signE)
    {
        QMessageBox msgBox;
        msgBox.setWindowTitle("Ошибка");
        msgBox.setStyleSheet(styleBack);
        msgBox.setText("Данные некорректны!");
        msgBox.setStandardButtons(QMessageBox::Yes);
        int ret = msgBox.exec();

        switch (ret) {
        case QMessageBox::Yes:
            this->close();
            break;
        }
    }
    return words;
}

void FullSpisok::saveSlots() //сохранение записи
{
    qDebug() << "saveSlots";
    QString words = getWords(reg);
    spisok = words;
    qDebug() << words;
    emit signSave(reg, numb, words);
}

void FullSpisok::closeSlots()
{
    qDebug() << "closeSlots";

    QString str = getWords(reg);
    qDebug() << str << spisok;
    if(str != spisok && str != ";;;;;" && str != ";;;;;|" && str != "|")
    {
        QMessageBox msgBox;
        msgBox.setWindowTitle("Предупреждение");
        msgBox.setIcon(QMessageBox::Warning);
        msgBox.setStyleSheet(styleBack);
        msgBox.setText("Были обнаружены изменения.");
        msgBox.setInformativeText("Хотите выйти без сохранения?");
        msgBox.setStandardButtons(QMessageBox::Save | QMessageBox::Yes |
QMessageBox::No);
        msgBox.setDefaultButton(QMessageBox::Save);
        int ret = msgBox.exec();
    }
}

```

```

        switch (ret) {
        case QMessageBox::Save:
            saveSlots();
            w->close();
            this->close();
            break;
        case QMessageBox::Yes:
            w->close();
            this->close();
            break;
        case QMessageBox::No:
            break;
        }
    }

    else
    {
        w->close();
        this->close();
    }
}

void FullSpisok::addStr()
{
    QString words = getWords(reg);
    table->setRowCount(table->rowCount()+1);
    viewTable();
    insertTable(words);
}

```

Файл «gyrnal.cpp»

```

#include "gyrnal.h"
#include "ui_gyrnal.h"

#include <qdebug.h>

Gyrnal::Gyrnal(QWidget *parent, Client *cl) :
    QWidget(parent),
    ui(new Ui::Gyrnal)
{
    ui->setupUi(this);
    this->cl = cl;
    ui->pushButton_2->hide();
    ui->pushButton_3->hide();
    ui->tableWidget->setStyleSheet(" QHeaderView::section {background:
#D8E6F3;} QTableWidget{background: #e4effa;} ");
    ui->pushButton->setStyleSheet("background: #b8c4cf");
    init();
}

Gyrnal::~Gyrnal()
{
    delete ui;
}

void Gyrnal::init()
{
    this->setWindowTitle("Журнал");
    connect(cl, &Client::signGetGyrnal, this, &Gyrnal::getGyrnal);
    cl->getGyrnal();
}

```

```

}

void Gyrnal::getGyrnal(QList<columnsTableMess> list)
{
    qDebug() << list.size();
    createTable();
    insertTable(list);
}

void Gyrnal::createTable() //создание таблицы
{
    QStringList list;
    list.append("Дата и Время");
    list.append("Логин");
    list.append("ID документа");
    list.append("Операция");
    ui->tableWidget->setColumnCount(list.size());

    ui->tableWidget->setColumnWidth(2,200);

    ui->tableWidget->setHorizontalHeaderLabels(list);
}

void Gyrnal::insertTable(QList<columnsTableMess> list) //вставка таблицы
{
    ui->tableWidget->setRowCount(list.size());

    if(list.size() != 0)
    {
        for(int i = 0; i < ui->tableWidget->rowCount(); i++)
            for(int j = 0; j < ui->tableWidget->columnCount(); j++)
            {
                QTableWidgetItem *view = new QTableWidgetItem();
                if(j != 2)
                {
                    view->setFlags(view->flags() & ~Qt::ItemIsEditable);
                    view->setTextAlignment(Qt::AlignJustify);
                    view->setTextAlignment(Qt::AlignCenter);
                }
                ui->tableWidget->setItem(i,j,view);
            }
        for(int i = 0 ; i < ui->tableWidget->rowCount(); i++)
        {
            ui->tableWidget->item(i,0)->setText(list.at(i).dateTime);
            ui->tableWidget->item(i,1)->setText(list.at(i).login);
            ui->tableWidget->item(i,2)-
            >setText(QString::number(list.at(i).uidRecord));
            ui->tableWidget->item(i,3)->setText(list.at(i).operation);
        }
    }
}

void Gyrnal::on_pushButton_clicked() //закрытие
{
    this->close();
}

```

Файл «login.cpp»

```

#include "login.h"
#include "QQuickWidget"

```

```

#include "QQmlApplicationEngine"
#include "QQmlComponent"
#include <menu.h>

Login::Login(QWidget *parent) : QWidget(parent)
{
    QDir dir(qApp->applicationDirPath() + "qrc:/main.qml");
    qDebug() << dir.absolutePath();
    QQmlApplicationEngine *eng = new QQmlApplicationEngine(this);
    QQmlComponent *comp = new QQmlComponent(eng, QUrl ("main.qml"));
    QObject* pobj = comp->create();
    QObject* pandOkButton = pobj->findChild<QObject*>("button1");
    if (pandOkButton){
        qDebug() << "button find";
        QObject::connect(pandOkButton, SIGNAL(buttonClicked(QString, QString)),
                        this, SLOT(slotClicked(QString, QString))) ;
    }

    QObject* pandRegButton = pobj->findChild<QObject*>("button4");
    if (pandRegButton){
        qDebug() << "button find";
        QObject::connect(pandRegButton, SIGNAL(buttonClicked()),
                        this, SLOT(createNewPol())) ;
    }
    textEdit1 = pobj->findChild<QObject*>("text1");
    textEdit2 = pobj->findChild<QObject*>("text2");
    endButton = pobj->findChild<QObject*>("button3");
    cl = new Client("localhost", 2323);
    styleBack = "background: #f8fbfc";
    connect(cl, &Client::signVVLogPar, this, &Login::getVVLogPar);
    connect(cl, &Client::signVVError, this, &Login::createErrorVV);
    connect(cl, &Client::signSuccCr, this, &Login::successfullyCreate);
}

void Login::successfullyCreate()
{
    QMessageBox *msgBox = new QMessageBox();
    msgBox->setWindowTitle("Информация");
    msgBox->setText("Регистрация прошла успешно!");
    msgBox->setIcon(QMessageBox::Information);
    msgBox->setStyleSheet(styleBack);
    msgBox->exec();
    w->close();
}

void Login::slotClicked(QString str, QString str2)
{
    qDebug() << "click" << str << str2;
    if(str != "" && str2 != "")
    {
        qDebug() << "if-1" << str << str2;
        this->loginG = str;
        this->passwordG = str2;
        cl->vvLogPar(str, str2);
    }
    else if(str == "" && str2 != "")
    {
        qDebug() << "if-2" << str << str2;
        QMessageBox *msgBox = new QMessageBox();
        msgBox->setWindowTitle("Ошибка");
        msgBox->setIcon(QMessageBox::Critical);
    }
}

```

```

        msgBox->setText("Введите логин!");
        msgBox->setStyleSheet(styleBack);
        msgBox->exec();
    }
    else if(str != "" && str2 == "")
    {
        qDebug() << "if-3" << str << str2;

        QMessageBox *msgBox = new QMessageBox();
        msgBox->setWindowTitle("Ошибка");
        msgBox->setIcon(QMessageBox::Critical);
        msgBox->setText("Введите пароль!");
        msgBox->setStyleSheet(styleBack);
        msgBox->exec();
    }
    else if(str == "" && str2 == "")
    {
        qDebug() << "if-4" << str << str2;
        QMessageBox *msgBox = new QMessageBox();
        msgBox->setIcon(QMessageBox::Critical);
        msgBox->setWindowTitle("Ошибка");
        msgBox->setStyleSheet(styleBack);
        msgBox->setText("Заполните все поля!");
        msgBox->exec();
    }
}

void Login::getVVLogPar(bool pr)
{
    qDebug() <<"login parol: " << pr;
    if(pr)
    {
        clickEndButton();
        Menu *menu = new Menu(nullptr,cl,loginG);
        menu->show();
    }
}

void Login::clickEndButton(){
    if(endButton)
    {
        QMetaObject::invokeMethod(endButton, "close",Qt::QueuedConnection);
    }
}

void Login::createErrorVV(int er){
    QMessageBox *msgBox = new QMessageBox();
    msgBox->setStyleSheet(styleBack);
    msgBox->setIcon(QMessageBox::Critical);
    msgBox->setWindowTitle("Ошибка");
    if(er == 0)
    {
        msgBox->setIcon(QMessageBox::Warning);
        msgBox->setWindowTitle("Предупреждение");
        msgBox->setText("Пользователя с таким логином не существует.\nХотите зарегистрироваться?");
        msgBox->setStandardButtons(QMessageBox::Yes | QMessageBox::Close);
        msgBox->setDefaultButton(QMessageBox::Yes);
        int ret = msgBox->exec();
        switch (ret) {
            case QMessageBox::Close:

```

```

        this->close();
        break;
    case QMessageBox::Yes:
        createNewPol();
        this->close();
        break;
    }
}
else if(er == 1)
{
    msgBox->setText("Введен неверный пароль!");
    msgBox->exec();
}
else if(er == 2)
{
    msgBox->setText("Пользователь с таким логином уже существует");
    msgBox->exec();
}
else if(er == 3)
{
    msgBox->setText("Неизвестная ошибка.\nОбратитесь к администрации");
    msgBox->exec();
}
}

void Login::createLogPar()
{
    qDebug() << "Login: createLogin";
    qDebug() << le1->text() << le2->text();
    if(le1->text() == "" && le2->text() != "")
    {
        QMessageBox *msgBox = new QMessageBox();
        msgBox->setIcon(QMessageBox::Information);
        msgBox->setStyleSheet(styleBack);
        msgBox->setWindowTitle("Ошибка");
        msgBox->setText("Введите логин!");
        msgBox->exec();
    }
    else if(le1->text() != "" && le2->text() == "")
    {
        QMessageBox *msgBox = new QMessageBox();
        msgBox->setIcon(QMessageBox::Information);
        msgBox->setStyleSheet(styleBack);
        msgBox->setWindowTitle("Ошибка");
        msgBox->setText("Введите пароль!");
        msgBox->exec();
    }
    else if(le1->text() == "" && le2->text() == "")
    {
        QMessageBox *msgBox = new QMessageBox();
        msgBox->setIcon(QMessageBox::Information);
        msgBox->setStyleSheet(styleBack);
        msgBox->setWindowTitle("Ошибка");
        msgBox->setText("Заполните все поля!");
        msgBox->exec();
    }
    else{
        cl->createLogPar(le1->text(), le2->text());
    }
}

```

```

void Login::createNewPol()
{
    w = new QWidget();
    w->setWindowTitle("Новый пользователь");
    w->setStyleSheet(styleBack);
    w->setMaximumSize(350,110);
    w->setMinimumSize(350,110);
    QLabel *l1 = new QLabel("Логин: ");
    QLabel *l2 = new QLabel("Пароль: ");
    l1->setStyleSheet("font: 14px");
    l2->setStyleSheet("font: 14px");
    le1 = new QLineEdit();
    le2 = new QLineEdit();

    QPushButton *b1 = new QPushButton("Создать пользователя", this);
    QGridLayout *l = new QGridLayout(w);
    l->addWidget(l1,0,0);
    l->addWidget(l2,1,0);
    l->addWidget(le1,0,1);
    l->addWidget(le2,1,1);
    l->addWidget(b1,2,1);
    connect(b1, &QPushButton::clicked, this, &Login::createLogPar);
    w->show();
}

```

Файл «mail.cpp»

```

#include "gyrnal.h"
#include "chat.h"
#include "mail.h"
#include "ui_mail.h"
#include <QDesktopServices>
#include <QLineEdit>
#include <QMessageBox>
#include <QScrollArea>

Mail::Mail(QWidget *parent, Client *cl, QString login) :
    QWidget(parent),
    ui(new Ui::Mail)
{
    ui->setUpUi(this);
    this->cl = cl;
    this->login = login;
    styleBack = "background: #f8fbfc";

    ui->lineEdit->setStyleSheet("background: #e4effa");
    ui->listWidget->setStyleSheet("background: #e4effa");
    ui->comboBox->setStyleSheet("background: #e4effa");
    ui->tableWidget->setStyleSheet("background: #e4effa");
    ui->pushButton->setStyleSheet("background: #b8c4cf");
    ui->pushButton_2->setStyleSheet("background: #b8c4cf");
    ui->pushButton_5->setStyleSheet("background: #b8c4cf");
    ui->pushButton_6->setStyleSheet("background: #b8c4cf");
    ui->textEdit->setAlignment(Qt::AlignCenter);
    ui->lineEdit->setStyleSheet("background: #f8fbfc");
    qDebug() << "login mail: " << login;
    loginPub = this->login;
    qDebug() << "login mail222: " << loginPub;
    ui->tableWidget->horizontalHeader()->hide();
    ui->tableWidget->verticalHeader()->hide();
}

```

```

        init();
    }

Mail::~Mail()
{
    delete ui;
}

void Mail::init(){
    setWindowTitle("Mail");
    ui->pushButton_3->setHidden(true); //open client
    ui->pushButton_4->setHidden(true); //чат
    ui->pushButton_5->setHidden(true); //сохранить
    connect(cl, &Client::signGetListen, this, &Mail::getListen);
    cl->getListen();
}

void Mail::setListen(QList<columnsTable> list)
{
    ui->listWidget->clear();
    if(!provList(list))
    {
        ui->listWidget->clear();
        qDebug() << "setListen" << listG.size();
        for(int i = 0; i < list.size(); i++)
        {
            QString name = list.at(i).name;
            if(name.isEmpty())
                ui->listWidget->addItem(QString::number(list.at(i).uid));
            else
            {
                ui->listWidget->addItem(name);
                ui->listWidget->item(ui->listWidget->count()-1)-
>setWhatsThis(QString::number(list.at(i).uid));
            }
        }
    }
}

bool Mail::provList(QList<columnsTable> list)
{
    bool pr = true;
    QList<int> listL;
    for(int i = 0; i < ui->listWidget->count(); i++)
        listL.push_back(ui->listWidget->item(i)->text().toInt());
    if(listL.size() == list.size())
        for(int i = 0; i < listL.size(); i++)
        {
            if(!(listL.at(i) == list.at(i).uid))
            {
                pr = false;
                break;
            }
        }
    else
        pr = false;
    qDebug() << "pr = " << pr;
    return pr;
}

void Mail::zapTable(int num)
{

```

```

qDebug() <<"zaptable"<< num;
columnsTable table = listG.at(num);
ui->lineEdit->setText(table.name);
QLineEdit *l = (QLineEdit *)ui->tableWidget->cellWidget(0,2);
l->setText(table.date01);
ui->tableWidget->item(1,2)->setText(table.number01);
l = (QLineEdit *)ui->tableWidget->cellWidget(2,2);
l->setText(table.date02);
l = (QLineEdit *)ui->tableWidget->cellWidget(3,2);
l->setText(table.date03);
ui->tableWidget->item(4,2)->setText(table.number03);
ui->tableWidget->item(5,2)->setText(table.number04);
l = (QLineEdit *)ui->tableWidget->cellWidget(6,2);
l->setText(table.date04);
l = (QLineEdit *)ui->tableWidget->cellWidget(7,2);
l->setText(table.date05);
l = (QLineEdit *)ui->tableWidget->cellWidget(8,2);
l->setText(table.date06);
ui->tableWidget->item(9,2)->setText(table.number06);

ui->tableWidget->item(10,2)->setText(table.number07);
l = (QLineEdit *)ui->tableWidget->cellWidget(11,2);
l->setText(table.date07);
ui->tableWidget->item(12,2)->setText(table.number08);
if(table.tz09 != "")
{
    QString nameDoc = table.tz09;
    if(nameDoc.contains("/"))
    {
        int numb = 0;
        for(int i = 0; i < nameDoc.size(); i++)
            if(nameDoc.at(i) == "/")
                numb = i;
        QString nameDoc2 = "";
        for(int i = numb+1; i < nameDoc.size(); i++)
            nameDoc2 += nameDoc.at(i);
        if(nameDoc2.contains("."))
        {
            numb = 0;
            for(int i = 0; i < nameDoc2.size(); i++)
                if(nameDoc2.at(i) == ".")
                    numb = i;
            nameDoc = "";
            for(int i = 0; i < numb; i++)
                nameDoc += nameDoc2.at(i);
        }
    }
    ui->tableWidget->item(13,2)->setText(table.tz09);
    butFile->setText(nameDoc);
}
ui->tableWidget->item(14,2)->setText(table.riperf10);
ui->tableWidget->item(15,2)->setText(table.rireason10);
ui->tableWidget->item(16,2)->setText(table.number10);
l = (QLineEdit *)ui->tableWidget->cellWidget(17,2);
l->setText(table.date10);
l = (QLineEdit *)ui->tableWidget->cellWidget(18,2);
l->setText(table.date11);
ui->tableWidget->item(19,2)->setText(table.number12);
ui->tableWidget->item(20,2)->setText(table.number13);
l = (QLineEdit *)ui->tableWidget->cellWidget(21,2);
l->setText(table.date13);

```

```

ui->tableWidget->item(22,2)->setText(table.whommadel3);
ui->tableWidget->item(23,2)->setText(table.whomshowl3);
ui->tableWidget->item(24,2)->setText(table.productl4);
ui->tableWidget->item(25,2)->setText(table.detaill5);
l = (QLineEdit *)ui->tableWidget->cellWidget(26,2);
l->setText(table.date16);
l = (QLineEdit *)ui->tableWidget->cellWidget(27,2);
l->setText(table.date17);
ui->tableWidget->item(28,2)->setText(table.defectreason18);
l = (QLineEdit *)ui->tableWidget->cellWidget(29,2);
l->setText(table.date19);
ui->tableWidget->item(30,2)->setText(table.riperf20);
ui->tableWidget->item(31,2)->setText(table.rireason20);
ui->tableWidget->item(32,2)->setText(table.number20);
ui->tableWidget->item(33,2)->setText(table.date20);
l = (QLineEdit *)ui->tableWidget->cellWidget(33,2);
l->setText(table.date20);
ui->tableWidget->item(34,2)->setText(table.number201);
l = (QLineEdit *)ui->tableWidget->cellWidget(35,2);
l->setText(table.date201);
ui->tableWidget->item(36,2)->setText(table.actreason21);
ui->tableWidget->item(37,2)->setText(table.number22);
l = (QLineEdit *)ui->tableWidget->cellWidget(38,2);
l->setText(table.date22);
for (int i = 0; i < ui->tableWidget->rowCount(); i++){
    ui->tableWidget->item(i,0)->setBackgroundColor("#D8E6F3");
    //////////////////////////////////
    ui->tableWidget->item(i,1)->setBackgroundColor("#D8E6F3");
    //////////////////////////////////
}
}

void Mail::createTable()
{
    qDebug() << "createTable";

    QTableWidget *table = ui->tableWidget;
    table->clear();
    QStringList list, list2; //первый и второй столбцы
    list.push_back("Дата и номер составления заявки от кадровой службы");
    list.push_back("");
    list.push_back("Дата получения заявки от кадровой службы");
    list.push_back("Номер резюме и дата \него составления");
    list.push_back("");
    list.push_back("Номер заключения трудового договора и дата его
заключения");
    list.push_back("");
    list.push_back("Дата первого выхода сотрудника");
    list.push_back("Дата и номер заявки по выдаче \птехнических средств");
    list.push_back("");
    list.push_back("Ответ сотрудника\n о получении технических средств");
    list.push_back("");
    list.push_back("Регистрационный номер технического оборудования");
    list.push_back("Перечень характеристик технического оборудования");
    list.push_back("Результаты введения в эксплуатацию \птехнического
оборудования, его номер и дата введения");
    list.push_back("");
    list.push_back("");
    list.push_back("");
    list.push_back("Дата получения документации по техническому
оборудованию");

```

```

list.push_back("Ответ по претензионному акту \not сотрудника");
list.push_back("Номер и дата составления претензионного акта по
\нполученным техническим средствам.\nКем составлен.\nКому предъявлен");
list.push_back("");
list.push_back("");
list.push_back("");
list.push_back("Полученные сотрудником \нпериферия ввода \ни вывода");
list.push_back("");
list.push_back("Дата приемки сотрудником программного обеспечения");
list.push_back("Дата обнаружения сбоя");
list.push_back("Причина сбоя по претензионному акту");
list.push_back("Дата получения претензионного акта");
list.push_back("Результаты исследования,\nномер и дата акта у
изготовителя.\nНомер и дата программы исследования");
list.push_back("");
list.push_back("");
list.push_back("");
list.push_back("");
list.push_back("");
list.push_back("Мероприятия по устранению причин сбоя");
list.push_back("Отметка об удовлетворении претензии");
list.push_back("");
list2.push_back("Дата");    list2.push_back("Номер");
list2.push_back("Дата");
list2.push_back("Дата");    list2.push_back("Номер");
list2.push_back("Номер");    list2.push_back("Дата");
list2.push_back("Дата");
list2.push_back("Дата");    list2.push_back("Номер");
list2.push_back("Номер");    list2.push_back("Дата");
list2.push_back("Номер");
list2.push_back("Характеристики");
list2.push_back("Обнаруженные дефекты");    list2.push_back("Степень
дефектности");    list2.push_back("Номер");    list2.push_back("Дата");
list2.push_back("Дата");
list2.push_back("Номер");
list2.push_back("Номер");    list2.push_back("Дата");
list2.push_back("Кем составлен");    list2.push_back("Кому предъявлен");
list2.push_back("Периферия ввода");
list2.push_back("Периферия вывода");
list2.push_back("Дата");
list2.push_back("Дата");
list2.push_back("Причина сбоя");
list2.push_back("Дата");
list2.push_back("Обнаруженный дефект");    list2.push_back("ФИО
сотрудника");    list2.push_back("Номер");    list2.push_back("Дата");
list2.push_back("Номер");    list2.push_back("Дата");
list2.push_back("Мероприятия");
list2.push_back("Номер");    list2.push_back("Дата");
table->setRowCount(list.size());
table->setColumnCount(3);
QTableWidgetItem* view;
for(int i = 0; i < table->rowCount(); i++)
    for(int j = 0; j < table->columnCount(); j++)
    {
        view = new QTableWidgetItem();
        if(j != 2)
        {
            view->setFlags(view->flags() & ~Qt::ItemIsEditable);
            view->setTextAlignment(Qt::AlignJustify);
            view->setTextAlignment(Qt::AlignCenter);
        }
    }

```

```

        table->setItem(i,j,view);
    }
    for(int i = 0; i < list.size(); i++)
    {
        table->item(i,0)->setText(list.at(i));
        table->item(i,1)->setText(list2.at(i));
        table->setRowHeight(i, 50);
    }

    table->setRowHeight(0, 40);
    table->setRowHeight(1, 40);
    table->setRowHeight(3, 40);
    table->setRowHeight(4, 40);
    table->setRowHeight(8, 40);
    table->setRowHeight(9, 40);
    table->setRowHeight(12, 80);
    table->setRowHeight(28, 80);
    table->setColumnWidth(0,200);
    table->setColumnWidth(2,150);
    table->setColumnWidth(3,200);
    table->setSpan(0,0,2,1);
    table->setSpan(3,0,2,1);
    table->setSpan(5,0,2,1);
    table->setSpan(8,0,2,1);
    table->setSpan(10,0,2,1);
    table->setSpan(14,0,4,1);
    table->setSpan(20,0,4,1);
    table->setSpan(24,0,2,1);
    table->setSpan(30,0,6,1);
    table->setSpan(37,0,2,1);
    table->item(0,2)->setToolTip("10-10-2021");

    //кнопки 3 столбца
    butFile = new QPushButton("Открыть файл");
    butS = new QPushButton("Периферия ввода");
    butS2 = new QPushButton("Периферия вывода");
    butS3 = new QPushButton("Мероприятия");
    for(int i = 0; i < ui->tableWidget->rowCount(); i++)
    {
        if(ui->tableWidget->item(i,1)->text() == "Дата")
        {
            ui->tableWidget->setCellWidget(i,2,new QLineEdit());
            QLineEdit *l = (QLineEdit *)ui->tableWidget->cellWidget(i,2);
            l->setInputMask("00-00-0000");
        }
    }
    ui->tableWidget->setCellWidget(13,2,butFile);
    ui->tableWidget->setCellWidget(24,2,butS);
    ui->tableWidget->setCellWidget(25,2,butS2);
    ui->tableWidget->setCellWidget(36,2,butS3);
}

QString Mail::getName(QString nameDoc)
{
    qDebug() <<nameDoc;
    if(nameDoc.contains("/"))
    {
        int numb = 0;
        for(int i = 0; i < nameDoc.size(); i++)
            if(nameDoc.at(i) == "/")
                numb = i;
    }
}

```

```

        QString nameDoc2 = "";
        for(int i = numb+1; i < nameDoc.size(); i++)
            nameDoc2 += nameDoc.at(i);

        if(nameDoc2.contains("."))
        {
            numb = 0;
            for(int i = 0; i < nameDoc2.size(); i++)
                if(nameDoc2.at(i) == ".")
                    numb = i;
            nameDoc = "";
            for(int i = 0; i < numb; i++)
                nameDoc += nameDoc2.at(i);
        }
    }
    return nameDoc;
}

QString Mail::getNameFormat(QString nameDoc)
{
    qDebug() <<nameDoc;
    if(nameDoc.contains("/"))
    {
        int numb = 0;
        for(int i = 0; i < nameDoc.size(); i++)
            if(nameDoc.at(i) == "/")
                numb = i;
        QString nameDoc2 = "";
        for(int i = numb+1; i < nameDoc.size(); i++)
            nameDoc2 += nameDoc.at(i);
        nameDoc = nameDoc2;
    }
    return nameDoc;
}

void Mail::getListen(QList<columnsTable> list) //???
{
    for(int i = 0; i < list.size(); i++)
    {
        QString name = list.at(i).name;
        QString date = "";
        if(name.contains("||")) //если в name есть ||
        {
            QString str = "";
            int a = -1;
            for(int j = 0; j < name.length(); j++)
            {
                if(str == name.at(j) && name.at(j) == "|")
                {
                    a = j;
                    break;
                }
                else
                    str = name.at(j);
            }
            for(int j = a+1; j < name.length(); j++)
            {
                date += name.at(j);
            }
            name.resize(a-1);
        }
    }
}

```

```

        columnsTable table = list.at(i);
        table.name = name;
        table.date = date;
        list.removeAt(i);
        list.insert(i, table);
    }
}
listG = list;
setListen(list);
on_comboBox_currentIndexChanged(ui->comboBox->currentText());
}
bool Mail::on_listWidget_prov(QListWidgetItem *item) {
    bool pr = true;
    if(ui->tableWidget->columnCount() != 0)
    {
        columnsTable table = getTable();
        if(curL != -1){
            columnsTable table2 = listG.at(curL);
            qDebug() << "UID" << listG.at(curL).uid;
            pr = equals(table, table2); //сравниваются введенная таблица и
...
            if(pr)
                qDebug() << "uspeshno (equals)";
            else
                qDebug() << "ne uspeshno";
        }
        else if(curL == -1 && !equals(table, colG)){
            pr = false;
        }
        else if(curL == -1)
            pr = true;
        else {
            pr = false;
        }
    }
    if(pr){ //если эти таблицы равны
        ui->pushButton_4->setHidden(true);
        ui->pushButton_5->setHidden(false);

        curL = getId(ui->listWidget->currentItem()->text(), ui->listWidget-
>currentItem()->whatsThis());
        qDebug() << "curL" << curL;
        createTable();
        zapTable(curL);
        if(listG.at(curL).tz09 == "") //если еще нет файла
            connect(butFile, &QPushButton::clicked, this, &Mail::createFile);
        else
            connect(butFile, &QPushButton::clicked, this, &Mail::mesFile);
        connect(butS, &QPushButton::clicked, this, &Mail::butS1SlotE);
        //заполнение других таблиц
        connect(butS2, &QPushButton::clicked, this, &Mail::butS2SlotE);
        connect(butS3, &QPushButton::clicked, this, &Mail::butS3SlotE);
    }
    else {
        QMessageBox msgBox;
        msgBox.setWindowTitle("Предупреждение");
        msgBox.setStyleSheet(styleBack);
        msgBox.setIcon(QMessageBox::Warning);
        msgBox.setText("Были обнаружены изменения.");
        msgBox.setInformativeText("Хотите выйти без сохранения?");
    }
}

```

```

        msgBox.setStandardButtons(QMessageBox::Save | QMessageBox::Yes |
QMessageBox::No);
        msgBox.setDefaultButton(QMessageBox::Save);
        int ret = msgBox.exec();
        switch (ret) {
        case QMessageBox::Save:
            on_pushButton_5_clicked();
            break;
        case QMessageBox::Yes:
            closeTab();
            break;

        case QMessageBox::No:
            break;
        }
    }
    return pr;
}

void Mail::on_listWidget_itemClicked(QListWidgetItem *item) //???
{
    qDebug() << "on_listWidget_itemClicked";

    on_listWidget_prov(item);
}
int Mail::getId(QString name, QString wat)
{
    int numb = -1;
    if(wat != "")
    {
        numb = wat.toInt();
        for(int i= 0 ; i < listG.size(); i++)
            if(listG.at(i).uid == numb)
                numb = i;
    }
    else
        for(int i = 0; i < listG.size(); i++)
        {
            columnsTable table = listG.at(i);
            if(QString::number(table.uid) == name && wat == "")
            {
                numb = i;
            }
        }
    return numb;
}

void Mail::on_tableWidget_itemClicked(QTableWidget *item)//клик
{
    qDebug() << item->text() << item->row();
}

void Mail::on_pushButton_2_clicked()// удаление - запись в журнал
{
    ui->pushButton_4->setHidden(true);
    ui->pushButton_5->setHidden(true);
    ui->tableWidget->setColumnCount(0);
    ui->tableWidget->setRowCount(0);

    if(ui->listWidget->currentRow() != -1)

```

```

{
    qDebug() << "currentRow: " << ui->listWidget->currentRow();
    int i = curL;
    int numb = listG.at(i).uid;
    columnsTableMess ctm;
    ctm.uid = -1;
    ctm.login = login;
    ctm.operation = "Удаление";
    ctm.uidRecord = listG.at(i).uid;
    ctm.dateTime = QDateTime::currentDateTime().toString("hh:mm dd-MM-
yyyy");;
    cl->addGyrnal(ctm);
    listG.removeAt(i);
    setListen(listG);
    cl->deleteLetter(numb);

    curL = -1;
    ui->lineEdit->clear();
}

}

void Mail::on_pushButton_clicked() //создание
{

    createTable();
    colG = getTable();
    ui->pushButton_4->setHidden(false);
    ui->pushButton_5->setHidden(true);
    ui->pushButton_4->setStyleSheet("background: #b8c4cf");
    ui->lineEdit->clear();
    for (int i = 0; i < ui->tableWidget->rowCount(); i++){
        ui->tableWidget->item(i,0)->setBackgroundColor("#D8E6F3");
        ///////////////////////////////////
        ui->tableWidget->item(i,1)->setBackgroundColor("#D8E6F3");
        ///////////////////////////////////
    }
    //кнопки третьего столбца
    connect(butFile, &QPushButton::clicked, this, &Mail::createFile);
    connect(butS, &QPushButton::clicked, this, &Mail::butS1SlotC);
    connect(butS2, &QPushButton::clicked, this, &Mail::butS2SlotC);
    connect(butS3, &QPushButton::clicked, this, &Mail::butS3SlotC);
}

void Mail::on_pushButton_4_clicked() //сохранить
{
    ui->pushButton_4->setHidden(true);
    columnsTable table;
    QString str = "";
    QDate date;
    bool signE = true;
    for(int i = 0; i < ui->tableWidget->rowCount(); i++)
    {
        QString st = ui->tableWidget->item(i,1)->text();
        if(!st.isNull())
        {
            str = ui->tableWidget->item(i,2)->text();
            if(st == "Дата")
            {
                QLineEdit *l = (QLineEdit *)ui->tableWidget->cellWidget(i,2);
                str = l->text();
                if(!str.isNull() && str != "--")

```

```

{
    date = QDate::fromString(str, "dd-MM-yyyy");
    if(!date.isValid())
        date = QDate::fromString(str, "dd-MM-yy");

    if(!date.isValid()){
        signE = false;
        l->setText("");
    }
    else {
        switch (i) {
            case 0:
                table.date01 = date.toString("dd-MM-yyyy");
                break;
            case 2:
                table.date02 = date.toString("dd-MM-yyyy");
                break;
            case 3:
                table.date03 = date.toString("dd-MM-yyyy");
                break;
            case 6:
                table.date04 = date.toString("dd-MM-yyyy");
                break;
            case 7:
                table.date05 = date.toString("dd-MM-yyyy");
                break;
            case 8:
                table.date06 = date.toString("dd-MM-yyyy");
                break;
            case 11:
                table.date07 = date.toString("dd-MM-yyyy");
                break;
            case 17:
                table.date10 = date.toString("dd-MM-yyyy");
                break;
            case 18:
                table.date11 = date.toString("dd-MM-yyyy");
                break;
            case 21:
                table.date13 = date.toString("dd-MM-yyyy");
                break;
            case 26:
                table.date16 = date.toString("dd-MM-yyyy");
                break;
            case 27:
                table.date17 = date.toString("dd-MM-yyyy");
                break;
            case 29:
                table.date19 = date.toString("dd-MM-yyyy");
                break;
            case 33:
                table.date20 = date.toString("dd-MM-yyyy");
                break;
            case 35:
                table.date201 = date.toString("dd-MM-yyyy");
                break;
            case 38:
                table.date22 = date.toString("dd-MM-yyyy");
                break;
        }
    }
}

```

					ПГУ 1.090301.01.001 ПЗ	Лист
Изм.	Лист	№ докум.№	Подпись	Дата		113

```

    }
}
else if(st == "Homep")
{
    if(!str.isNull())
    {
        bool pr = true;
        for(int j = 0; j < str.length(); j++)
            if(!str[j].isDigit())
                pr = false;
        if(!pr)
        {
            ui->tableWidget->item(i,2)->setText("");
            signE = false;
        }
        else
        {
            switch (i) {
            case 1:
                table.number01 = str;
                break;
            case 4:
                table.number03 = str;
                break;
            case 5:
                table.number04 = str;
                break;
            case 9:
                table.number06 = str;
                break;
            case 10:
                table.number07 = str;
                break;
            case 12:
                table.number08 = str;
                break;
            case 16:
                table.number10 = str;
                break;
            case 19:
                table.number12 = str;
                break;
            case 20:
                table.number13 = str;
                break;
            case 32:
                table.number20 = str;
                break;
            case 34:
                table.number201 = str;
                break;
            case 37:
                table.number22 = str;
                break;
            }
        }
    }
}
else {
    switch (i) {
    case 13:

```

```

        table.tz09 = str;
        break;
    case 14:
        table.riperf10 = str;
        break;
    case 15:
        table.rireason10 = str;
        break;
    case 22:
        table.whommade13 = str;
        break;
    case 23:
        table.whomshow13 = str;
        break;
    case 24:
        table.product14 = str;
        break;
    case 25:
        table.detail15 = str;
        break;
    case 28:
        table.defectreason18 = str;
        break;
    case 30:
        table.riperf20 = str;
        break;
    case 31:
        table.rireason20 = str;
        break;
    case 36:
        table.actreason21 = str;
        break;
    }
}
}
}
table.uid = listG.size();
if(ui->lineEdit->text().length() != 0){
    table.name = ui->lineEdit->text();
    table.name.append("||");
    table.name.append(QDateTime::currentDateTime().toString("hh:mm:ss dd-
MM-yyyy"));
}
qDebug() << table.uid << table.name << table.date01
    << table.number01 << table.date02
    << table.date03 << table.number03
    << table.date04 << table.number04 << table.date05
    << table.date06 << table.number06 << table.date07
    << table.number07 << table.number08 << table.tz09
    << table.date10 << table.number10 << table.riperf10 <<
table.rireason10
    << table.date11 << table.number12 << table.date13
    << table.number13 << table.whommade13
    << table.whomshow13 << table.product14 << table.detail15
    << table.date16 << table.date17 << table.defectreason18
    << table.date19 << table.riperf20 << table.rireason20 <<
table.date20 << table.number20
    << table.date201 << table.number201 << table.actreason21
    << table.date22 << table.number22;
saveFile(table.tz09);
if(!signE)

```

```

{
    QMessageBox msgBox;
    msgBox.setIcon(QMessageBox::Critical);
    msgBox.setStyleSheet(styleBack);
    msgBox.setWindowTitle("Ошибка");
    msgBox.setText("Данные некорректны!");
    msgBox.setStandardButtons(QMessageBox::Yes);
    int ret = msgBox.exec();
    switch (ret) {
    case QMessageBox::Yes:
        break;
    }
}
else{
    closeTab();
    ui->lineEdit->clear();
    columnsTableMess ctm;
    ctm.uid = -1;
    ctm.login = login;
    ctm.operation = "Создание";
    ctm.uidRecord = table.uid;
    ctm.dateTime = QDateTime::currentDateTime().toString("hh:mm dd-MM-
yyyy");;
    cl->addGyrnal(ctm);
    cl->saveLetter(table);
}
}
void Mail::saveFile(QString path)
{
    QFile file(path);

    QDir dir(qApp->applicationDirPath() + "/files");
    dir.mkpath(qApp->applicationDirPath() + "/files");

    QString nameFile = dir.absolutePath() + "/" + getNameFormat(path);
    copyFile(nameFile);
    QFile::copy(path, nameFile);
}
void Mail::closeTab()
{
    ui->tableWidget->setColumnCount(0);
    ui->tableWidget->setRowCount(0);
    ui->tableWidget->clear();
    isClose = true;
}
void Mail::copyFile(QString nameFile)
{
    QFile file(nameFile);
    if(file.exists())
        file.remove();
}

columnsTable Mail::getTable()
{
    columnsTable table;
    QString str = "";
    QDate date;
    for(int i = 0; i < ui->tableWidget->rowCount(); i++)
    {
        QString st = ui->tableWidget->item(i,1)->text();
        if(!st.isNull())

```

```

{
    str = ui->tableWidget->item(i,2)->text();

    if(st == "Дата")
    {
        QLineEdit *l = (QLineEdit *)ui->tableWidget->cellWidget(i,2);
        str = l->text();
        if(!str.isNull() && str!="--")
        {
            date = QDate::fromString(str, "dd-MM-yyyy");
            if(!date.isValid())
                date = QDate::fromString(str, "dd-MM-yy");
            switch (i) {
            case 0:
                table.date01 = date.toString("dd-MM-yyyy");
                break;
            case 2:
                table.date02 = date.toString("dd-MM-yyyy");
                break;
            case 3:
                table.date03 = date.toString("dd-MM-yyyy");
                break;
            case 6:
                table.date04 = date.toString("dd-MM-yyyy");
                break;
            case 7:
                table.date05 = date.toString("dd-MM-yyyy");
                break;
            case 8:
                table.date06 = date.toString("dd-MM-yyyy");
                break;
            case 11:
                table.date07 = date.toString("dd-MM-yyyy");
                break;
            case 17:
                table.date10 = date.toString("dd-MM-yyyy");
                break;
            case 18:
                table.date11 = date.toString("dd-MM-yyyy");
                break;
            case 21:
                table.date13 = date.toString("dd-MM-yyyy");
                break;
            case 26:
                table.date16 = date.toString("dd-MM-yyyy");
                break;
            case 27:
                table.date17 = date.toString("dd-MM-yyyy");
                break;
            case 29:
                table.date19 = date.toString("dd-MM-yyyy");
                break;
            case 33:
                table.date20 = date.toString("dd-MM-yyyy");
                break;
            case 35:
                table.date201 = date.toString("dd-MM-yyyy");
                break;
            case 38:
                table.date22 = date.toString("dd-MM-yyyy");
                break;
            }
        }
    }
}

```

```

    }
}
else if(st == "Homep")
{
    if(!str.isNull() && str != "")
    {
        switch (i) {
            case 1:
                table.number01 = str;
                break;
            case 4:
                table.number03 = str;
                break;
            case 5:
                table.number04 = str;
                break;
            case 9:
                table.number06 = str;
                break;
            case 10:
                table.number07 = str;
                break;
            case 12:
                table.number08 = str;
                break;
            case 16:
                table.number10 = str;
                break;
            case 19:
                table.number12 = str;
                break;
            case 20:
                table.number13 = str;
                break;
            case 32:
                table.number20 = str;
                break;
            case 34:
                table.number201 = str;
                break;
            case 37:
                table.number22 = str;
                break;
        }
    }
}
else {
    switch (i) {
        case 13:
            table.tz09 = str;
            break;
        case 14:
            table.riperf10 = str;
            break;
        case 15:
            table.rireason10 = str;
            break;
        case 22:
            table.whommadel3 = str;
            break;
    }
}

```

```

        case 23:
            table.whomshow13 = str;
            break;
        case 24:
            table.product14 = str;
            break;
        case 25:
            table.detail15 = str;
            break;
        case 28:
            table.defectreason18 = str;
            break;
        case 30:
            table.riperf20 = str;
            break;
        case 31:
            table.rireason20 = str;
            break;
        case 36:
            table.actreason21 = str;
            break;
    }
}

}

if(ui->listWidget->currentRow() != -1)
    table.uid = curL;
else
    table.uid = listG.size();
table.name = ui->lineEdit->text();
return table;
}

bool Mail::equals(columnsTable tab1, columnsTable tab2)
{
    qDebug() << tab1.uid << tab1.name << tab1.date01 << tab1.number01 <<
    tab1.date02 << tab1.date03 << tab1.number03
        << tab1.date04 << tab1.number04 << tab1.date05 <<
    tab1.date06 << tab1.number06 << tab1.date07
        << tab1.number07 << tab1.number08 << tab1.tz09 <<
    tab1.date10 << tab1.number10 << tab1.riperf10 << tab1.rireason10
        << tab1.date11 << tab1.number12 << tab1.date13 <<
    tab1.number13 << tab1.whommade13
        << tab1.whomshow13 << tab1.product14 << tab1.detail15 <<
    tab1.date16 << tab1.date17 << tab1.defectreason18
        << tab1.date19 << tab1.riperf20 << tab1.rireason20 <<
    tab1.date20 << tab1.number20 << tab1.date201 << tab1.number201 <<
    tab1.actreason21
        << tab1.date22 << tab1.number22;
    qDebug() << tab2.uid << tab2.name << tab2.date01 << tab2.number01 <<
    tab2.date02 << tab2.date03 << tab2.number03
        << tab2.date04 << tab2.number04 << tab2.date05 <<
    tab2.date06 << tab2.number06 << tab2.date07
        << tab2.number07 << tab2.number08 << tab2.tz09 <<
    tab2.date10 << tab2.number10 << tab2.riperf10 << tab2.rireason10
        << tab2.date11 << tab2.number12 << tab2.date13 <<
    tab2.number13 << tab2.whommade13
        << tab2.whomshow13 << tab2.product14 << tab2.detail15 <<
    tab2.date16 << tab2.date17 << tab2.defectreason18

```

```

        << tab2.date19 << tab2.riperf20 << tab2.rireason20 <<
tab2.date20 << tab2.number20 << tab2.date201 << tab2.number201 <<
tab2.actreason21
        << tab2.date22 << tab2.number22;
bool pr = true;

if (tab1.name != tab2.name)
    pr = false;
else if (tab1.date01 != tab2.date01)
    pr = false;
else if (tab1.number01 != tab2.number01)
    pr = false;
else if (tab1.date02 != tab2.date02)
    pr = false;
else if (tab1.date03 != tab2.date03)
    pr = false;
else if (tab1.number03 != tab2.number03)
    pr = false;
else if (tab1.date04 != tab2.date04)
    pr = false;
else if (tab1.number04 != tab2.number04)
    pr = false;
else if (tab1.date05 != tab2.date05)
    pr = false;
else if (tab1.date06 != tab2.date06)
    pr = false;
else if (tab1.number06 != tab2.number06)
    pr = false;
else if (tab1.date07 != tab2.date07)
    pr = false;
else if (tab1.number06 != tab2.number06)
    pr = false;
else if (tab1.number07 != tab2.number07)
    pr = false;
else if (tab1.number08 != tab2.number08)
    pr = false;
else if (tab1.tz09 != tab2.tz09)
    pr = false;
else if (tab1.date10 != tab2.date10)
    pr = false;
else if (tab1.number10 != tab2.number10)
    pr = false;
else if (tab1.riperf10 != tab2.riperf10)
    pr = false;
else if (tab1.rireason10 != tab2.rireason10)
    pr = false;
else if (tab1.date11 != tab2.date11)
    pr = false;
else if (tab1.number12 != tab2.number12)
    pr = false;
else if (tab1.date13 != tab2.date13)
    pr = false;
else if (tab1.number13 != tab2.number13)
    pr = false;
else if (tab1.whommade13 != tab2.whommade13)
    pr = false;
else if (tab1.whomshow13 != tab2.whomshow13)
    pr = false;
else if (tab1.product14 != tab2.product14)
    pr = false;
else if (tab1.detail15 != tab2.detail15)

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						120
Изм.	Лист	№ докум.№	Подпись	Дата		

```

        pr = false;
    else if (tab1.date16 != tab2.date16)
        pr = false;
    else if (tab1.date17 != tab2.date17)
        pr = false;
    else if (tab1.defectreason18 != tab2.defectreason18)
        pr = false;
    else if (tab1.date20 != tab2.date20)
        pr = false;
    else if (tab1.number20 != tab2.number20)
        pr = false;
    else if (tab1.date201 != tab2.date201)
        pr = false;
    else if (tab1.number201 != tab2.number201)
        pr = false;
    else if (tab1.actreason21 != tab2.actreason21)
        pr = false;
    else if (tab1.date22 != tab2.date22)
        pr = false;
    else if (tab1.number22 != tab2.number22)
        pr = false;
    qDebug() << pr;
    return pr;
}
void Mail::on_pushButton_5_clicked()//сохранить - обновить
{
    ui->pushButton_5->setHidden(true);
    columnsTable table;
    QString str = "";
    QDate date;
    bool signE = true;
    for(int i = 0; i < ui->tableWidget->rowCount(); i++)
    {
        QString st = ui->tableWidget->item(i,1)->text();
        if(!st.isNull())
        {
            str = ui->tableWidget->item(i,2)->text();
            if(st == "Дата")
            {
                QLineEdit *l = (QLineEdit *)ui->tableWidget->cellWidget(i,2);
                str = l->text();

                if(!str.isNull() && str!="--")
                {
                    date = QDate::fromString(str, "dd-MM-yyyy");
                    if(!date.isValid())
                        date = QDate::fromString(str, "dd-MM-yy");

                    if(!date.isValid())
                    {
                        //ui->tableWidget->item(i,2)->setText("неправильные
данные");

                        signE = false;
                        l->setText("");
                    }
                    else {
                        switch (i) {
                            case 0:
                                table.date01 = date.toString("dd-MM-yyyy");
                                break;
                            case 2:

```

```

        table.date02 = date.toString("dd-MM-yyyy");
        break;
    case 3:
        table.date03 = date.toString("dd-MM-yyyy");
        break;
    case 6:
        table.date04 = date.toString("dd-MM-yyyy");
        break;
    case 7:
        table.date05 = date.toString("dd-MM-yyyy");
        break;
    case 8:
        table.date06 = date.toString("dd-MM-yyyy");
        break;
    case 11:
        table.date07 = date.toString("dd-MM-yyyy");
        break;
    case 17:
        table.date10 = date.toString("dd-MM-yyyy");
        break;
    case 18:
        table.date11 = date.toString("dd-MM-yyyy");
        break;
    case 21:
        table.date13 = date.toString("dd-MM-yyyy");
        break;
    case 26:
        table.date16 = date.toString("dd-MM-yyyy");
        break;
    case 27:
        table.date17 = date.toString("dd-MM-yyyy");
        break;
    case 29:
        table.date19 = date.toString("dd-MM-yyyy");
        break;
    case 33:
        table.date20 = date.toString("dd-MM-yyyy");
        break;
    case 35:
        table.date201 = date.toString("dd-MM-yyyy");
        break;
    case 38:
        table.date22 = date.toString("dd-MM-yyyy");
        break;
    }
    }
}
else if(st == "Homep")
{
    if(!str.isNull() && str != "")
    {
        bool pr = true;

        for(int j = 0; j < str.length(); j++)
            if(!str[j].isDigit())
                pr = false;

        if(!pr)
        {
            ui->tableWidget->item(i,2)->setText("");
            signE = false;
        }
    }
}

```

```

    }
    else
    {
        switch (i) {
            case 1:
                table.number01 = str;
                break;
            case 4:
                table.number03 = str;
                break;
            case 5:
                table.number04 = str;
                break;
            case 9:
                table.number06 = str;
                break;
            case 10:
                table.number07 = str;
                break;
            case 12:
                table.number08 = str;
                break;
            case 16:
                table.number10 = str;
                break;
            case 19:
                table.number12 = str;
                break;
            case 20:
                table.number13 = str;
                break;
            case 32:
                table.number20 = str;
                break;
            case 34:
                table.number201 = str;
                break;
            case 37:
                table.number22 = str;
                break;
        }
    }
}
else {
    switch (i) {
        case 13:
            table.tz09 = str;
            break;
        case 14:
            table.riperfl0 = str;
            break;
        case 15:
            table.rireason10 = str;
            break;
        case 22:
            table.whommade13 = str;
            break;
        case 23:
            table.whomshow13 = str;
            break;
    }
}

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						123
Изм.	Лист	№ докум.№	Подпись	Дата		

```

        case 24:
            table.product14 = str;
            break;
        case 25:
            table.detail15 = str;
            break;
        case 28:
            table.defectreason18 = str;
            break;
        case 30:
            table.riperf20 = str;
            break;
        case 31:
            table.rireason20 = str;
            break;
        case 36:
            table.actreason21 = str;
            break;
    }
}

}

table.uid = listG.at(curL).uid;
if(ui->lineEdit->text().length() != 0)
{
    table.name = ui->lineEdit->text();
    table.name.append("||");
    table.name.append(QDateTime::currentDateTime().toString("hh:mm:ss dd-
MM-yyyy"));
}
saveFile(table.tz09);
if(!signE)
{
    ui->pushButton_5->setHidden(false);
    QMessageBox msgBox;
    msgBox.setIcon(QMessageBox::Critical);
    msgBox.setStyleSheet(styleBack);
    msgBox.setWindowTitle("Ошибка");
    msgBox.setText("Данные некорректны!");
    msgBox.setStandardButtons(QMessageBox::Yes);
    int ret = msgBox.exec();
    switch (ret) {
        case QMessageBox::Yes:
            break;
    }
}
else{
    ui->lineEdit->clear();
    closeTab();
    columnsTableMess ctm;
    ctm.uid = -1;
    ctm.login = login;
    ctm.operation = "Изменение";
    ctm.uidRecord = table.uid;
    ctm.dateTime = QDateTime::currentDateTime().toString("hh:mm dd-MM-
yyyy");
    cl->addGyrnal(ctm);
    cl->updateLetter(table);
}
}

```

```

void Mail::createFile(){
    qDebug() << "Create_File";
    QFileDialog fd;
    QString nameDoc = QFileDialog::getOpenFileName(this,tr("Open File"),
"/home", tr("Full (*.*)"));

    if(nameDoc != ""){
        ui->tableWidget->item(13,2)->setText(nameDoc);
        nameDoc = getName(nameDoc);
        butFile->setText(nameDoc);
    }
    qDebug() <<nameDoc;
}

void Mail::deleteFile()
{
    ui->tableWidget->item(13,2)->setText("");
    butFile->setText("Открыть файл");
}

void Mail::mesFile()
{
    QWidget *w = new QWidget();
    w->setStyleSheet(styleBack);
    w->setWindowTitle("Файл");
    w->setMaximumSize(500,100);
    w->setMinimumSize(500,100);
    QLabel *l1 = new QLabel("Выберете действие с файлом:", this);
    l1->setStyleSheet("font: 16px");
    l1->setAlignment(Qt::AlignCenter);
    QPushButton *b1 = new QPushButton("Открыть", this);
    QPushButton *b2 = new QPushButton("Создать", this);
    QPushButton *b3 = new QPushButton("Удалить", this);
    b1->setStyleSheet("font: 13px");
    b2->setStyleSheet("font: 13px");
    b3->setStyleSheet("font: 13px");
    QVBoxLayout *vbl = new QVBoxLayout(w);

    vbl->addWidget(l1);
    QGridLayout *l = new QGridLayout();
    l->addWidget(b1,1,0);
    l->addWidget(b2,1,1);
    l->addWidget(b3,1,2);
    vbl->addLayout(l);
    connect(b1, &QPushButton::clicked, this, &Mail::openFile);
    connect(b2, &QPushButton::clicked, this, &Mail::createFile);
    connect(b3, &QPushButton::clicked, this, &Mail::deleteFile);
    w->show();
}

void Mail::openFile(){
    qDebug() << "Open_File";
    qDebug() << butFile->text();
    int num = getId(ui->listWidget->currentItem()->text(),ui->listWidget-
>currentItem()->whatsThis());
    columnsTable table = listG.at(num);
    QString str = table.tz09;
    if(!QDesktopServices::openUrl(QUrl::fromLocalFile(str)))
    {
        QDir dir(qApp->applicationDirPath() + "/files");
    }
}

```

```

        str = dir.absolutePath() + "/" + getNameFormat(str);
        if(!QDesktopServices::openUrl(QUrl::fromLocalFile(str)))
        {
            QMessageBox msgBox;
            msgBox.setIcon(QMessageBox::Critical);
            msgBox.setWindowTitle("Ошибка");
            msgBox.setStyleSheet(styleBack);
            msgBox.setText("Файл не найден!");
            msgBox.exec();
        }
    }
    qDebug() << str;
}

void Mail::on_pushButton_3_clicked()
{
    cl->show();
}

void Mail::butS1SlotE()
{
    int numb = ui->listWidget->currentRow();
    columnsTable table = listG.at(numb);
    fs = new FullSpisok(0, numb, ui->tableWidget->item(24,2)->text());
    connect(fs, &FullSpisok::signSave, this, &Mail::saveSignSlots);
}

void Mail::butS2SlotE()
{
    int numb = ui->listWidget->currentRow();
    columnsTable table = listG.at(numb);
    fs = new FullSpisok(1, numb, ui->tableWidget->item(25,2)->text());
    connect(fs, &FullSpisok::signSave, this, &Mail::saveSignSlots);
}

void Mail::butS3SlotE()
{
    int numb = ui->listWidget->currentRow();
    columnsTable table = listG.at(numb);
    fs = new FullSpisok(2, numb, ui->tableWidget->item(36,2)->text());
    connect(fs, &FullSpisok::signSave, this, &Mail::saveSignSlots);
}

void Mail::butS1SlotC()
{
    int numb = ui->listWidget->currentRow();
    fs = new FullSpisok(0, numb, "");
    connect(fs, &FullSpisok::signSave, this, &Mail::saveSignSlots);
}

void Mail::butS2SlotC()
{
    int numb = ui->listWidget->currentRow();
    fs = new FullSpisok(1, numb, "");
    connect(fs, &FullSpisok::signSave, this, &Mail::saveSignSlots);
}

void Mail::butS3SlotC()
{
    int numb = ui->listWidget->currentRow();
    fs = new FullSpisok(2, numb, "");
    connect(fs, &FullSpisok::signSave, this, &Mail::saveSignSlots);
}

```

```

void Mail::saveSignSlots(int reg, int numb, QString words)
{
    qDebug() << "saveSlots" << words;
    switch (reg) {
    case 0:
        ui->tableWidget->item(24, 2)->setText(words);
        break;
    case 1:
        ui->tableWidget->item(25, 2)->setText(words);
        break;
    case 2:
        ui->tableWidget->item(36, 2)->setText(words);
        break;
    }
}

void Mail::keyPressEvent(QKeyEvent *event)
{
    switch (event->key()) {
    case Qt::Key_Delete:
        qDebug() << "Key Delete";
        if(ui->tableWidget->columnCount() != 0 && ui->tableWidget-
>currentItem() && ui->tableWidget->currentColumn() == 2)
            ui->tableWidget->currentItem()->setText("");
        break;
    }
}

void Mail::on_comboBox_currentIndexChanged(const QString &name) //сортировка
{
    qDebug() << "on_comboBox_currentIndexChanged" << name;

    if(ui->listWidget->currentRow() == -1)
    {
        if(name == "Дата изменения")
            setListen(listG);
        else if(name == "А-Я")
            ui->listWidget->sortItems(Qt::AscendingOrder);
    }
    else
        if(on_listWidget_prov(ui->listWidget->currentItem()))
        {
            closeTab();
            if(name == "Дата изменения")
                setListen(listG);
            else if(name == "А-Я")
                ui->listWidget->sortItems(Qt::AscendingOrder);
        }
}

void Mail::on_pushButton_6_clicked() //журнал
{
    Gyrnal *gr = new Gyrnal(nullptr, cl);
    gr->show();
}

```

Файл «main.cpp»

```

#include <QtWidgets>
#include "mail.h"
#include <QApplication>
#include <login.h>

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						127
Изм.	Лист	№ докум.№	Подпись	Дата		

```
#include <gyrnal.h>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    Login l;
    return app.exec();
}
```

Файл «menu.cpp»

```
#include "menu.h"
#include "ui_menu.h"

Menu::Menu(QWidget *parent, Client *cl, QString login) :
    QWidget(parent),
    ui(new Ui::Menu)
{
    ui->setupUi(this);
    this->cl = cl;
    this->loginPub = login;
    setWindowTitle(loginPub);
    ui->widget->setStyleSheet("background: #D8E6F3");
    ui->listWidget->setStyleSheet("background: #D8E6F3");
    ui->listWidget->item(0)->setBackgroundColor("#b8c4cf");
    ui->listWidget->setEditTriggers(QAbstractItemView::NoEditTriggers);
    ui->listWidget->setFocusPolicy(Qt::NoFocus);
    ui->listWidget->setSelectionMode(QAbstractItemView::NoSelection);
    box = new QHBoxLayout(ui->widget);
    ml = new Mail(nullptr, cl, loginPub);
    ch = new Chat(nullptr, cl, loginPub);
    box->addWidget(ml);
}

Menu::~Menu()
{
    delete ui;
}

void Menu::on_listWidget_itemClicked(QListWidgetItem *item)
{
    delete box;
    box = new QHBoxLayout(ui->widget);

    delete ml;
    delete ch;
    ml = new Mail(nullptr, cl, loginPub);
    ch = new Chat(nullptr, cl, loginPub);

    if(item->text() == "    Рабочее пространство ") {
        ui->listWidget->item(0)->setBackgroundColor("#b8c4cf");
        ui->listWidget->item(1)->setBackgroundColor("#D8E6F3");
        box->addWidget(ml);
    }
    else if(item->text() == "    Сообщения ") {
        ui->listWidget->item(0)->setBackgroundColor("#D8E6F3");
        ui->listWidget->item(1)->setBackgroundColor("#b8c4cf");
        box->addWidget(ch);
    }
}
```

Файл «dbclaims.cpp»

```
#include "dbclaims.h"

DBClaims::DBClaims(QWidget *parent) : QWidget(parent)
{
    logPassBool = true;//false;
    messBool = true;//false;
    QString fileSettings = qApp->applicationDirPath() +
"/set/settingsDB.ini";
    settings = new QSettings(fileSettings, QSettings::IniFormat, this);
    // qDebug() << "/123455454456/" + settings->value("FILEDB",
"dbclaims.db").toString();
    if (settings->value("FILEDB", "99").toString() == "99") settings-
>setValue("FILEDB", "dbclaims.db");
    createDB();
    createTables();
}

DBClaims::~DBClaims()
{
}

bool DBClaims::addRecordInTable(columnsTable _ct)
{
    bool rez = FALSE;
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return rez;
        }
    QList<int> li;
    li.clear();
    li = selectIdAllRecordsIntables();
    columnsTable ct;
    ct = _ct;
    if(li.contains(ct.uid) || (ct.uid <= 0))
    {
        if(li.isEmpty())
        {
            ct.uid = 1;
        }
        else
        {
            ct.uid = li.last() + 1;
        }
    }
    QString stringTempNames;
    stringTempNames.clear();
    stringTempNames += "(" + constNamesColumnsTable[0];
    for(int i = 1; i < constCountColumnsTable; i++)
        stringTempNames += ", " + constNamesColumnsTable[i];
    stringTempNames += ")";

    QString stringTempData;
    stringTempData.clear();
    QStringList sl = structToStringlist(ct);
    stringTempData += tr("(%1").arg(sl.at(0));
    for(int i = 1; i < sl.size(); i++)
        stringTempData += tr(", '%1'").arg(sl.at(i));
```

```

        stringTempData += tr(")");

QString queryRequestString = "";
queryRequestString.clear();
queryRequestString =
    tr("INSERT INTO %1 %2 VALUES %3")
        .arg(constNameTable1)
        .arg(stringTempNames)
        .arg(stringTempData);
QSqlQuery query(dbc);
rez = query.exec(tr("%1").arg(queryRequestString));
return rez;
}

bool DBClaims::dbIsOpen()
{
    bool rez = dbc.isOpen();
    return rez;
}

bool DBClaims::removeRecordInTable(columnsTable _ct)
{
    bool rez = FALSE;
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return rez;
        }
    QSqlQuery query(dbc);
    rez = query.exec(tr("DELETE FROM %1 WHERE %2 =
%3").arg(constNameTable1).arg(constNamesColumnsTable[0]).arg(_ct.uid));
    return rez;
}

bool DBClaims::removeRecordInTable(int id_)
{
    bool rez = FALSE;
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return rez;
        }
    QSqlQuery query(dbc);
    rez = query.exec(tr("DELETE FROM %1 WHERE %2 =
%3").arg(constNameTable1).arg(constNamesColumnsTable[0]).arg(id_));
    return rez;
}

QList<int> DBClaims::selectIdAllRecordsIntables()
{
    QList<int> listId;
    listId.clear();
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return listId;
        }
    QSqlQuery query(dbc);

```

```

        QString queryRequestString = "";
        queryRequestString = tr("SELECT %1 FROM
%2").arg(constNamesColumnsTable[0]).arg(constNameTable1);
        if(!query.exec(queryRequestString))
            return listId;
        while(query.next())
        {
            listId.append(query.value(0).toInt());
        }
        return listId;
    }

QList<columnsTable> DBClaims::selectAllRecordsInTable()
{
    QList<columnsTable> listRecords;
    listRecords.clear();
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return listRecords;
        }
    QSqlQuery query(dbc);
    columnsTable ct;
    query.exec("SELECT * FROM " + constNameTable1);
    while(query.next())
    {
        ct = slotClearStructTables();
        if(query.record().count() == constCountColumnsTable)
        {
            QStringList sl;
            sl.clear();
            for(int i = 0; i < query.record().count(); i++)
                sl.append(query.value(i).toString());
            ct = stringlistToStruct(sl);
            listRecords.append(ct);
        }
    }
    return listRecords;
}

columnsTable DBClaims::selectRecordInTables(int rec)
{
    columnsTable ct;
    ct = slotClearStructTables();
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return ct;
        }
    QSqlQuery query(dbc);
    query.exec(tr("SELECT * FROM %1 WHERE %2=%3
").arg(constNameTable1).arg(constNamesColumnsTable[0]).arg(rec));
    while(query.next())
    {
        if(query.record().count() == constCountColumnsTable)
        {
            QStringList sl;
            sl.clear();
            for(int i = 0; i < query.record().count(); i++)

```

```

        sl.append(query.value(i).toString());
        ct = stringlistToStruct(sl);
    }
}
return ct;
}

bool DBClaims::updateRecordInTable(columnsTable ct_)
{
    bool rez = FALSE;
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return rez;
        }
    if(!selectIdAllRecordsIntables().contains(ct_.uid))
        return rez;
    QSqlQuery query(dbc);
    QString queryRequestString;
    QString stringTempData;
    stringTempData.clear();
    QStringList sl = structToStringlist(ct_);
    stringTempData += tr("%1=%2",
    ").arg(constNamesColumnsTable[0]).arg(sl.at(0));
    for(int i = 1; i < sl.size(); i++)
    {
        if((i + 1) < sl.size())
        {
            stringTempData += tr("%1='%2'",
            ").arg(constNamesColumnsTable[i]).arg(sl.at(i));
        }
        else
        {
            stringTempData +=
            tr("%1='%2'").arg(constNamesColumnsTable[i]).arg(sl.at(i));
        }
    }
    queryRequestString.clear();
    queryRequestString =
        tr("UPDATE %1 SET %2 WHERE %3=%4")
        .arg(constNameTable1)
        .arg(stringTempData)
        .arg(constNamesColumnsTable[0])
        .arg(ct_.uid);
    rez = query.exec(queryRequestString);
    return rez;
}

bool DBClaims::addRecordInLogPass(columnsTableLogPass _ct)
{
    bool rez = FALSE;
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return rez;
        }
    QList<int> li;
    li.clear();
    li = selectIdAllRecordsIntableLogPass();

```

```

columnsTableLogPass ct;
ct = _ct;
if(li.contains(ct.uid) || (ct.uid <= 0))
{
    if(li.isEmpty())
    {
        ct.uid = 1;
    }
    else
    {
        ct.uid = li.last() + 1;
    }
}
QString stringTempNames;
stringTempNames.clear();
stringTempNames += "(" + constNamesColumnsTableLogPass[0];
for(int i = 1; i < constCountColumnsTableLogPass; i++)
    stringTempNames += ", " + constNamesColumnsTableLogPass[i];
stringTempNames += ")";

QString stringTempData;
stringTempData.clear();
QStringList sl = structToStringlistLogPass(ct);
stringTempData += tr("%1").arg(sl.at(0));
for(int i = 1; i < sl.size(); i++)
    stringTempData += tr(", '%1'").arg(sl.at(i));
stringTempData += tr(")");
QString queryRequestString = "";
queryRequestString.clear();
queryRequestString =
    tr("INSERT INTO %1 %2 VALUES %3")
    .arg(constNameTable1LogPass)
    .arg(stringTempNames)
    .arg(stringTempData);

QSqlQuery query(dbc);
rez = query.exec(tr("%1").arg(queryRequestString));
return rez;
}

QList<int> DBClaims::selectIdAllRecordsIntableLogPass()
{
    QList<int> listId;
    listId.clear();
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return listId;
        }
    QSqlQuery query(dbc);
    QString queryRequestString = "";
    queryRequestString = tr("SELECT %1 FROM
%2").arg(constNamesColumnsTableLogPass[0]).arg(constNameTable1LogPass);
    if(!query.exec(queryRequestString))
        return listId;
    while(query.next())
    {
        listId.append(query.value(0).toInt());
    }
    return listId;
}

```

```

bool DBClaims::removeRecordInLogPass(columnsTableLogPass _ct)
{
    bool rez = FALSE;
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return rez;
        }
    QSqlQuery query(dbc);
    rez = query.exec(tr("DELETE FROM %1 WHERE %2 =
%3").arg(constNameTable1LogPass).arg(constNamesColumnsTableLogPass[0]).arg(_c
t.uid));
    return rez;
}

bool DBClaims::updateRecordInLogPass(columnsTableLogPass ct_)
{
    bool rez = FALSE;
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return rez;
        }
    if(!selectIdAllRecordsIntableLogPass().contains(ct_.uid))
        return rez;
    QSqlQuery query(dbc);
    QString queryRequestString;
    QString stringTempData;
    stringTempData.clear();
    QStringList sl = structToStringlistLogPass(ct_);
    stringTempData += tr("%1=%2,
").arg(constNamesColumnsTableLogPass[0]).arg(sl.at(0));
    for(int i = 1; i < sl.size(); i++)
    {
        if((i + 1) < sl.size())
        {
            stringTempData += tr("%1='%2',
").arg(constNamesColumnsTableLogPass[i]).arg(sl.at(i));
        }
        else
        {
            stringTempData +=
tr("%1='%2'").arg(constNamesColumnsTableLogPass[i]).arg(sl.at(i));
        }
    }
    queryRequestString.clear();
    queryRequestString =
        tr("UPDATE %1 SET %2 WHERE %3=%4")
        .arg(constNameTable1LogPass)
        .arg(stringTempData)
        .arg(constNamesColumnsTableLogPass[0])
        .arg(ct_.uid);
    rez = query.exec(queryRequestString);
    return rez;
}

QList<columnsTableLogPass> DBClaims::selectAllRecordsInLogPass()
{

```

```

QList<columnsTableLogPass> listRecords;
listRecords.clear();
if(!dbc.isOpen())
    if(!dbc.open())
    {
        qDebug() << "Ошибка : БД не открыта ";
        return listRecords;
    }
QSqlQuery query(dbc);
columnsTableLogPass ct;
query.exec("SELECT * FROM " + constNameTable1LogPass);
while(query.next())
{
    ct.uid = 0;
    ct.login.clear();
    ct.password.clear();
    if(query.record().count() == constCountColumnsTableLogPass)
    {
        QStringList sl;
        sl.clear();
        for(int i = 0; i < query.record().count(); i++)
            sl.append(query.value(i).toString());
        ct = stringlistToStructLogPass(sl);

        listRecords.append(ct);
    }
}
return listRecords;
}

columnsTableLogPass DBClaims::selectRecordInLogPass(QString logstr)
{
    columnsTableLogPass ct;
    ct.uid = 0;
    ct.login.clear();
    ct.password.clear();
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return ct;
        }
    QSqlQuery query(dbc);
    query.exec(tr("SELECT * FROM %1 WHERE %2='%3'")
        .arg(constNameTable1LogPass).arg(constNamesColumnsTableLogPass[1]).arg(logs
        tr));
    while(query.next())
    {
        if(query.record().count() == constCountColumnsTableLogPass)
        {
            QStringList sl;
            sl.clear();
            for(int i = 0; i < query.record().count(); i++)
                sl.append(query.value(i).toString());
            ct = stringlistToStructLogPass(sl);
        }
    }
    return ct;
}

bool DBClaims::addRecordInMess(columnsTableMess _ct)

```

```

{
    bool rez = FALSE;
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return rez;
        }
    QList<int> li;
    li.clear();
    li = selectIdAllRecordsIntableMess();
    columnsTableMess ct;
    ct = _ct;
    if(li.contains(ct.uid) || (ct.uid <= 0))
    {
        if(li.isEmpty())
        {
            ct.uid = 1;
        }
        else
        {
            ct.uid = li.last() + 1;
        }
    }
    QString stringTempNames;
    stringTempNames.clear();
    stringTempNames += "(" + constNamesColumnsTableMess[0];
    for(int i = 1; i < constCountColumnsTableMess; i++)
        stringTempNames += ", " + constNamesColumnsTableMess[i];
    stringTempNames += ")";

    QString stringTempData;
    stringTempData.clear();
    QStringList sl = structToStringlistMess(ct);
    stringTempData += tr("(%1").arg(sl.at(0));
    for(int i = 1; i < sl.size(); i++)
        stringTempData += tr(", %1").arg(sl.at(i));
    stringTempData += tr(")");
    QString queryRequestString = "";
    queryRequestString.clear();
    queryRequestString =
        tr("INSERT INTO %1 %2 VALUES %3")
        .arg(constNameTable1Mess)
        .arg(stringTempNames)
        .arg(stringTempData);

    QSqlQuery query(dbc);
    rez = query.exec(tr("%1").arg(queryRequestString));
    return rez;
}

QList<int> DBClaims::selectIdAllRecordsIntableMess()
{
    QList<int> listId;
    listId.clear();
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return listId;
        }
}

```

```

        QSqlQuery query(dbc);
        QString queryRequestString = "";
        queryRequestString = tr("SELECT %1 FROM
%2").arg(constNamesColumnsTableMess[0]).arg(constNameTable1Mess);
        if(!query.exec(queryRequestString))
            return listId;
        while(query.next())
        {
            listId.append(query.value(0).toInt());
        }
        return listId;
    }

QList<columnsTableMess> DBClaims::selectAllRecordsInMess()
{
    QList<columnsTableMess> listRecords;
    listRecords.clear();
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return listRecords;
        }
    QSqlQuery query(dbc);
    columnsTableMess ct;
    query.exec("SELECT * FROM " + constNameTable1Mess);
    while(query.next())
    {
        ct.uid = 0;
        ct.dateTime.clear();
        ct.login.clear();
        ct.uidRecord = 0;;
        ct.operation.clear();
        if(query.record().count() == constCountColumnsTableMess)
        {
            QStringList sl;
            sl.clear();
            for(int i = 0; i < query.record().count(); i++)
                sl.append(query.value(i).toString());
            ct = stringlistToStructMess(sl);
            listRecords.append(ct);
        }
    }
    return listRecords;
}

bool DBClaims::createDB()
{
    bool rez = FALSE;
    QFile fi;
    fi.setFileName(qApp->applicationDirPath() + "/db/" + settings-
>value("FILEDB", "dbclaims.db").toString());
    qDebug() << "DBClaims::createDB() : БД открыта - " << fi.isOpen() << ",
существует - " << fi.exists() << ", имя(путь) - " << fi.fileName();
    QDir dir(qApp->applicationDirPath() + "/db");
    dir.mkpath(qApp->applicationDirPath() + "/db");
    dbc = QSqlDatabase::addDatabase("QSQLITE");
    qDebug() << "dbc.drivers() " << dbc.drivers();
    dbc.setDatabaseName(fi.fileName());
    if(!fi.exists())
    {

```

```

        qDebug() << "DBClaims::createDB() - " << "файл БД не найден";
        dbc.setDatabaseName(fi.fileName());
        rez = dbc.open();
        if(rez)
            qDebug() << "БД создана ";
        else
            qDebug() << "БД не создана ";
        dbc.close();
    }
    else
    {
        QFileInfo inf(fi);
        qDebug() << tr("БД %1 существует").arg(inf.fileName());
    }
    return rez;
}

bool DBClaims::createTables()
{
    bool rez = FALSE;
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return rez;
        }
    QSqlQuery query(dbc);
    if(!dbc.tables().contains(constNameTable1))
    {
        QString stringQuery = "";
        stringQuery = "create table " + constNameTable1 + " (" ;
        stringQuery += constNamesColumnsTable[0] + " " +
constTypesColumnsTable[0];
        for(int i = 1; i < constCountColumnsTable; i++)
            stringQuery += ", " + constNamesColumnsTable[i] + " " +
constTypesColumnsTable[i];
        stringQuery += ")";
        rez = query.exec(stringQuery);
        if (rez) qDebug() << tr("таблица %1 создана").arg(constNameTable1);
        else qDebug() << tr("таблица %1 не создана").arg(constNameTable1);
    }
    else
    {
        qDebug() << "Таблица " << constNameTable1 << "существует";
    }
    if(logPassBool)
    {
        if(!dbc.tables().contains(constNameTable1LogPass))
        {
            QString stringQuery = "";
            stringQuery = "create table " + constNameTable1LogPass + " (" ;
            stringQuery += constNamesColumnsTableLogPass[0] + " " +
constTypesColumnsTableLogPass[0];
            for(int i = 1; i < constCountColumnsTableLogPass; i++)
                stringQuery += ", " + constNamesColumnsTableLogPass[i] + " " +
+ constTypesColumnsTableLogPass[i];
            stringQuery += ")";
            rez = query.exec(stringQuery);
            if (rez) qDebug() << tr("таблица %1
создана").arg(constNameTable1LogPass);

```

```

        else qDebug() << tr("таблица %1 не
создана").arg(constNameTable1LogPass);
    }
    else
    {
        qDebug() << "Таблица " << constNameTable1LogPass << "существует";
    }
}
if(messBool)
{
    if(!dbc.tables().contains(constNameTable1Mess))
    {
        QString stringQuery = "";
        stringQuery = "create table " + constNameTable1Mess + " (";
        stringQuery += constNamesColumnsTableMess[0] + " " +
constTypesColumnsTableMess[0];
        for(int i = 1; i < constCountColumnsTableMess; i++)
            stringQuery += ", " + constNamesColumnsTableMess[i] + " " +
constTypesColumnsTableMess[i];
        stringQuery += ")";
        rez = query.exec(stringQuery);
        if (rez) qDebug() << tr("таблица %1
создана").arg(constNameTable1Mess);
        else qDebug() << tr("таблица %1 не
создана").arg(constNameTable1Mess);
    }
    else
    {
        qDebug() << "Таблица " << constNameTable1Mess << "существует";
    }
}
qDebug() << "Таблицы : " << dbc.tables();
return rez;
}

void DBClaims::selectAllRecordsTables()
{
    if(!dbc.isOpen())
        if(!dbc.open())
        {
            qDebug() << "Ошибка : БД не открыта ";
            return ;
        }
    QSqlQuery query(dbc);
    query.exec("SELECT " + constNamesColumnsTable[0] + " FROM " +
constNameTable1);
    int counts = 0;
    while(query.next())
    {
        counts++;
    }
    query.exec("SELECT * FROM " + constNameTable1);
    QString rezstr = tr("Результат, количество строк : %1").arg(counts);
    qDebug() << rezstr;
    while(query.next())
    {
        rezstr = "";
        for(int i = 0; i < query.record().count(); i++)
            rezstr += " : " + query.value(i).toString();
        qDebug() << rezstr;
    }
}

```

```

}

columnsTable DBClaims::slotClearStructTables()
{
    columnsTable ct_;
    ct_.uid = 0;
    ct_.name.clear();
    ct_.date01.clear();
    ct_.number01.clear();
    ct_.date02.clear();
    ct_.date03.clear();
    ct_.number03.clear();
    ct_.date04.clear();
    ct_.number04.clear();
    ct_.date05.clear();
    ct_.date06.clear();
    ct_.number06.clear();
    ct_.date07.clear();
    ct_.number07.clear();
    ct_.number08.clear();
    ct_.tz09.clear();
    ct_.date10.clear();
    ct_.number10.clear();
    ct_.riperf10.clear();
    ct_.rireason10.clear();
    ct_.date11.clear();
    ct_.number12.clear();
    ct_.date13.clear();
    ct_.number13.clear();
    ct_.whommade13.clear();
    ct_.whomshow13.clear();
    ct_.product14.clear();
    ct_.detail15.clear();
    ct_.date16.clear();
    ct_.date17.clear();
    ct_.defectreason18.clear();
    ct_.date19.clear();
    ct_.riperf20.clear();
    ct_.rireason20.clear();
    ct_.date20.clear();
    ct_.number20.clear();
    ct_.date201.clear();
    ct_.number201.clear();
    ct_.actreason21.clear();
    ct_.date22.clear();
    ct_.number22.clear();
    return ct_;
}

QString DBClaims::dataToStringInsert(columnsTable ct_)
{
    QString stringTempData;
    stringTempData += tr("(%1").arg(ct_.uid);
    stringTempData += tr(", '%1'").arg(ct_.name);
    stringTempData += tr(", '%1'").arg(ct_.date01);
    stringTempData += tr(", '%1'").arg(ct_.number01);
    stringTempData += tr(", '%1'").arg(ct_.date02);
    stringTempData += tr(", '%1'").arg(ct_.date03);
    stringTempData += tr(", '%1'").arg(ct_.number03);
    stringTempData += tr(", '%1'").arg(ct_.date04);
    stringTempData += tr(", '%1'").arg(ct_.number04);

```

```

stringTempData += tr(", '%1'").arg(ct_.date05);
stringTempData += tr(", '%1'").arg(ct_.date06);
stringTempData += tr(", '%1'").arg(ct_.number06);
stringTempData += tr(", '%1'").arg(ct_.date07);
stringTempData += tr(", '%1'").arg(ct_.number07);
stringTempData += tr(", '%1'").arg(ct_.number08);
stringTempData += tr(", '%1'").arg(ct_.tz09);
stringTempData += tr(", '%1'").arg(ct_.date10);
stringTempData += tr(", '%1'").arg(ct_.number10);
stringTempData += tr(", '%1'").arg(ct_.riperf10);
stringTempData += tr(", '%1'").arg(ct_.rireason10);
stringTempData += tr(", '%1'").arg(ct_.date11);
stringTempData += tr(", '%1'").arg(ct_.number12);
stringTempData += tr(", '%1'").arg(ct_.date13);
stringTempData += tr(", '%1'").arg(ct_.number13);
stringTempData += tr(", '%1'").arg(ct_.whommade13);
stringTempData += tr(", '%1'").arg(ct_.whomshow13);
stringTempData += tr(", '%1'").arg(ct_.product14);
stringTempData += tr(", '%1'").arg(ct_.detail15);
stringTempData += tr(", '%1'").arg(ct_.date16);
stringTempData += tr(", '%1'").arg(ct_.date17);
stringTempData += tr(", '%1'").arg(ct_.defectreason18);
stringTempData += tr(", '%1'").arg(ct_.date19);
stringTempData += tr(", '%1'").arg(ct_.riperf20);
stringTempData += tr(", '%1'").arg(ct_.rireason20);
stringTempData += tr(", '%1'").arg(ct_.date20);
stringTempData += tr(", '%1'").arg(ct_.number20);
stringTempData += tr(", '%1'").arg(ct_.date201);
stringTempData += tr(", '%1'").arg(ct_.number201);
stringTempData += tr(", '%1'").arg(ct_.actreason21);
stringTempData += tr(", '%1'").arg(ct_.date22);
stringTempData += tr(", '%1'").arg(ct_.number22);
return stringTempData;
}

```

```

QString DBClaims::dataToStringUpdate(columnsTable ct_)
{
    QString stringTempData;
    stringTempData += tr("
%1='%2'").arg(constNamesColumnsTable[1]).arg(ct_.name);
    stringTempData += tr("
%1='%2'").arg(constNamesColumnsTable[2]).arg(ct_.date01);
    stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[3]).arg(ct_.number01);
    stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[4]).arg(ct_.date02);
    stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[5]).arg(ct_.date03);
    stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[6]).arg(ct_.number03);
    stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[7]).arg(ct_.date04);
    stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[8]).arg(ct_.number04);
    stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[9]).arg(ct_.date05);
    stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[10]).arg(ct_.date06);
    stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[11]).arg(ct_.number06);
}

```

```

        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[12]).arg(ct_.date07);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[13]).arg(ct_.number07);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[14]).arg(ct_.number08);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[15]).arg(ct_.tz09);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[16]).arg(ct_.date10);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[17]).arg(ct_.number10);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[18]).arg(ct_.riperf10);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[19]).arg(ct_.rireason10);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[20]).arg(ct_.date11);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[21]).arg(ct_.number12);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[22]).arg(ct_.date13);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[23]).arg(ct_.number13);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[24]).arg(ct_.whommade13);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[25]).arg(ct_.whomshow13);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[26]).arg(ct_.product14);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[27]).arg(ct_.detail15);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[28]).arg(ct_.date16);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[29]).arg(ct_.date17);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[30]).arg(ct_.defectreason18);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[31]).arg(ct_.date19);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[32]).arg(ct_.riperf20);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[33]).arg(ct_.rireason20);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[34]).arg(ct_.date20);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[35]).arg(ct_.number20);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[36]).arg(ct_.date201);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[37]).arg(ct_.number201);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[38]).arg(ct_.actreason21);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[39]).arg(ct_.date22);
        stringTempData += tr(",
%1='%2'").arg(constNamesColumnsTable[40]).arg(ct_.number22);
        return stringTempData;
    }

```

```

columnsTable DBClaims::stringlistToStruct(QStringList sl)
{
    columnsTable ct;
    ct = slotClearStructTables();
    if(sl.size() == constCountColumnsTable)
    {
        int i = 0;
        ct.uid                = sl.at(i++).toInt();
        ct.name               = sl.at(i++);
        ct.date01             = sl.at(i++);
        ct.number01           = sl.at(i++);
        ct.date02             = sl.at(i++);
        ct.date03             = sl.at(i++);
        ct.number03           = sl.at(i++);
        ct.date04             = sl.at(i++);
        ct.number04           = sl.at(i++);
        ct.date05             = sl.at(i++);
        ct.date06             = sl.at(i++);
        ct.number06           = sl.at(i++);
        ct.date07             = sl.at(i++);
        ct.number07           = sl.at(i++);
        ct.number08           = sl.at(i++);
        ct.tz09               = sl.at(i++);
        ct.date10             = sl.at(i++);
        ct.number10           = sl.at(i++);
        ct.riperf10           = sl.at(i++);
        ct.rireason10         = sl.at(i++);
        ct.date11             = sl.at(i++);
        ct.number12           = sl.at(i++);
        ct.date13             = sl.at(i++);
        ct.number13           = sl.at(i++);
        ct.whommade13         = sl.at(i++);
        ct.whomshow13         = sl.at(i++);
        ct.product14          = sl.at(i++);
        ct.detail15           = sl.at(i++);
        ct.date16             = sl.at(i++);
        ct.date17             = sl.at(i++);
        ct.defectreason18     = sl.at(i++);
        ct.date19             = sl.at(i++);
        ct.riperf20           = sl.at(i++);
        ct.rireason20         = sl.at(i++);
        ct.date20             = sl.at(i++);
        ct.number20           = sl.at(i++);
        ct.date201            = sl.at(i++);
        ct.number201          = sl.at(i++);
        ct.actreason21        = sl.at(i++);
        ct.date22             = sl.at(i++);
        ct.number22           = sl.at(i++);
    }
    return ct;
}

columnsTableLogPass DBClaims::stringlistToStructLogPass(QStringList sl)
{
    columnsTableLogPass ct;
    ct.uid = 0;
    ct.login.clear();
    ct.password.clear();
    if(sl.size() == constCountColumnsTableLogPass)
    {
        int i = 0;
        ct.uid                = sl.at(i++).toInt();
    }
}

```

```

        ct.login          = sl.at(i++);
        ct.password       = sl.at(i++);
    }
    return ct;
}

columnsTableMess DBClaims::stringlistToStructMess(QStringList sl)
{
    columnsTableMess ct;
    ct.uid = 0;
    ct.dateTime.clear();
    ct.login.clear();
    ct.uidRecord = 0;
    ct.operation.clear();
    if(sl.size() == constCountColumnsTableMess)
    {
        int i = 0;
        ct.uid          = sl.at(i++).toInt();
        ct.dateTime     = sl.at(i++);
        ct.login        = sl.at(i++);
        ct.uidRecord    = sl.at(i++).toInt();
        ct.operation    = sl.at(i++);
    }
    return ct;
}

QStringList DBClaims::structToStringlist(columnsTable ct)
{
    QStringList sl;
    sl.clear();
    sl.append(tr("%1").arg(ct.uid));
    sl.append(tr("%1").arg(ct.name));
    sl.append(tr("%1").arg(ct.date01));
    sl.append(tr("%1").arg(ct.number01));
    sl.append(tr("%1").arg(ct.date02));
    sl.append(tr("%1").arg(ct.date03));
    sl.append(tr("%1").arg(ct.number03));
    sl.append(tr("%1").arg(ct.date04));
    sl.append(tr("%1").arg(ct.number04));
    sl.append(tr("%1").arg(ct.date05));
    sl.append(tr("%1").arg(ct.date06));
    sl.append(tr("%1").arg(ct.number06));
    sl.append(tr("%1").arg(ct.date07));
    sl.append(tr("%1").arg(ct.number07));
    sl.append(tr("%1").arg(ct.number08));
    sl.append(tr("%1").arg(ct.tz09));
    sl.append(tr("%1").arg(ct.date10));
    sl.append(tr("%1").arg(ct.number10));
    sl.append(tr("%1").arg(ct.riperf10));
    sl.append(tr("%1").arg(ct.rireason10));
    sl.append(tr("%1").arg(ct.date11));
    sl.append(tr("%1").arg(ct.number12));
    sl.append(tr("%1").arg(ct.date13));
    sl.append(tr("%1").arg(ct.number13));
    sl.append(tr("%1").arg(ct.whommade13));
    sl.append(tr("%1").arg(ct.whomshow13));
    sl.append(tr("%1").arg(ct.product14));
    sl.append(tr("%1").arg(ct.detail15));
    sl.append(tr("%1").arg(ct.date16));
    sl.append(tr("%1").arg(ct.date17));
    sl.append(tr("%1").arg(ct.defectreason18));
}

```

```

        sl.append(tr("%1").arg(ct.date19));
        sl.append(tr("%1").arg(ct.riperf20));
        sl.append(tr("%1").arg(ct.rireason20));
        sl.append(tr("%1").arg(ct.date20));
        sl.append(tr("%1").arg(ct.number20));
        sl.append(tr("%1").arg(ct.date201));
        sl.append(tr("%1").arg(ct.number201));
        sl.append(tr("%1").arg(ct.actreason21));
        sl.append(tr("%1").arg(ct.date22));
        sl.append(tr("%1").arg(ct.number22));
        return sl;
    }

QStringList DBClaims::structToStringlistLogPass(columnsTableLogPass ct)
{
    QStringList sl;
    sl.clear();
    sl.append(tr("%1").arg(ct.uid));
    sl.append(tr("%1").arg(ct.login));
    sl.append(tr("%1").arg(ct.password));
    return sl;
}

QStringList DBClaims::structToStringlistMess(columnsTableMess ct)
{
    QStringList sl;
    sl.clear();
    sl.append(tr("%1").arg(ct.uid));
    sl.append(tr("'%1'").arg(ct.dateTime));
    sl.append(tr("'%1'").arg(ct.login));
    sl.append(tr("%1").arg(ct.uidRecord));
    sl.append(tr("'%1'").arg(ct.operation));
    return sl;
}

```

Файл «main.cpp»

```

#include <QtWidgets>
#include "SystemTray.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    SystemTray st(2323);
    if(st.prG)
        return 0;
    QApplication::setQuitOnLastWindowClosed(false);
    return app.exec();
}

```

Файл «systemtray.cpp»

```

#include "systemtray.h"
#include <qdebug.h>
#include <QListWidget>

SystemTray::SystemTray(int nPort, QWidget* pwgt)
    : QWidget(pwgt)
    , m_bIconSwitcher(false) , m_nNextBlockSize(0)

```

```

{
    createMain(); //создание гуи
    initServer(nPort); //создание сокета
    QAction* pactShowHide = //???
        new QAction("&Show/Hide Application Window", this);
    connect(pactShowHide, SIGNAL(triggered()),
        this, SLOT(slotShowHide())
        ) ;
    QAction* pactShowMessage = new QAction("&Show Message", this);
    connect(pactShowMessage, SIGNAL(triggered()),
        this, SLOT(slotShowMessage())
        ) ;
    QAction* pactChangeicon = new QAction("&Change Icon", this);
    connect(pactChangeicon, SIGNAL(triggered() ),
        this, SLOT(slotChangeicon())
        ) ;

    QAction* pactQuit = new QAction("&Quit", this);
    connect(pactQuit, SIGNAL(triggered() ), qApp ,SLOT(quit()));
    m_ptrayIconMenu = new QMenu(this);
    m_ptrayIconMenu->addAction(pactShowHide);
    m_ptrayIconMenu->addAction(pactShowMessage);
    m_ptrayIconMenu->addAction(pactChangeicon);
    m_ptrayIconMenu->addAction(pactQuit);
    m_ptrayIcon = new QSystemTrayIcon(this);
    m_ptrayIcon->setContextMenu(m_ptrayIconMenu);
    m_ptrayIcon->setToolTip("Server Tray");
    connect(m_ptrayIcon,
        SIGNAL(activated(QSystemTrayIcon::ActivationReason)), this,
            SLOT(slotDouble(QSystemTrayIcon::ActivationReason)));
    slotChangeicon();
    m_ptrayIcon->show();

    this->show();
}

void SystemTray::slotDouble(QSystemTrayIcon::ActivationReason reson){
    if(reson == QSystemTrayIcon::DoubleClick)
    {
        slotShowHide();
    }
}

void SystemTray::closeEvent(QCloseEvent *){
    if (m_ptrayIcon->isVisible())
        hide();
}

void SystemTray::slotShowHide()
{
    setVisible(!isVisible());
}

void SystemTray::slotShowMessage()
{
    m_ptrayIcon->showMessage("For your information",
        "You have selected the "
        "\"Show Message!\" option",
        QSystemTrayIcon::Information,
        3000
    ) ;
}

```

```

}

void SystemTray::slotChangeicon()
{
    m_bIconSwitcher = !m_bIconSwitcher;
    QString path = qApp->applicationDirPath() + "/pic/";
    QString strPixmapName = m_bIconSwitcher ? path + "dl.png"
                                             : path + "undo.png";
    m_ptrayIcon->setIcon(QPixmap(strPixmapName));
}

void SystemTray::createMain()
{
    setWindowTitle("Server");
    setMinimumSize(200,100);
    but1 = new QPushButton("Открыть клиентов");
    but3 = new QPushButton("Отключить сервер");
    m_ptxt = new QTextEdit();
    QGridLayout *layout = new QGridLayout();
    layout->addWidget(m_ptxt);
    m_ptxt->setReadOnly(true);
    layout->addWidget(but1);
    layout->addWidget(but3);
    connect(but1, &QPushButton::clicked, this, &SystemTray::slotOpenClient);
    connect(but3, SIGNAL(clicked()), qApp, SLOT(quit()));
    setLayout(layout);
}

void SystemTray::slotOpenClient() {
    qDebug() << "slotOpenClient";
    update();
    QWidget *w = new QWidget();
    w->setMinimumSize(200,100);
    w->setWindowTitle("Список клиентов");
    QGridLayout *layout = new QGridLayout();
    QPushButton *butClose = new QPushButton("Заккрыть");
    QListWidget *listUsers = new QListWidget();
    QList<columnsTableLogPass> listCTLP;
    columnsTableLogPass cTLP;
    listCTLP = db.selectAllRecordsInLogPass();
    listUsers->clear();
    for (int i=0; i < listCTLP.size(); i++){
        cTLP = listCTLP.at(i);
        listUsers->addItem(cTLP.login);
    }
    layout->addWidget(listUsers);
    layout->addWidget(butClose);
    w->setLayout(layout);
    w->show();
    connect(butClose, SIGNAL(clicked()), w, SLOT(close()));
}

void SystemTray::initServer(int nPort){
    m_server = new QTcpServer(this);
    m_server->close();
    prG = true;
    return;
}

connect(m_server, SIGNAL(newConnection()), this,
SLOT(slotNewConnection()));
m_ptxt->append("Server activation on number Port: " +
QString::number(m_server->serverPort()));
transferFile();

```

```

}

/*virtual*/ void SystemTray::slotNewConnection(){
    QTcpSocket* pClientSocket = m_server->nextPendingConnection();
    socG.push_back(pClientSocket);
    connect(pClientSocket, SIGNAL(disconnected()), pClientSocket,
    SLOT(deleteLater()));
    connect(pClientSocket, SIGNAL(disconnected()), this,
    SLOT(slotDeleteS()));
    connect(pClientSocket, SIGNAL(readyRead()), this,
    SLOT(slotReadClient()));
    m_ptxt->append(QString::number(pClientSocket->peerPort()));
    m_ptrayIcon->showMessage("New Client connected",
                            QString::number(pClientSocket->peerPort()),
                            QSystemTrayIcon::Information,
                            3000
                            ) ;
    sendToClient(pClientSocket, "Server Response: Connected!");
}

void SystemTray::slotDeleteS()
{
    qDebug() << "DELETE";
    vector<QTcpSocket*> socl;
    vector<clientsOnline> cl;
    foreach(QTcpSocket* s, socG)
        if(s != nullptr)
            socl.push_back(s);
    foreach(clientsOnline c, co)
        if(c.tcp != nullptr)
            cl.push_back(c);
    socG = socl;
    co = cl;
}

void SystemTray::slotReadClient()
{
    QTcpSocket* pClientSocket = (QTcpSocket*)sender();
    QDataStream in(pClientSocket);
    in.setByteOrder(QDataStream::BigEndian);

    for (;;) {
        if (!m_nNextBlockSize) {
            if (pClientSocket->bytesAvailable() < sizeof (quint64)) {
                break;
            }
            in >> m_nNextBlockSize;
        }
        if (pClientSocket->bytesAvailable() < m_nNextBlockSize)
            break;
        QString str;
        in >> str;
        qDebug() << "in.atEnd(): " << in.atEnd() << pClientSocket-
>bytesAvailable() << m_nNextBlockSize;
        QString strMessage = "Client has sent - " + str;
        m_ptxt->append(strMessage);
        qDebug() << "Read Client" << str << m_nNextBlockSize;
        if(str == "getListen")
            sendToClient(pClientSocket, str);
        else if(str == "saveLetter")
        {
            QList<columnsTable> list;

```

```

        in >> list;
        qDebug() <<"List.size: "<< list.size();
        db.addRecordInTable(list.at(0));
        sendToClient(pClientSocket, "getListen");
    }
    else if(str == "deleteLetter")
    {
        int numb;
        in >> numb;
        db.removeRecordInTable(numb);
        sendToClient(pClientSocket, "getListen");
    }
    else if(str == "updateLetter")
    {
        QList<columnsTable> list;
        in >> list;
        qDebug() <<"List.size: "<< list.size();
        db.updateRecordInTable(list.at(0));
        sendToClient(pClientSocket, "getListen");
    }
    else if(str == "fileTransfer")
    {
        QString str2;
        in >> str2;
        qDebug() << "Name files: "<< str2;
        qDebug() << "bytesAvailable nextBlockSize: " << pClientSocket->bytesAvailable() << m_nNextBlockSize;
        QByteArray buf;
        in >> buf;
        QDir dir(qApp->applicationDirPath() + "/files");
        dir.mkpath(qApp->applicationDirPath() + "/files");
        QString nameFile = dir.absolutePath() + "/" + str2;
        qDebug()<<"buf size: "<<buf.size();
        QFile file(nameFile);
        file.open(QIODevice::WriteOnly);
        file.write(buf);
    }
    else if(str == "vvLogPar")
    {
        QString login, parol;
        in >> login;
        in >> parol;
        qDebug() << "login: " << login << "parol: " << parol;
        int prN = prLogPar(login, parol);
        if(prN == 0){//vvLogPar - все ок. vvLogPar1 - нет такого логина.
vvLogPar2 - неправильный пароль
            qDebug() << "prLogPar: " << prN;
            clientsOnline c;
            c.tcp = pClientSocket;
            c.login = login;
            co.push_back(c);
            sendToClient(pClientSocket, "vvLogPar");
            try{
                foreach(QTcpSocket* s, socG)
                    sendToClient(s, "getClientOnline");
            }
            catch(exception e){
                qDebug() << e.what();
            }
        }
        else if(prN == 1)

```

```

        {
            qDebug() << "prLogPar1: " << prN;
            sendToClient(pClientSocket, "vvLogPar1");
        }
        else if(prN == 2)
        {
            qDebug() << "prLogPar2: " << prN;
            sendToClient(pClientSocket, "vvLogPar2");
        }
    }
    else if(str == "createLogPar")
    {
        QString login, parol;
        in >> login;
        in >> parol;
        qDebug() << "login: " << login << "parol: " << parol;
        int prN = prLogPar(login, parol);
        if(prN == 0)
        {
            qDebug() << "createLogPar1: " << prN;
            sendToClient(pClientSocket, "createLogPar1");
        }
        else if(prN == 1)
        {
            qDebug() << "createLogPar0: " << prN;
            columnsTableLogPass lp;
            lp.uid = -1;
            lp.login = login;
            lp.password = parol;
            if(db.addRecordInLogPass(lp))
                sendToClient(pClientSocket,
"createLogPar0");//createLogPar0 - все ок, 1 - не ок, 2 - ошибка
            else
                sendToClient(pClientSocket, "createLogPar2");
        }
    }
    else if(str == "getlogParFull")
    {
        sendToClient(pClientSocket, str);
    }
    else if(str == "getGyrnal")
    {
        sendToClient(pClientSocket, str);
    }
    else if(str == "addGyrnal")
    {
        QList<columnsTableMess> col;
        in >> col;
        qDebug() << "Gyrnal size: " << col.size();
        qDebug() << col.at(0).uid << col.at(0).login <<
col.at(0).dateTime << col.at(0).operation << col.at(0).uidRecord;
        qDebug() << "Gyrnal bool: " << db.addRecordInMess(col.at(0));
        qDebug() << "Gyrnal size: " <<
db.selectAllRecordsInMess().size();
    }
    else if(str == "addMesChat")
    {
        QString str;
        in >> str;
        chatMes.push_back(str);
        qDebug() << str;
    }

```

```

        foreach(QTcpSocket* s, socG)
            sendToClient(s, "getMesChat");
    }
    else if(str == "getMesChat")
    {
        qDebug() << str;
        sendToClient(pClientSocket, str);
    }
    else if(str == "getClientOnline")
    {
        qDebug() << str;
        sendToClient(pClientSocket, str);
    }
    else{
        sendToClient(pClientSocket,
                    "Server Response: Received \"" + str + "\""
                    ) ;

        QByteArray buf;
        while(!in.atEnd())
            in >> buf;
        buf.clear();
    }
    m_nNextBlockSize = 0;
}

void SystemTray::sendToClient(QTcpSocket* pSocket, const QString& str){
    QByteArray arrBlock;
    QDataStream out(&arrBlock, QIODevice::ReadWrite);
    out.setByteOrder(QDataStream::BigEndian);
    if(str == "getListen")
    {
        out << quint64(0) << QString::fromUtf8("getListen") << getList();
        m_ptxt->append("Server to Client - getList");
    }
    else if(str == "vvLogPar")
    {
        out << quint64(0) << QString::fromUtf8("vvLogPar");
        m_ptxt->append("Server to Client - vvLogPar");
    }
    else if(str == "vvLogPar1")
    {
        out << quint64(0) << QString::fromUtf8("vvLogPar1");
        m_ptxt->append("Server to Client - vvLogPar1");
    }
    else if(str == "vvLogPar2")
    {
        out << quint64(0) << QString::fromUtf8("vvLogPar2");
        m_ptxt->append("Server to Client - vvLogPar2");
    }
    else if(str == "getGyrnal")
    {
        out << quint64(0) << QString::fromUtf8("getGyrnal") << getGyrnal();
        m_ptxt->append("Server to Client - getList");
    }
    else if(str == "getlogParFull")
    {
        out << quint64(0) << QString::fromUtf8("getlogParFull") <<
        db.selectAllRecordsInLogPass();
        m_ptxt->append("Server to Client - getlogParFull");
    }
}

```

```

    }
    else if(str == "getMesChat")
    {
        qDebug() << str;
        out << quint64(0) << QString::fromUtf8("getMesChat") << chatMes;
        m_ptxt->append("Server to Client - getMesChat");
    }
    else if(str == "getClientOnline")
    {
        QList<QString> list;
        foreach (clientsOnline c, co)
            list.push_back(c.login);

        qDebug() << str;
        out << quint64(0) << QString::fromUtf8("getClientOnline") << list;
        m_ptxt->append("Server to Client - getClientOnline");
    }
    else {
        qDebug() << str;
        out << quint64(0) << str;
    }
    out.device()->seek(0);

    out << quint64(arrBlock.size() - sizeof(quint64));
    pSocket->write(arrBlock);
}

QDataStream &operator>>(QDataStream &s, columnsTableLogPass &c)
{
    s >> c.uid >> c.login >> c.password;
    return s;
}

QDataStream &operator<<(QDataStream &s, const columnsTableLogPass &c)
{
    s << c.uid << c.login << c.password;
    return s;
}

QDataStream &operator>>(QDataStream &s, columnsTableMess &c)
{
    s >> c.uid >> c.dateTime >> c.login >> c.uidRecord >> c.operation;
    return s;
}

QDataStream &operator<<(QDataStream &s, const columnsTableMess &c)
{
    s << c.uid << c.dateTime << c.login << c.uidRecord << c.operation;
    return s;
}

QDataStream &operator>>(QDataStream &s, tagcolumnsTable &c)
{
    s >> c.uid >> c.name >> c.date01 >> c.number01 >> c.date02 >> c.date03 >>
c.number03
    >> c.date04 >> c.number04 >> c.date05 >> c.date06 >> c.number06
>> c.date07
    >> c.number07 >> c.number08 >> c.tz09 >> c.date10 >> c.number10
>> c.riperf10 >> c.rireason10
    >> c.date11 >> c.number12 >> c.date13 >> c.number13 >>
c.whommade13
    >> c.whomshow13 >> c.product14 >> c.detail15 >> c.date16 >>
c.date17 >> c.defectreason18
    >> c.date19 >> c.riperf20 >> c.rireason20 >> c.date20 >>
c.number20 >> c.date201 >> c.number201 >> c.actreason21

```

```

        >> c.date22 >> c.number22;
    return s;
}
QDataStream &operator<<(QDataStream &s, const tagcolumnsTable &c)
{
    qDebug() << "<<";
    s << c.uid << c.name << c.date01 << c.number01 << c.date02 << c.date03 <<
c.number03
    << c.date04 << c.number04 << c.date05 << c.date06 << c.number06 <<
c.date07
    << c.number07 << c.number08 << c.tz09 << c.date10 << c.number10 <<
c.riperf10 << c.rireason10
    << c.date11 << c.number12 << c.date13 << c.number13 << c.whommade13
    << c.whomshow13 << c.product14 << c.detail15 << c.date16 << c.date17 <<
c.defectreason18
    << c.date19 << c.riperf20 << c.rireason20 << c.date20 << c.number20 <<
c.date201 << c.number201 << c.actreason21
    << c.date22 << c.number22/**/;
    return s;
}

QList<columnsTable> SystemTray::getList()
{
    qDebug() << "getList";
    listG.clear();
    listG = db.selectAllRecordsInTable();
    m_ptxt->append("GetList size - " + listG.size());
    qDebug() <<"listG: "<< listG.size();
    return listG;
}

QList<columnsTableMess> SystemTray::getGyrnal()
{
    QList<columnsTableMess> list = db.selectAllRecordsInMess();
    for(int i = 0; i < list.size(); i++)
    {
        qDebug() << list.at(i).uid << list.at(i).login << list.at(i).dateTime
<< list.at(i).operation << list.at(i).uidRecord;
    }
    m_ptxt->append("getGyrnal size - " + list.size());
    qDebug() <<"getGyrnal: "<< list.size();
    return list;
}

void SystemTray::transferFile()
{
    listG.clear();
    listG = db.selectAllRecordsInTable();
    QSet<QString> set, set2;
    for(int i = 0; i < listG.size(); i++)
        if(listG.at(i).tz09 != "")
            set.insert(getName(listG.at(i).tz09));

    foreach (const QString &value, set)
        qDebug() << value;
    QDir dir(qApp->applicationDirPath() + "/files");
    foreach (const QString &value, set)
        set2.insert(dir.absolutePath() + "/" + value);
    foreach (const QString &value, set2)
    {
        qDebug() << value;
        QFile file(value);
    }
}

```

```

        qDebug() << file.exists();
    }
}

QString SystemTray::getName(QString nameDoc)
{
    if(nameDoc.contains("/"))
    {
        int numb = 0;
        for(int i = 0; i < nameDoc.size(); i++)
            if(nameDoc.at(i) == "/")
                numb = i;
        QString nameDoc2 = "";
        for(int i = numb+1; i < nameDoc.size(); i++)
            nameDoc2 += nameDoc.at(i);
        nameDoc = nameDoc2;
    }
    return nameDoc;
}

int SystemTray::prLogPar(QString login, QString parol)
{
    int pr = 0;
    columnsTableLogPass ctl = db.selectRecordInLogPass(login);
    qDebug() << "login:" << ctl.uid << ctl.login << ctl.password;
    if(ctl.login.isNull())
        return 1;
    if(ctl.password != parol)
        return 2;
    return pr; }

```

Приложение Б

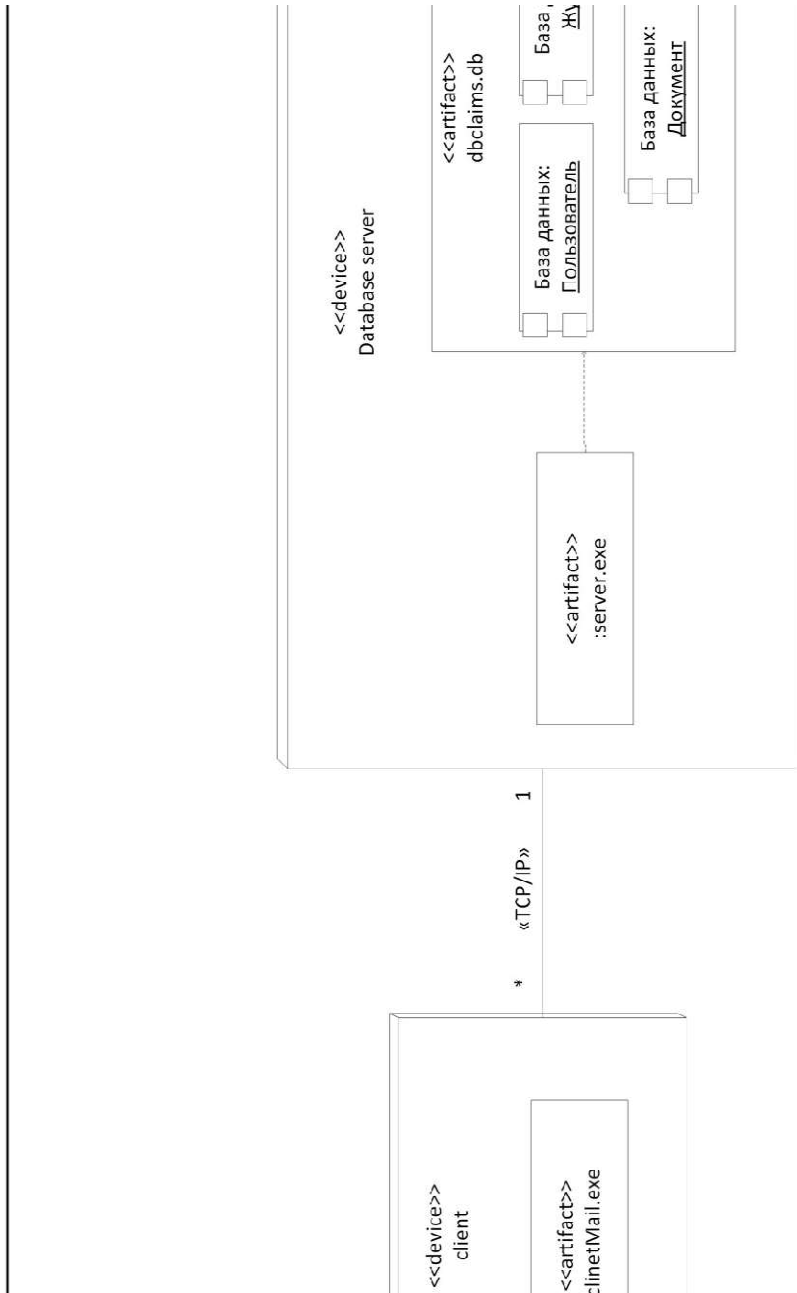
(Обязательное)

CRM-система предприятия

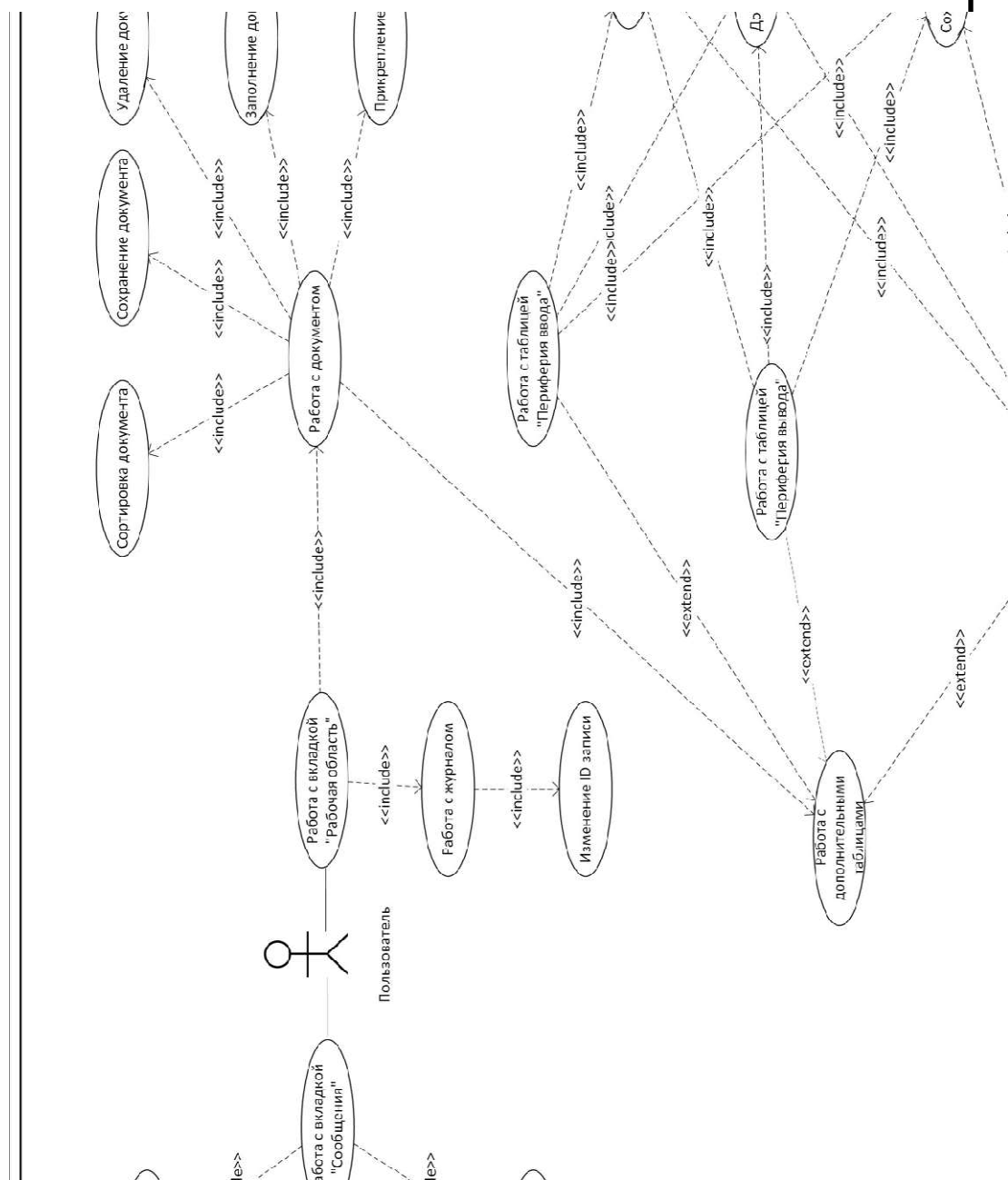
UML-диаграммы

					ПГУ 1.090301.01.001 ПЗ	Лист
						155
Изм.	Лист	№ докум. №	Подпись	Дата		

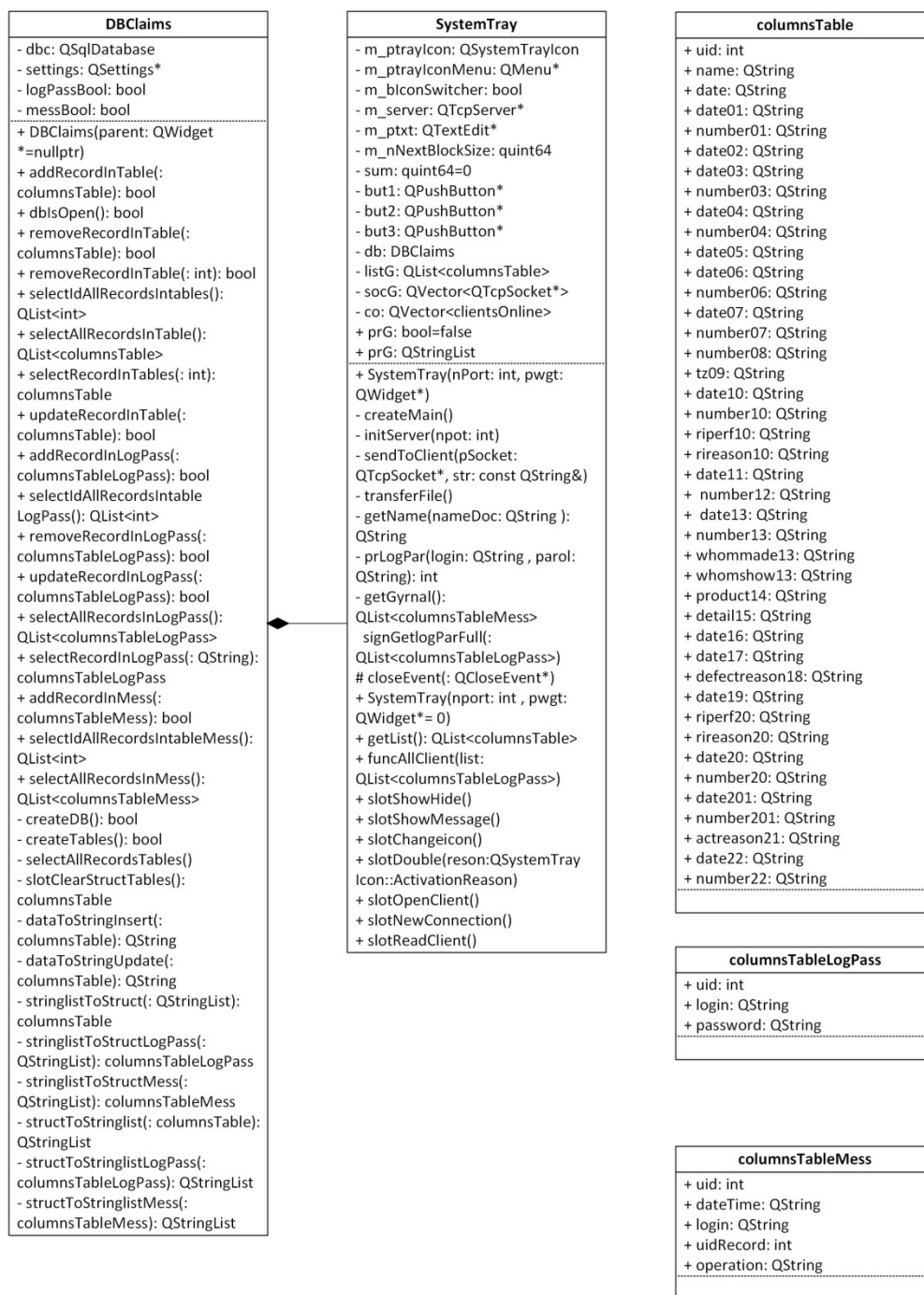
Приложение Б.1 - Диаграмма развертывания



Приложение Б.2 – Диаграмма прецедентов



Приложение Б.3 – Диаграмма классов сервера



ПГУ 1.09.03.01.02.004 Д11									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	СРМ-система предприятия Диаграмма классов			
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия				
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия				
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия				
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия				
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия				
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия				
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия				
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия				

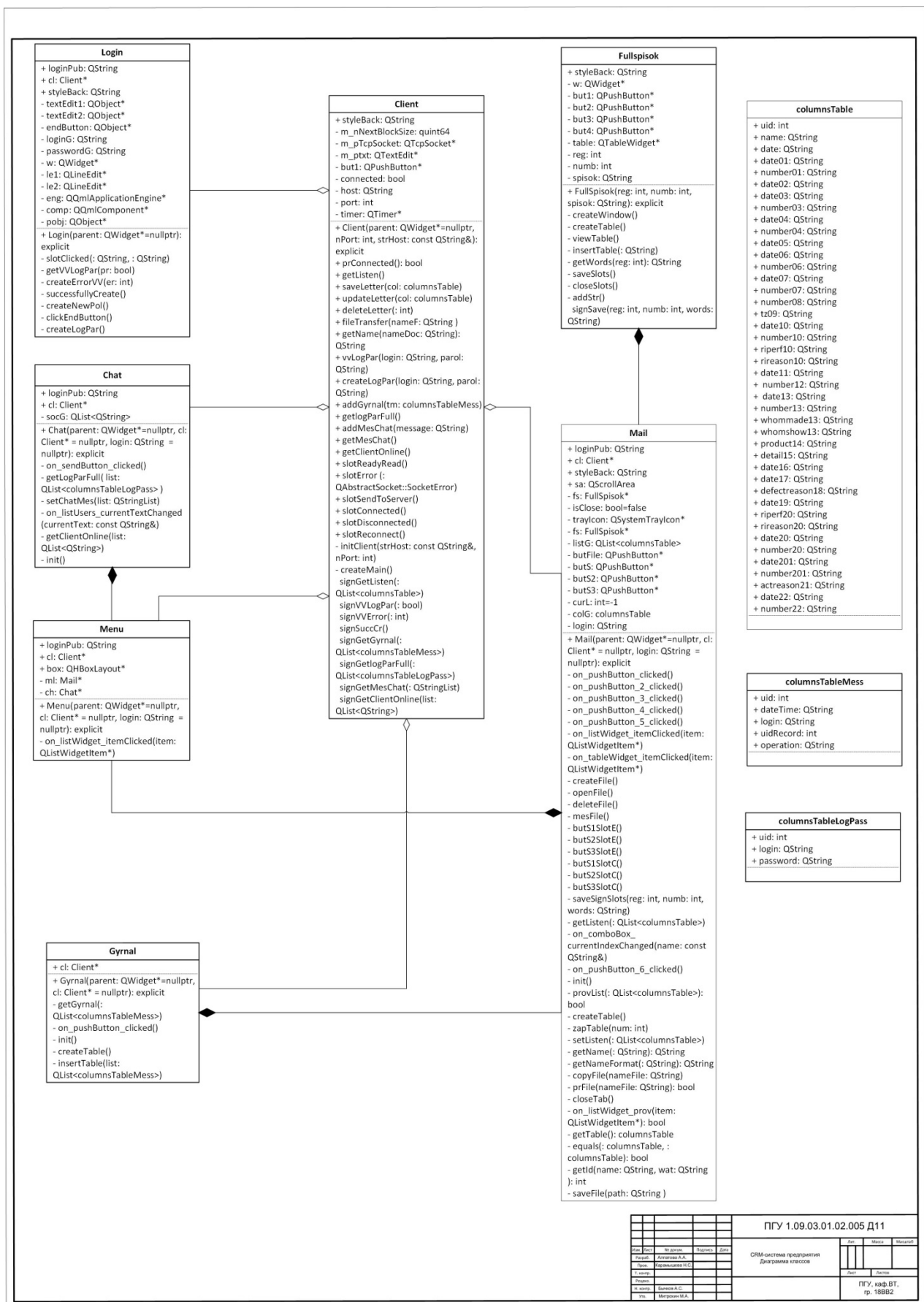
Приложение Б.4 – Диаграмма классов клиента

Изм.	Лист	№ докум. №	Подпись	Дата

ПГУ 1.090301.01.001 ПЗ

Лист

158



Приложение В

Изм.	Лист	№ докум.№	Подпись	Дата
------	------	-----------	---------	------

ПГУ 1.090301.01.001 ПЗ

Лист

159

(Обязательное)

CRM-система предприятия

Презентация

					ПГУ 1.090301.01.001 ПЗ	Лист
						160
Изм.	Лист	№ докум.№	Подпись	Дата		

CRM-система предприятия

Автор работы: ст. гр. 18BB2
Алпатова Алёна Андреевна

Научный
руководитель: к.т.н., доцент каф. ВТ ПГУ
Карамышева Надежда Сергеевна

Актуальность

CRM (англ. Customer Relationship Management) - управление взаимоотношениями с клиентами.

CRM – это программное обеспечение прикладного характера с множеством функций, предназначенное для автоматизации взаимоотношений между клиентами и предприятием.

Роль CRM-систем на предприятиях:

- автоматизация работы персонала и предприятия;
- избавление сотрудников от рутинных задач;
- быстрое формирование отчетов;
- соединение в единую систему множества отделов компании.

Изм.	Лист	№ докум. №	Подпись	Дата

ПГУ 1.090301.01.001 ПЗ

Лист

161

Обзор аналогов существующих CRM-систем

Название системы	Достоинства	Недостатки
Bitrix24	Комплексность. Возможность создания бесконечного числа групп. Упрощенный документооборот.	Низкая скорость работы.
A2B	Простой и разнообразный функционал. Быстрое внедрение. Широкие возможности для доработок и администрирования.	Малое количество функций в почтовом разделе. Не подходит для использования в сложноустроенных компаниях.
CRM Простой Бизнес	Легкость в управлении заказами, формировании отчетов Наличие телефонии. Грамотная система оповещения. Удобство при проведении конференций.	Отсутствие возможности интеграции с другими сервисами. Требуется обучение пользователя.
amoCRM	Простой интерфейс, легкость в освоении. Наличие модуля для связи сотрудников. Присутствует возможность интеграции и внесения доработок.	Избыточность. Рассчитана только на работы с клиентами и продажи.
РосБизнесСофт	Гибкая настройка прав доступа. Наличие нескольких уровней настройки.	Сложность в использовании. Необходимость подключать отдельные опции. Долгое внедрение.

Постановка задачи

Цель работы:

разработка CRM-системы предприятия, предназначенной для повышения эффективности, оперативности, устойчивости работы предприятия и управленческих решений.

Основные задачи:

- анализ аналогов, шаблонов CRM-систем;
- изучение архитектуры клиент-серверных приложений;
- изучение сетевых моделей передачи данных (TCP/IP);
- овладение навыками создания архитектуры хранения данных;
- изучение основы разработки удобного и функционального интерфейса;
- проектирование и разработка многомодульного клиент-серверного приложения для оптимизации производства.

Изм.	Лист	№ докум.№	Подпись	Дата

ПГУ 1.090301.01.001 ПЗ

Лист

162

Диаграмма развертывания

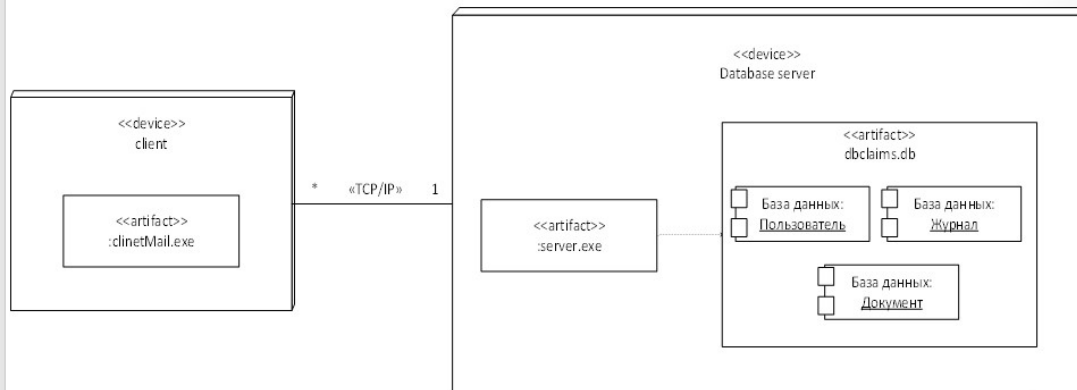
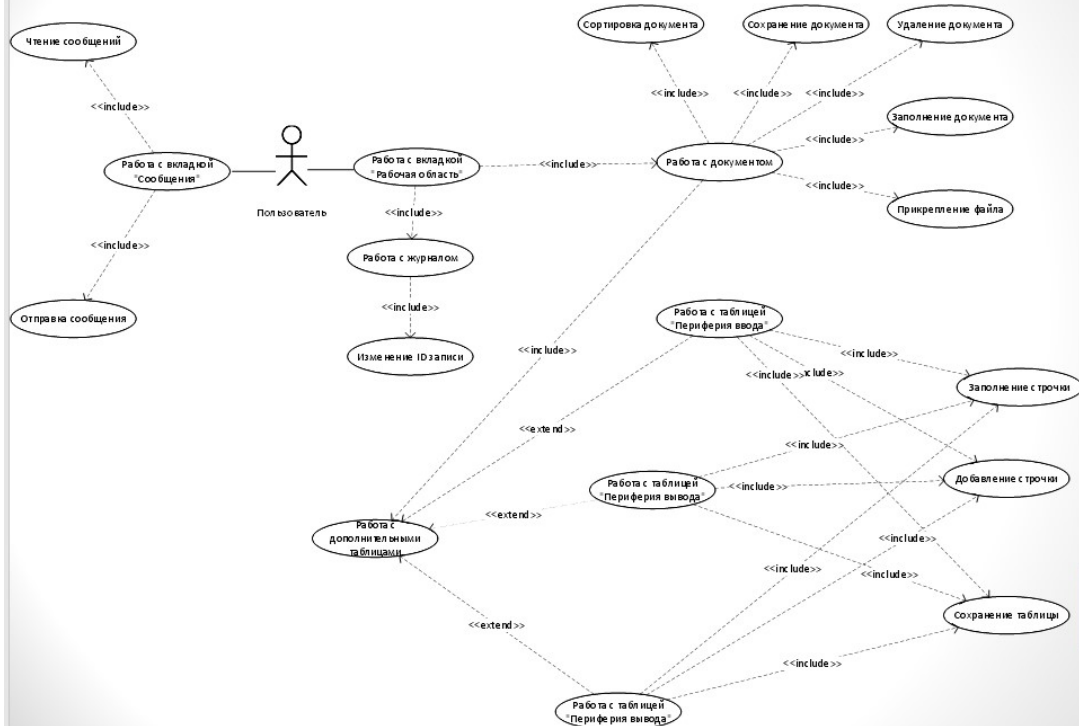


Диаграмма прецедентов



Диаграммы деятельности

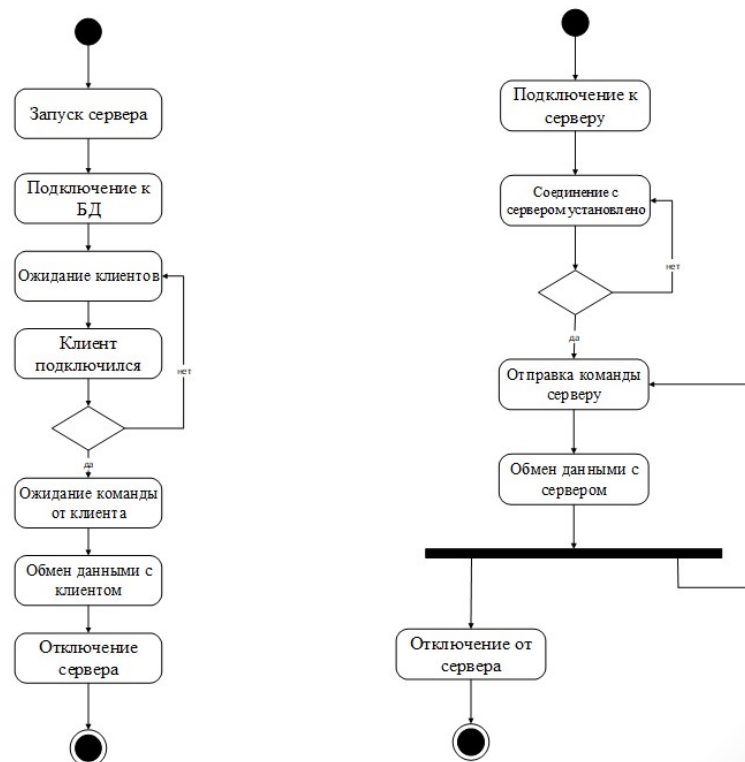


Диаграмма классов клиента

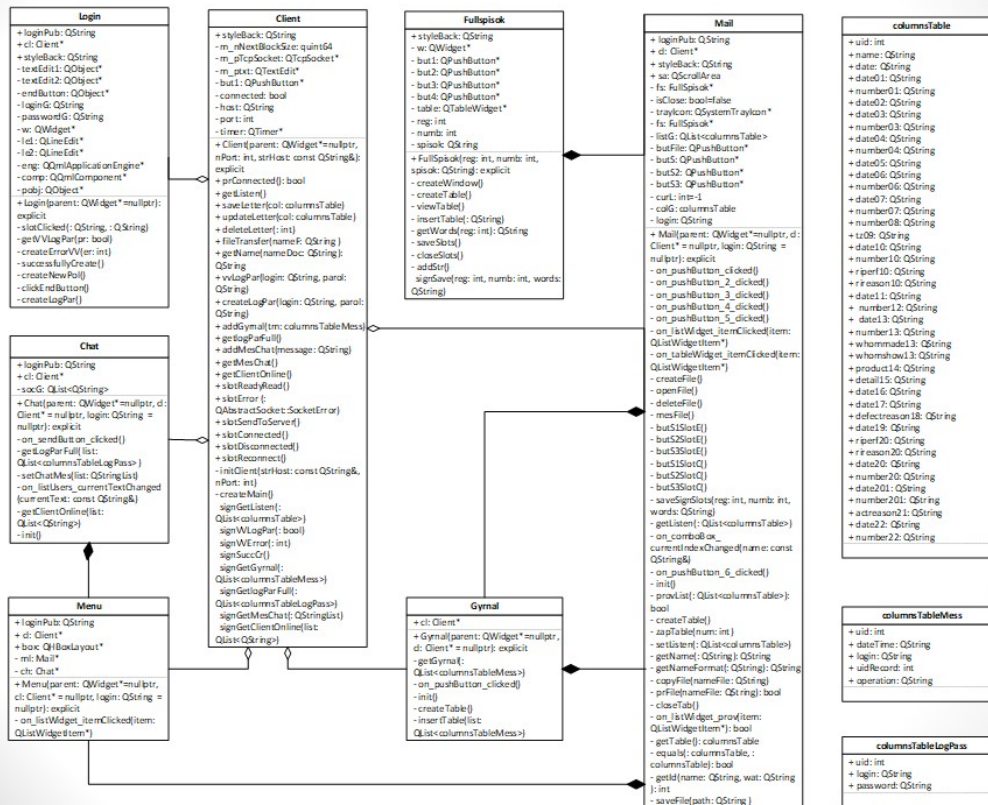
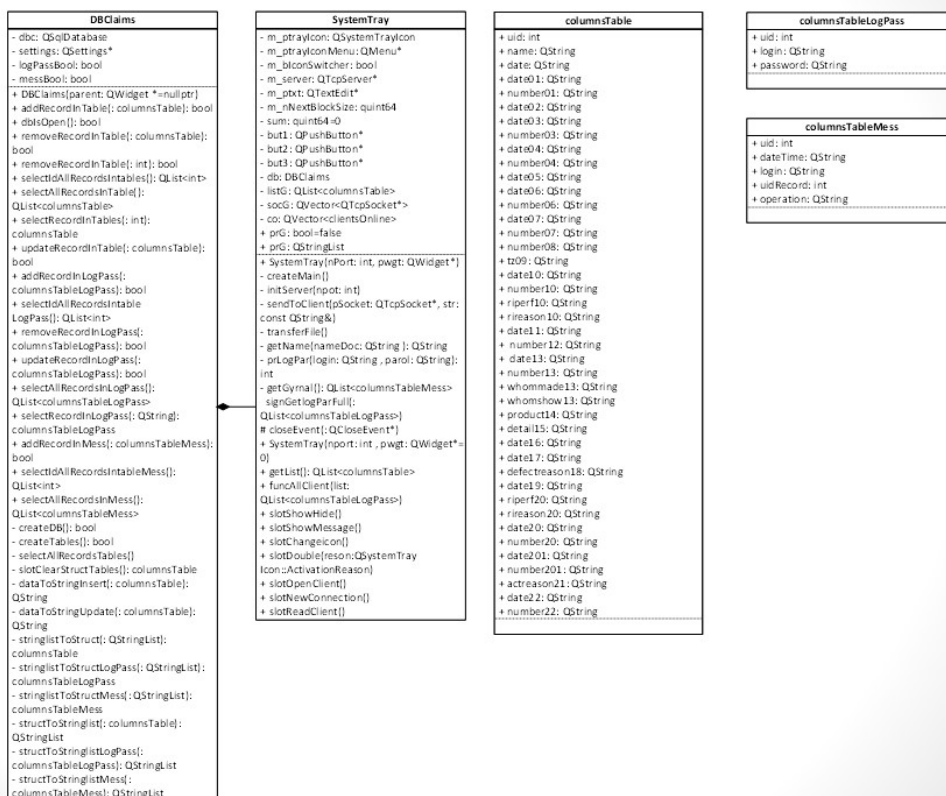


Диаграмма классов сервера



Окно авторизации

Логин:

Пароль:

Рабочее пространство

Alpatova A.A.

Рабочее пространство Сообщения

Документы

Название документа
Отчет Сидорова Н.Г.

Сортировка: Дата изменения Создать Удалить Сохранить Журнал

Наскладная за январь	Дата и номер составления заявки от кадровой службы	Дата	01-07-2021
Отчет Сидорова Н.Г.		Номер	1365
Конец апреля 2021	Дата получения заявки от кадровой службы	Дата	12-07-2021
Результат проверки НТЦ	Номер резюме и дата его составления	Дата	15-07-2021
Прибытие оборудования за 2020		Номер	445
Получение ПО июль 2021	Номер заключения трудового договора и дата его заключения	Номер	678
Запланированные мероприятия на 2022		Дата	23-08-2021
Наскладная за март 2022	Дата первого выхода сотрудника	Дата	24-08-2021
Получение оборудования за 2020	Дата и номер заявки по выдаче технических средств	Дата	01-09-2021
Отчет за май 2022		Номер	32
	Ответ сотрудника о получении технических средств	Номер	121
		Дата	01-09-2021

Рабочее пространство (продолжение)

Alpatova A.A.

Рабочее пространство Сообщения

Документы

Название документа
Отчет Сидорова Н.Г.

Сортировка: Дата изменения Создать Удалить Сохранить Журнал

Наскладная за январь	Полученные сотрудником периферия ввода и вывода	Кому предъявлен	Артемиеву Д.В.
Отчет Сидорова Н.Г.		Периферия ввода	Периферия ввода
Конец апреля 2021		Периферия вывода	Периферия вывода
Результат проверки НТЦ	Дата приемки сотрудником программного обеспечения	Дата	01-07-2020
Прибытие оборудования за 2020	Дата обнаружения сбоя	Дата	14-10-2022
Получение ПО июль 2021	Причина сбоя по претензионному акту	Причина сбоя	выход из строя ламп
Запланированные мероприятия на 2022	Дата получения претензионного акта	Дата	21-11-2020
Наскладная за март 2022		Обнаруженный дефект	кабель на лампах поврежден
Получение оборудования за 2020		ФИО сотрудника	Лесин И.С.
Отчет за май 2022	Результаты исследования, номер и дата акта у изготовителя. Номер и дата программы исследования	Номер	5437
		Дата	01-12-2020

Журнал

	Дата и время	Логин	ID документа	Операция
1	14:57 27-04-2022	Alpatova A.A.	0	Создание
2	15:31 27-04-2022	Alpatova A.A.	1	Изменил
3	15:32 27-04-2022	Alpatova A.A.	1	Создание
4	15:32 27-04-2022	Alpatova A.A.	2	Создание
5	15:33 27-04-2022	Alpatova A.A.	1	Изменил
6	15:33 27-04-2022	Alpatova A.A.	3	Создание
7	15:34 27-04-2022	Alpatova A.A.	4	Создание
8	15:34 27-04-2022	Alpatova A.A.	5	Создание
9	15:34 27-04-2022	Alpatova A.A.	6	Создание
10	15:35 27-04-2022	Alpatova A.A.	7	Создание
11	15:37 27-04-2022	Alpatova A.A.	2	Изменил
12	15:38 27-04-2022	Ivanov G.M.	5	Изменил
13	15:39 27-04-2022	Ivanov G.M.	8	Создание
14	15:39 27-04-2022	Ivanov G.M.	9	Создание
15	15:39 27-04-2022	Ivanov G.M.	10	Удалил
16	15:40 27-04-2022	Starodubova D.I.	4	Изменил
17	15:40 27-04-2022	Starodubova D.I.	7	Изменил
18	15:41 27-04-2022	Starodubova D.I.	9	Создание
19	15:44 27-04-2022	Gulyakova N.D.	10	Изменение
20	15:45 27-04-2022	Gulyakova N.D.	10	Создание

Закрыть

Сообщения

Makarov M.S.

Рабочее пространство

Сообщения

Alpatova A.A.: Здравствуйте! Кто вносил последние изменения?

Zacepin N.F.: Добрый день. Вы спрашиваете про отчет за август?

Makarov M.S.: Посмотрите, пожалуйста, по журналу!

Alpatova A.A.: Хорошо. Спасибо!

Пользователи

- Alpatova A.A.
- Ivanov G.M.
- Makarov M.S.
- Starodubova D.I.
- Astafiev V.I.
- Gulyakova N.D.
- Speranskij D.D.
- Nazarova U.A.
- Goryinin A.M.
- Zacepin N.F.

Пользователь в сети

Пользователь не в сети

Отправить

Заключение

В результате данной работы спроектирована, разработана и отлажена CRM-система предприятия.

Данное приложение:

- работает на основе клиент-серверного соединения с использованием TCP/IP;
- использует базу данных (хранилище данных);
- каждая программа характеризуется многомодульностью;
- имеет дружелюбный и интуитивно-понятный интерфейс;
- обладает возможностью функционального расширения в связи с нуждами предприятия.

					ПГУ 1.090301.01.001 ПЗ	Лист
						168
Изм.	Лист	№ докум.№	Подпись	Дата		