Частное учреждение образования

«Колледж бизнеса и права»

ПРОГРАММА ДЛЯ СИНХРОНИЗАЦИИ ПЕРЕДАЧИ ДАННЫХ "Device Sync"

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

по учебному предмету «Конструирование программ  
 и языки программирования»

КП Т.096020.401

Руководитель проекта (В.Ю.Михалевич)

Обучающийся (В.А. Лукашонок)

2023

**СОДЕРЖАНИЕ**

Изм.

Изм.

Изм.

Изм.

Лист

Лист

Лист

Лист

№ докум.

№ докум.

№ докум.

№ докум.

Подпись

Подпись

Подпись

Подпись

Дата

Дата

Дата

Дата

Лист

Лист

Лист

Лист

3

3

3

3

КП Т.096020.401 ПЗ

КП Т.716007.401 ПЗ

КП Т.716009.401 ПЗ

КП Т.716007.401 ПЗ

Разраб.

Разраб.

Разраб.

Разраб.

*Лукашонок В.А.*

*Голубев И.В.*

*Гринь А.М.*

*Голубев И.В.*

Провер.

Провер.

Провер.

Провер.

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Утверд.

Утверд.

Утверд.

Утверд.

Программа для синхронизации передачи данных "Device Sync"

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Листов

Листов

Листов

Листов

60

50

55

50

КБП

КБП

КБП

КБП

[Введение 3](#_Toc132640062)

[1 Описание задачи 5](#_Toc132640063)

[1.1 Анализ предметной области 5](#_Toc132640064)

[1.2 Постановка задачи 6](#_Toc132640065)

[2 Проектирование системы 7](#_Toc132640066)

[2.1 Требования к приложению 7](#_Toc132640067)

[2.2 Проектирование модели 7](#_Toc132640068)

[2.3 Организация данных 9](#_Toc132640069)

[2.4 Концептуальный прототип 10](#_Toc132640070)

[3 Описание реализации программного средства 13](#_Toc132640071)

[3.1 Инструменты разработки и применяемые технологии 13](#_Toc132640072)

[3.2 Порядок авторизации пользователей 14](#_Toc132640073)

[3.3 Организация данных 14](#_Toc132640074)

[3.4 Функции: логическая и физическая организация 17](#_Toc132640075)

[3.5 Входные и выходные данные 22](#_Toc132640076)

[3.6 Функциональное тестирование 23](#_Toc132640077)

[3.7 Описание справочной системы 26](#_Toc132640078)

[4 Применение 27](#_Toc132640079)

[4.1 Назначение программного средства 27](#_Toc132640080)

[4.2 Условия применения 27](#_Toc132640081)

[Заключение 29](#_Toc132640082)

[Список использованных источников 30](#_Toc132640083)

[Приложение А Текст программных модулей 31](#_Toc132640084)

[Приложение Б Формы выходных документов 32](#_Toc132640085)

[Приложение В Результаты работы программы 33](#_Toc132640086)

**Введение**

Фото, видео, сообщения – в современном мире не проходит и дня без взаимодействия с теми или иными данными, и вопрос об их передачи обстоит особенно остро. Возможность быстро передать необходимы файл или сообщения на устройство является незаменимой, именно для таких целей создаются экосистемы позволяющие “бесшовно” перемещается между устройствами.

Существует множество программных средств, решающих данную проблему в той или иной мере начиная от мессенджеров заканчивая как было оговорено экосистемами. Спрос на быстрый и удобный способ обмена данными постоянный и большой, что доказывает существования бесчисленного количество мессенджеров и альтернативных им способов обмена данными.

Целью курсового проекта на тему «Программа для синхронизации передачи данных "Device Sync"» является разработка программы, с помощью которой пользователи могли бы передавать текстовыми сообщениями и файлы. Для того, чтобы пользователь мог передавать данные, ему необходимо настроить соединение между устройствами. Для этого необходимо указать внешний Ip адрес получателя порт по которому он сможет принимать данные, а также собственный порт для получения данных. После успешного подключения, пользователям будут доступны функции отправки текстовых сообщений и файлов.

Решение поставленных задач отражено в пояснительной записке, которая состоит из четырех разделов и содержит необходимую и достаточную информацию по использованию данного программного средства.

В первом разделе «Описание задачи» рассматривается сущность и актуальность поставленной задачи, описание существующих аналогов, описание основных функций для разработки программного средства.

Второй раздел «Проектирование системы» содержит проектирование модели на основе выбранных диаграмм, требования к аппаратным и операционным ресурсам, средства защиты, организация данных и описание концептуального прототипа.

Третий раздел «Описание реализации программного средства» включает описание избранной технологии для реализации программы и избранные инструменты разработки, физическая организация данных, способы реализации функции пользователя, формат входных и выходных данных, функциональное тестирование, описание справочной системы.

Четвертый раздел «Применение» содержит информацию, необходимую в процессе эксплуатации программного средства: его назначение и условия применения.

В заключении описывается выполнение поставленной задачи, степень соответствия проектных решений задания, причины несоответствия, если таковые имеются.

В приложении А представлен текст программных модулей, в приложении Б – формы выходных документов, в приложении В – результаты работы программы. (порядок согласно упоминанию по ПЗ)

Графическая часть представлена диаграммами вариантов использования, классов, последовательности, деятельности (выбрать нужную).

**1 Описание задачи**

**1.1 Анализ предметной области**

Требуется разработать программное средство для переписки пользователей – мессенджер «RPYNOD».

Предметной областью решаемой задачи является мессенджер.

Мессенджер – это программное средство, предназначенное для обмена сообщениями двумя пользователями.

Чаще всего под мессенджером понимают программу, в которую пишут сообщения и где их можно прочесть. Однако, за каждой такой программой стоит сеть обмена сообщениями, которая тоже входит в понятие «мессенджер». Это может быть сеть внутри компании, а может быть глобальная сеть, например Jabber.

К основным функциям мессенджеров относятся:

* обмен текстовыми сообщениями – самый простой вариант, наподобие СМС;
* возможность прикрепить к сообщению смайлик, фото, картинку, документ, видео или другой необходимый файл – некий аналог электронной почты, но только работающей в режиме онлайн;
* создание группы пользователей для общения по интересам, по работе, по учебе – коллективное общение и даже дистанционное обучение;
* в некоторых мессенджерах можно записать голосовое сообщение и отправить его, либо есть набор голосом – удобно для тех, кто не любит набирать текст вручную;
* встречается иногда возможность сделать звонок из мессенджера на мобильный телефон другу или даже на городской телефон – обычно за дополнительную плату.
* разработчики постоянно работают над улучшением мессенджеров, оснащают их новыми функциями, например, наличием облачного сервера или возможностью проводить видеоконференции.

Мессенджеры стараются предоставить возможности архивации чатов, чтобы можно было в случае чего восстановить переписку и переговоры. Такое может понадобиться, скажем, при смене номера телефона абонента.

Программной средство будет предназначено для людей, которые желают вести переписку с помощью интернета.

Мессенджер будет состоять из двух частей: клиент и сервер. Пользователи будут видеть только клиентское приложение, в котором будут реализованы все основные функции.

Работа сервера будет заключаться в обработке запросов от клиентов. Сервер будет иметь базу данных, в которой будут храниться таблицы пользователей, сообщений, запросов, друзей.

Для оптимизации работы сервера, каждое клиентское программное средство будет иметь локальную базу данных, чтобы хранить там некоторые данные и не запрашивать их у сервера в очередной раз. Например, этими данными могут быть сообщения между пользователями [8].

**1.2 Постановка задачи**

Исходя из анализа предметной области можно выделить следующие задачи:

* ведение базы данных на сервере, в которой будет храниться вся информация;
* ведение локальной базы данных на клиенте, для сокращения количества обращений к серверу;
* авторизация пользователей;
* регистрация пользователей;
* восстановления пароля с помощью email;
* изменение пароля;
* переписка пользователей текстовыми сообщениями, с использованием текстовых сообщений и стикеров;
* отправление запросов в друзья другим пользователям;
* поиск других пользователей;
* отклонение, либо принятие запросов в друзья.

Существует довольно много аналогов для данного приложения, такие как Телеграмм, Вайбер, ВКонтакте и другие. Разрабатываемый мессенджер направлен только на пользователей Windows 10. Программное средство сильно упрощено и не имеет больших массивов данных.

**2 Проектирование системы**

**2.1 Требования к приложению**

Разрабатываемое программное средство должно иметь понятный и удобный в использовании интерфейс, чтобы взаимодействие между программой и пользователем было максимально упрощено.

Согласно общим требованиям, графический интерфейс разрабатываемого программного средства должен:

* ориентироваться на пользователя, который общается с программой на внешнем уровне взаимодействия;
* сохранять стандартизированное назначение и местоположение на экране графических объектов, работающих в среде Windows.

Интерфейс программного средства будет разрабатываться с учетом общих требований к пользовательскому интерфейсу. Пользователь может изменять настройки по своему усмотрению при работе с программным средством.

Существуют общие требования, которые предъявляются к программным приложениям:

* соответствие стандартам организации интерфейса: использование многооконного подхода, реализация управления работой программного приложения с помощью элементов управления;
* требования к выбранной цветовой схеме рабочего экрана;
* выполнение одной функции с помощью разных элементов управления.

Для обучения пользователя необходимо разработать справочную систему, в которой должны быть раскрыты все аспекты работы с программой, возможные трудности, возникшие во время работы и пути их решения.

Для удобства работы пользователя с программным средством необходимо при разработке форм придерживаться единого стиля оформления. Формы не должны быть перегружены излишней информацией или содержать информацию, не относящуюся к данной форме. Также необходимо предусмотреть защиту данных от удаления и изменения, а также от ввода некорректных данных. В случае ввода некорректных данных или попытке совершить запрещенные действия, пользователь должен быть проинформирован о своих действиях с помощью диалоговых окон.

**2.2 Проектирование модели**

Цель моделирования данных состоит в обеспечении разработчика информационной системы концептуальной схемой базы данных в форме одной или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему без данных.

Наиболее распространенным средством моделирования данных является диаграмма «Сущность-связь» (ERD). С их помощью определяется важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных.

Диаграмма «Сущность-связь» в нотации Чена представлена на рисунке 2.1.

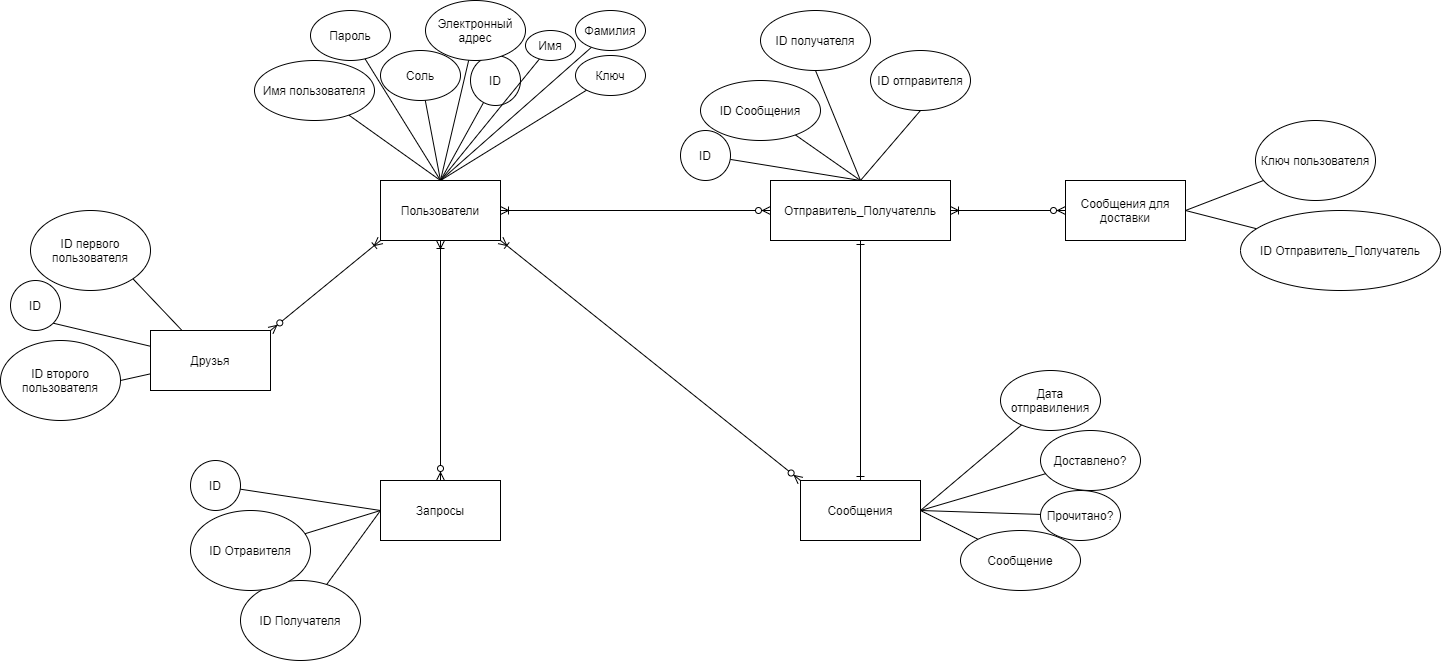


Рисунок 2.1

Исходя из предметной области можно выделить следующие сущности разработки: «Пользователи», «Друзья», «Запросы», «ОтправительПолучатель», «Сообщения для доставки», «Сообщения».

Для сущности «Пользователь» атрибутами будут являться:

* имя пользователя;
* пароль;
* соль;
* адрес электронной почты;
* имя;
* фамилия;
* ключ.

Для сущности «Друг» атрибутами будут являться:

* id первого пользователя;
* id второго пользователя.

Для сущности «Сообщение» атрибутами будут являться:

* текст сообщения;
* состояние;
* состояние доставки;
* дата отправки.

Для сущности «Запрос» атрибутами будут являться:

* id пользователя отправителя;
* id второго получателя.

Для сущности «ОтправительПолучатель» атрибутами будут являться:

* id пользователя отправителя;
* id второго получателя;
* id сообщения.

Для сущности «Сообщения для доставки» атрибутами будут являться:

* ключ;
* id записи из таблицы ОтправительПолучатель.

Суть диаграммы вариантов использования заключается в том, что проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования.

К «extend» относится информация о неудаче:

* восстановления пароля;
* регистрации;
* входа;
* добавления друга;
* отправки сообщения;
* отклонения запроса.

Также к «extend» относится подключение к базе данных и вызов справочной системы.

К «include» относится:

* авторизация;
* регистрация;
* восстановление пароля;
* поиск пользователя;
* добавление друга;
* чтение сообщений;
* отклонение запроса.

Диаграмма вариантов использования представлена на листе 1 графической части.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывать их внутреннюю структуру и типы отношений.

При реализации программного средства были реализованы следующие основные классы:

* Авторизация;
* ФормаДрузья;
* ФормаЧат;
* ФормаГлавная;
* ФормаПрофиля;
* МанипуляцияСервером.

Класс «Авторизация» отвечает за включение форм входа, регистрации, восстановления пароля, а также за их логику. Класс «ФормаДрузей» необходим для просмотра списка друзей и запросов, а также для управления (добавление, удаление, отклонение) друзьями и запросами. Класс «ФормаЧат» необходим для отображения истории сообщений с пользователем, а также для переписки. Класс «ФормаПрофиля» позволяет пользователям изменять текущий пароль, а также производить выход из аккаунта. Класс «ФормаГлавная» объединяет в себе вышеописанные классы и предоставляет возможность перехода от одной формы к другой. Класс «МанипуляцияСервером» является самым важным, так как он отвечает за общение с сервером, т.е. получение и обработку данных.

Диаграмма классов представлена на листе 2 графической части.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности.

Диаграммы деятельности – частный случай диаграмм состояний. Основная цель использования таких диаграмм – визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Диаграмма деятельности представлена на листе 3 графической части.

Временной аспект поведения имеет существенное значение при моделировании синхронных процессов, описывающих взаимодействия объектов. Именно для этой цели и используются диаграммы последовательности, в которых ключевым моментом является динамика взаимодействия объектов во времени. Например, сотрудник переходит с вкладки на вкладку и добавляет данные, что является временным аспектом поведения сотрудника. Он включает программу, вводит данные, сохраняет данные – последовательность действий. Диаграмма последовательности представлена на листе 4 графической части.

**2.3 Концептуальный прототип**

Программа для создания макетов: <https://balsamiq.com/wireframes/>

Ссылка на диске exe <https://drive.google.com/file/d/1gWYdgFSz-sJUJpz6r1oRJ69nLGEbFetU/view?usp=share_link>

Концептуальный прототип состоит из описания внешнего пользовательского интерфейса, а именно, элементов управления.

При создании данного приложения важную роль играют формы, так как они являются основным средством работы с пользователем. Разрабатываемое приложение будет содержать несколько форм.

Все формы будут содержать стандартные пользовательские элементы управления.

В рабочем режиме программы, пользователю, для удобной навигации, будет предоставлено меню.

Первая форма, которую увидит пользователь будет форма Входа, с помощью которой пользователь сможет авторизоваться в приложения. Данная форма будет иметь два текстовых поля для ввода имени пользователя и пароля, а также две кнопки «Забыли пароль?» и «Регистрация». Кнопка «Забыли пароль» будет открывать вкладку для восстановления пароля. Кнопка «Регистрация» будет открывать вкладку для регистрации нового аккаунта. Для входа в аккаунт пользователь должен будет заполнить поля верными данными и нажать кнопку «Войти». Макет формы представлен на рисунке 2.2. Макеты форм для восстановления пароля и регистрации аналогичны макету, изображенному на рисунке 2.2.

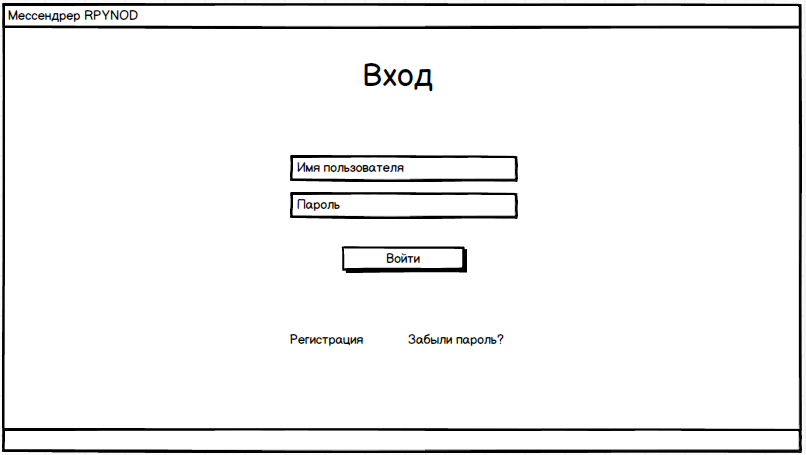


Рисунок 2.2

После успешного входа, будет открыта главная форма. Данная форма будет иметь главное меню с двумя вкладками: «Сообщения» и «Друзья». Вкладка «Сообщения» будет иметь список пользователей, с которыми начата переписка, а также сама переписка с выбранным пользователем. Форма «Друзья» будет иметь две вкладки «Мои друзья» и «Запросы». На вкладке «Мои друзья» будет список друзей, а также кнопки для удаления и перехода к переписке для выбранного пользователя. На вкладке «Запросы» будет список входящих и исходящих запросов, а также кнопки для отклонения или принятия запроса. Макет формы «Мои сообщения» предоставлен на рисунке 2.3.

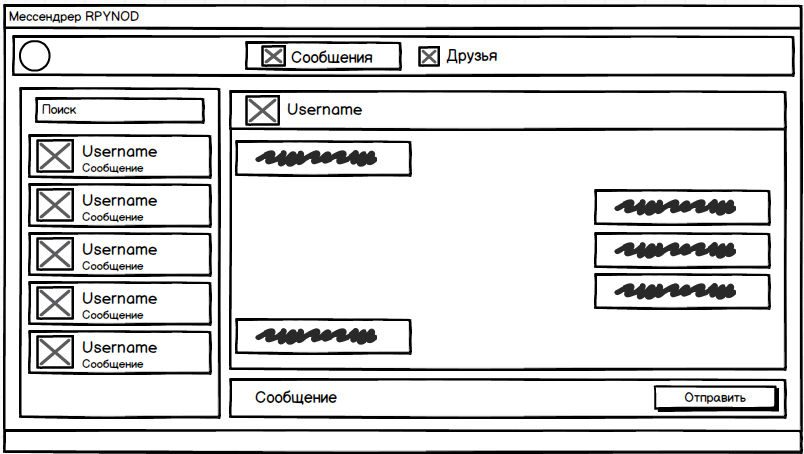


Рисунок 2.3

Макет Вкладке «Мои друзья» на форме «Друзья» представлен на рисунке 2.4. Макет вкладки «Запросы» аналогичен макету «Мои друзья».

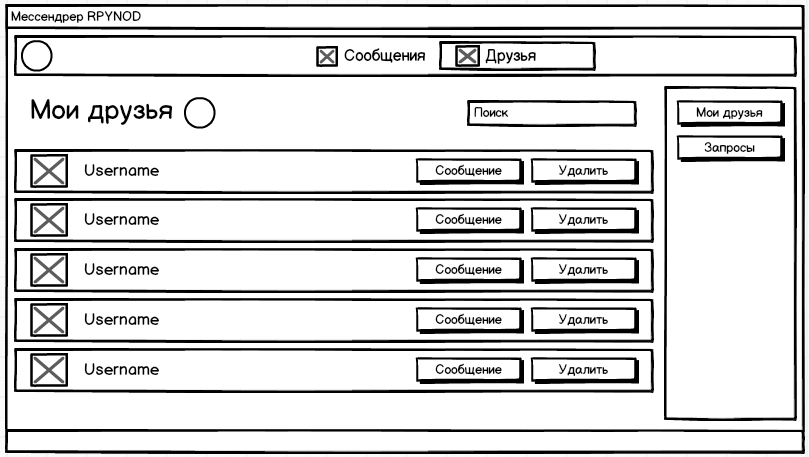


Рисунок 2.4

**3 Описание реализации программного средства**

**3.1 Инструменты разработки и применяемые технологии**

Инструментами для разработки данного программного средства будут являться:

* операционная система Windows 10;
* среда программирования Visual Studio;
* язык программирования C#.
* база данных SQLite;
* менеджер базами данных SQLiteStudio;
* система контроля версий GitHub;
* средство для разработки диаграмм Diagrams net.

Для разработки программного средства выбрана операционная система Windows 10.

Интегрированная среда разработки Visual Studio – это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки [7].

Язык программирования C# – это объектно- и **компонентно-ориентированный** язык программирования. Он предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО [2].

SQLite  – компактная встраиваемая реляционная база данных. Исходный код библиотеки передан в общественное достояние. Является чисто реляционной базой данных.

Слово «встраиваемый» означает, что SQLite не использует парадигму клиент-сервер. Т.е. движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID функции достигаются в том числе за счёт создания файла журнала [9].

SQLiteStudio – польский **менеджер баз данных SQLite.** Имеет русский интерфейс (язык выбирается при первом запуске). Распространяется бесплатно, [скачать SQLiteStudio](http://sqlitestudio.pl/?act=download) можно с официального сайта разработчиков. Важно отменить, что менеджер баз данных SQLiteStudio является кроссплатформенным, на странице закачек вы сможете выбрать нужную вам версию. Этот менеджер баз данных устанавливается путем распаковки архива в любую папку на вашем компьютере. С помощью SQLiteStudio можно манипулировать базой данных [10].

GitHub  – крупнейший [веб-сервис](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81) для [хостинга](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B3) [IT-проектов](https://ru.wikipedia.org/wiki/IT) и их совместной разработки. Веб-сервис основан на системе контроля версий [Git](https://ru.wikipedia.org/wiki/Git" \o "Git) и разработан на [Ruby on Rails](https://ru.wikipedia.org/wiki/Ruby_on_Rails) и [Erlang](https://ru.wikipedia.org/wiki/Erlang" \o "Erlang) компанией GitHub, Inc (ранее Logical Awesome). Сервис бесплатен для проектов с [открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) и (с 2019 года) небольших частных проектов, предоставляя им все возможности (включая [SSL](https://ru.wikipedia.org/wiki/SSL)), а для крупных корпоративных проектов предлагаются различные платные тарифные планы [11].

Diagrams.net – это условно бесплатная утилита, разработанная компанией Metaproducts для работы под управлением операционной системы Microsoft Windows, которая позволяет легко контролировать текущую деятельность Интернета на компьютере [6].

**3.2 Порядок авторизации пользователей**

Определить порядок аутентификации и авторизации пользователей, в том числе:

* описать необходимые роли;
* указать набор функций, которые будет выполнять каждая из ролей.

Подраздел 3.2 будет необходим, если предусмотрено разделение ролей в ПС.

**3.3 Организация данных**

Организация данных подразумевает создание модели данных, главными элементами которой являются сущности и их связи.

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и внешние ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Структура базы данных разрабатываемого программного средства включает шесть таблиц. Описание таблиц приводится в таблицах 1-5.

Таблица «Users» хранит информацию о пользователях, структура которой представлена в таблице 3.1.

Таблица 3.1 – Структура таблицы «Users»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID | integer | 4 | Идентификатор пользователя |
| Login | varchar | 50 | Уникальное имя пользователя |
| Password | varchar | 100 | Зашифрованный пароль пользователя |
| Salt | varchar | 100 | Соль для пароля |
| Email | varchar | 100 | Электронная почта пользователя |
| Key | integer | 4 | Ключ сессии |

Таблица «Requests» хранит информацию о входящих и исходящих запросах в друзья, структура которой представлена в таблице 3.2.

Таблица 3.2 – Структура таблицы «Requests»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID | integer | 4 | Идентификатор запроса |
| SenderId | integer | 4 | Внешний ключ, ссылающийся на идентификатор пользователя |
| ReceiverId | integer | 4 | Внешний ключ, ссылающийся на идентификатор пользователя |

Таблица «Friends» хранит информацию о связях, которые отображают дружбу между пользователями, структура таблицы представлена в таблице 3.3.

Таблица 3.3 – Структура таблицы «Friends»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID | integer | 4 | Идентификатор связи |
| FirstUserId | integer | 4 | Внешний ключ, ссылающийся на идентификатор пользователя |
| SecondUserId | integer | 4 | Внешний ключ, ссылающийся на идентификатор пользователя |

Таблица «Messages» хранит информацию о сообщениях. Структура таблицы представлена в таблице 3.4.

Таблица 3.4 – Структура таблицы «Messages»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID | integer | 4 | Идентификатор связи |
| Message | text | - | Текст сообщения |
| Read | boolean | 1 | Флаг, отображающий состояние сообщения (прочитано, не прочитано) |
| Date | datetime | 2 | Время отправки сообщения |

Таблица «Communication» хранит информацию об отправляемых сообщениях, структура которой представлена в таблице 3.5.

Таблица 3.5 – Структура таблицы «Communication»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID | integer | 4 | Идентификатор связи |
| MessageId | integer | 4 | Внешний ключ, ссылающийся на идентификатор сообщения |
| SenderId | integer | 4 | Внешний ключ, ссылающийся на идентификатор пользователя отправителя |
| ReceiverId | integer | 4 | Внешний ключ, ссылающийся на идентификатор пользователя получателя |

Таблица «DeliveringMessages» хранит информацию о сообщениях, которые, структура которой представлена в таблице 3.6.

Таблица 3.6 – Структура таблицы «Communication»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID | integer | 4 | Идентификатор связи |
| Key | integer | 4 | Внешний ключ, ссылающийся на идентификатор сообщения |
| CommunicationId | integer | 4 | Внешний ключ, ссылающийся на идентификатор пользователя отправителя |
| UserId | integer | 4 | Внешний ключ, ссылающийся на идентификатор пользователя получателя |

Продолжение таблицы 3.6

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| ID | integer | 4 | Идентификатор связи |

Схема базы данных представлена в виде ERD диаграммы в нотации Баркера на рисунке 3.1.

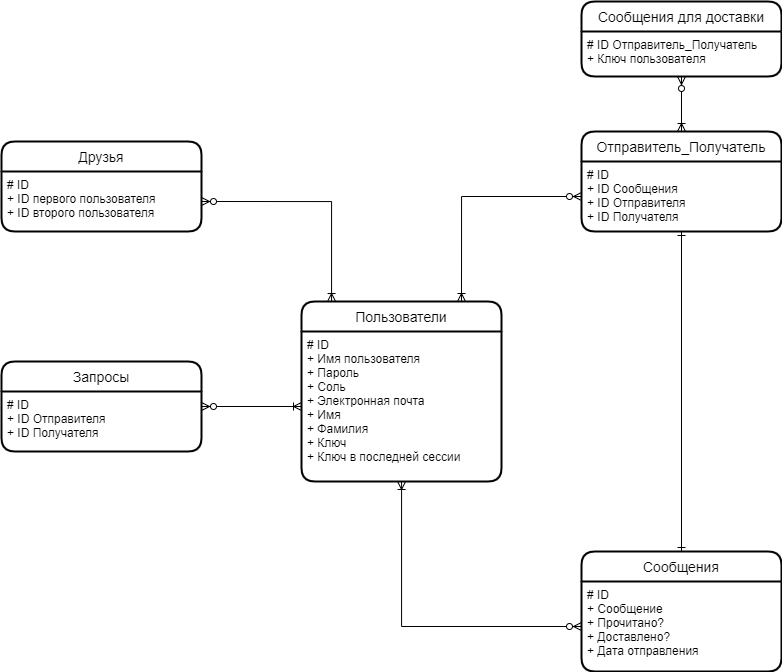


Рисунок 3.1

**3.4 Функции: логическая и физическая организация**

На основании диаграммы вариантов использования в программном средстве реализованы функции авторизации, регистрации и восстановления пароля.

Так как модель приложения имеет клиент-серверную структуру, код функций будет приводится как с сервера, так и с клиента.

Рассмотрим данные функции программы.

Для того, чтобы авторизоваться в приложении, пользователю необходимо ввести имя пользователя и пароль. Предварительно пользователь должен быть зарегистрирован в системе. После нажатия кнопки «Войти» клиент отправляет запрос на сервер, на котором уже производится дальнейшая обработка запроса. Если в результате проверки данные не были найдены в базе данных сервера, то вход в приложение отменяется и на клиенте отображается сообщение об ошибке.

Код функции авторизации на клиенте предоставлен ниже.

public static bool Login(string username, string password)

{

Clear();

QueryResult data;

while (true)

{

if (!Server.ConnectToServer()) break;

if (!Server.SendMessage("[LOG]")) break;

if (!Server.ReadMessage()) break; // + {SYS}

if (!Server.SendMessage(username)) break;

if (!Server.ReadMessage()) break; // + salt

string hashPassword = Server.ComputePasswordHash(password, Server.LastQueryResult.Data);

if (!Server.SendMessage(hashPassword)) break; // - hash

if (!Server.ReadMessage()) break;

data = Server.LastQueryResult;

if (data.Type == QueryResultType.Data)

{

// {101} + ключ

Username = username;

Key = data.Data;

return true;

}

break;

}

Server.Close();

data = Server.LastQueryResult;

if (data.Type == QueryResultType.Message)

Message = data.Data;

return false;

}

Код функции авторизации на сервере предоставлен ниже.

private bool Login()

{

SqlResultTable table;

string type;

string username;

string key;

string salt;

string hash;

type = "[LOG]";

username = GetData(false);

//Console.WriteLine($"[Server-Query] {type}: Login-{username}");

table = db.ExecuteQuery($"SELECT Salt FROM Users WHERE Login='{username}'");

// если пользователь имеется в БД, вытянуть соль

if (table.HasRows)

{

salt = table[0][0];

SendData("{101}" + salt);

hash = GetData(false);

table = db.ExecuteQuery($"SELECT ID FROM Users WHERE Password='{hash}' AND Login='{username}'");

// пароль верный

if (table.HasRows)

{

// Сгенерировать уникальный ключ

do

{

key = new Random().Next(100000000, 1000000000).ToString();

// Не занят ли ключ

table = db.ExecuteQuery($"SELECT ID FROM Users WHERE key='{key}'");

if (!table?.HasRows ?? true)

{

break;

}

} while (true);

// Обновить ключ

db.ExecuteQuery($"UPDATE Users SET key='{key}' WHERE Login='{username}'");

SendData("{101}" + key.ToString());

//CWLColor($"[Server-Query] {type}: Succes username -> {username}, {key}.", ConsoleColor.Cyan);

string id = GetIdByLogin(username);

clientInfo = new Client()

{

Username = username,

Key = key,

Salt = salt,

HashPassword = hash,

ID = id

};

AConsole.Print("Клиент подключен", clientInfo, type, ConsoleColor.Green);

return true;

}

// пароль не верный {301}

else

{

SendData("{301}");

AConsole.Print("Неверный пароль", clientInfo, type, ErrorColor);

}

}

// Пользователя нет в БД {302]

else

{

AConsole.Print("Неверное имя пользователя", clientInfo, type, ErrorColor);

SendData("{302}");

}

return false;

}

Для регистрации, пользователю необходимо перейти на форму регистрации, нажав кнопку «Регистрация». Далее пользователь должен заполнить все текстовые поля. Для успешной регистрации пользователю должен ввести уникальное имя пользователя, электронный адрес, пароль. После нажатия кнопки «Зарегистрироваться», клиент отправляет запрос на сервер.

Код функции регистрации на клиенте представлен ниже.

public static bool Register(string username, string email, string password)

{

Clear();

QueryResult data;

while (true)

{

if (!Server.ConnectToServer()) break;

if (!Server.SendMessage("[REG]")) break; // - type

if (!Server.ReadMessage()) break; // + {SYS}

if (!Server.SendMessage(username)) break; // - username

if (!Server.ReadMessage()) break; // + {SYS}

string salt = Server.GenSalt(16);

string hash = Server.ComputePasswordHash(password, salt);

if (!Server.SendMessage(hash)) break; // - hash

if (!Server.ReadMessage()) break; // + {SYS}

if (!Server.SendMessage(salt)) break; // - salt

if (!Server.ReadMessage()) break; // + {SYS}

if (!Server.SendMessage(email)) break; // - email

if (!Server.ReadMessage()) break; // + answer

//Server.Close();

data = Server.LastQueryResult;

if (data.Type == QueryResultType.Information)

{

// {100}

Message = "Аккаунт зарегистрирован успешно";

return true;

}

break;

}

// {304} | {305}

Server.Close();

data = Server.LastQueryResult;

if (data.Type == QueryResultType.Message)

Message = data.Data;

return false;

}

Код функции регистрации на сервере представлен ниже.

private bool Register()

{

SqlResultTable table;

string type;

string username;

string salt;

string hash;

string email;

type = "[REG]";

username = GetData(true);

hash = GetData(true);

salt = GetData(true);

email = GetData(false);

Console.WriteLine($"[Server-Query] {type}: Login-{username}");

Console.WriteLine($"[Server-Query] {type}: Email-{email}");

table = db.ExecuteQuery($"SELECT Login FROM Users WHERE lower(Login)='{username.ToLower()}'");

// Если пользователь с данным логином существует {304}

if (table.HasRows)

{

SendData("{304}");

AConsole.Print($"Данное имя пользователя уже занято", clientInfo, type, ErrorColor);

}

// Если логин свободен

else

{

table = db.ExecuteQuery($"SELECT Email FROM Users WHERE lower(Email)='{email.ToLower()}'");

// Если почта занята {305}

if (table.HasRows)

{

SendData("{305}");

AConsole.Print($"Данный электронный адрес уже занят", clientInfo, type, ErrorColor);

}

// Если почта не занятна {100}

else

{

db.ExecuteQuery($"INSERT INTO Users(Login, Password, Salt, Email) VALUES('{username}', '{hash}', '{salt}', '{email}')");

SendData("{100}");

clientInfo.Username = username;

clientInfo.ID = GetIdByKey(username);

AConsole.Print($"Пользователь зарегистрирован", clientInfo, type, ConsoleColor.Green);

}

}

return false;

}

Если пользователю, необходимо восстановить пароль, он должен перейти на форму восстановления пароля и ввести электронный адрес, на который был зарегистрирован аккаунт.

После нажатия кнопки «Восстановить», клиент отправляет запрос на сервере. Если на сервере была найдена почта, то сервер отправляет письмо с кодом восстановления.

Код функции восстановления пароля на клиенте предоставлен ниже.

public static bool ResetPassword(string newPassword)

{

Clear();

QueryResult data;

while (true)

{

if (!Server.ConnectToServer()) break;

if (!Server.SendMessage("[RESET\_PASSWORD]")) break;

if (!Server.ReadMessage()) break; // + {SYS}

string salt = Server.GenSalt(16);

string hash = Server.ComputePasswordHash(newPassword, salt);

string hashAndSalt = hash + "|" + salt;

if (!Server.SendMessage(hashAndSalt)) break;

if (!Server.ReadMessage()) break; // + answer

data = Server.LastQueryResult;

if (data.Type == QueryResultType.Information)

{

// {100} - Пароль изменен

return true;

}

break;

}

// {316}

Server.Close();

data = Server.LastQueryResult;

if (data.Type == QueryResultType.Message)

Message = data.Data;

return false;

}

Код функции восстановления пароля на сервере, представлен ниже.

private bool ResetPassword()

{

string type;

string hashANDSalt;

string hash;

string salt;

type = "[RESET\_PASSWORD]";

hashANDSalt = GetData(false);

if (string.IsNullOrEmpty(resetingClientInfo.Email))

{

AConsole.Print($"Изменить пароль невозможно, т.к. сессия была прервана", type, ConsoleColor.DarkYellow);

SendData("{317}");

return false;

}

hash = hashANDSalt.Substring(0, hashANDSalt.IndexOf('|'));

salt = hashANDSalt.Substring(hashANDSalt.IndexOf('|') + 1);

db.ExecuteQuery($"UPDATE Users SET Password='{hash}', Salt='{salt}' WHERE Email='{resetingClientInfo.Email}'");

// Пароль восстановлен

if (db.CountHandledRows > 0)

{

AConsole.Print($"Пароль изменен успешно", type, ConsoleColor.DarkYellow);

SendData("{100}");

}

// Пароль не восстановлен

else

{

AConsole.Print($"Не удалось восстановить пароль", type, ErrorColor);

SendData("{316}");

return true;

}

return false;

}

Полный текст программы представлен в приложении А.

**3.5 Входные и выходные данные**

Входными являются данные, введённые пользователем в таблицы базы данных: «Users», «Messages», «Friends», «DeliveringMessages».

Входными данным при добавлении данных о новом пользователе будут являться:

* имя пользователя;
* пароль;
* адрес электронной почты.

Входными данными при добавлении данных о новом сообщении являются:

* текст сообщения;
* дата отправки.

Входными данными при добавлении друзей являются:

* id первого пользователя;
* id второго пользователя.

Вся вносимая в программное средство информация хранится на сервере в базе данных.

Выходные данные – это полученные с сервера массивы данных, преобразующиеся в понятный для пользователя вид, отображаемые в качестве списков. Пример выходных данных представлен на рисунке Б.1.

**3.6 Функциональное тестирование**

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям и проводится для выявления неполадок и недочетов программы на этапе ее сдачи в эксплуатацию.

Проведем тестирование проверки каждого пункта меню с целью проверки всех функций.

Тестирование программы будет производиться последовательно, переходя из одной части программы в другую. Во время теста будут проверяться все действия с программой, навигация пунктам меню, которые может произвести пользователь.

В таблице 3.7 представлены тест-кейсы, подготовленные для проведения функционального тестирования.

Таблица 3.7 – Тест-кейсы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Модуль / Функция | Шаги выполнения | Ожидаемый результат | Фактический результат |
| 1 | Авторизация | 1. Перейти на вкладку «Вход».  2. В поле «Имя пользователя» ввести строку «rPypty».  3. В поле «Пароль» ввести строку «password».  4. Нажать кнопку «Войти». | Генерируется новый ключ сессии для данного пользователя и записывается в базу данных. Открывается главная форма. | В базу данных добавился ключ сессии. Открылась главная форма. Результат представлен на рисунке В.1 |
| 2 | Регистрация с занятым именем пользователя | 1. Перейти на вкладку «Регистрация».  2. В поле «Имя пользователя» ввести строку «rPypty».  3. В поле «Почта» ввести строку «yourmail@gmail.com»  4. В поле «Пароль» ввести строку «passs123».  5. Нажать кнопку «Зарегистрироваться». | Пользователь не регистрируется, данные не добавляются в базу данных, отображается сообщение с ошибкой. | Отобразилось сообщение с ошибкой регистрации. Результат представлен на рисунке В.2 |
| 3 | Восстановление аккаунта | 1. Перейти на вкладку «Восстановление»  2. В поле «Почта» ввести почту, указанную при регистрации.  3. Нажать кнопку «Восстановить».  4. В поле «Код» ввести код, отправленный на почту.  5. Нажать кнопку «Далее».  6. В поле «Пароль» введите «pas»  7. В поле «Подтвердите пароль» введите «pas» | На почту отправляется код восстановление, новый пароль перезаписывается в базе данных, отображается сообщение об успешной операции. | Код пришел на почту, пароль был восстановлен. Результат представлен на рисунке В.3 |

Продолжение таблицы 3.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Модуль / Функция | Шаги выполнения | Ожидаемый результат | Фактический результат |
| 4 | Отправка запроса в друзья | 1. Войдите в аккаунт.  2. Перейдите на вкладку «Друзья».  3. В поле «Поиск» введите имя пользователя «rPypty2».  4. Нажмите на кнопку добавления. | В базу данных добавляется новый запрос в друзья.  На вкладке «Друзья» => «Запросы» отобразится новый запрос. | В базу данных добавились новые данные. На вкладке «Друзья» => «Запросы» отобразится новый запрос. Результат представлен на рисунке В.4 |
| 5 | Изменение пароля | 1. Войти в аккаунт.  2. Перейдите на вкладку «Мой профиль».  3. В поле «Старый пароль» ввести текущий пароль.  4. В поля «Новый пароль» и «Подтверждение пароля» введите новый пароль.  5. Нажмите кнопку «Изменить». | Пароль изменяется, данные в базе данных перезаписываются, отображается сообщение. | Сообщение отобразилось, пароль изменился. Результат представлен на рисунке В.5 |
| 6 | Выход из аккаунта | 1. Войти в аккаунт.  2. Перейти на вкладку «Мой профиль».  3. Нажать кнопку «Выйти». | Главная форма сменяется формой входа.  Сессионный ключ обнуляется. | Главная форма сменилась формой входа. Результат представлен на рисунке В.6 |
| 7 | Отправка сообщения | 1. Войти в аккаунт.  2. Перейти на вкладку «Сообщения».  3. Нажать на пользователя в списке диалогов.  4. В поле «Сообщение» ввести данные.  5. Нажать кнопку отправки. | Сообщение отправляется.  Текстовое поле очищается.  Сообщение появляется в списке. | Сообщение отправилось, текстовое поле очистилось. Результат представлен на рисунке В.7 |

**3.7 Описание справочной системы**

Для эффективной работы пользователя с приложением необходимо обеспечить его качественной справочной системой, в которой должны быть приведены методы и приемы работы с приложением, включающие данные о том, что произойдет после нажатия на определенную кнопку или при выборе пункта меню, сведения о том, какую информацию и в каком виде следует вводить в соответствующие поля, каким образом можно работать.

Справочная система будет представлена в виде встроенного файла в приложении, содержащая информацию по эксплуатации программного средства. Файл будет разработан в программе Dr. Explain.

Система справки данного программного средства будет содержать раздел «Руководство по использованию». Он будет подразделяться на три подраздела:

«Управление аккаунтом»;

«Переписка с пользователями»;

«Система друзей».

Подраздел «Управление аккаунтом» будет разделен на разделы «Вход», «Регистрация», «Восстановление пароля», «Изменение пароля».

Подраздел «Переписка с пользователями» будет содержать информацию о том, как начать переписку с определенным пользователем.

Подраздел «Система друзей» будет иметь разделы «Друзья», «Запросы».

Структура справочной системы приведена на рисунке 3.2.

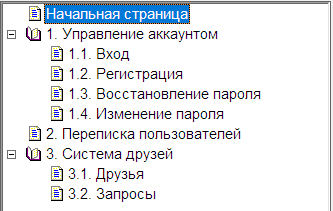


Рисунок 3.2

**4** **Применение**

**4.1 Назначение программного средства**

Программное средство «RPYNOD» разрабатывалось с целью создания приложения, с помощью которого люди смогут вести переписку текстовыми сообщениями друг с другом. В приложении присутствует система друзей, позволяющая добавлять новый пользователей, а также просматривать список запросов. Данная программа была разработана на ноутбуке Huawei Matebook D 14 с операционной системой Windows 10 Home, со следующей конфигурацией:

* процессор AMD Ryzen 7 3700U;
* интегрированная видеокарта Radeon Vega Mobile Gfx 2.30 GHz;
* ОЗУ 8,00 ГБ.

Программа создана в средстве разработки Microsoft Visual Studio 2019 на языке программирования С#. Она может работать в средах операционных систем семейства Microsoft Windows, начиная с Windows 7.

Для работы программы, необходима серверная часть, имеющая базу данных «Database.db». Программа не требовательна к системным ресурсам и не имеет больших массивов данных, также проста в использовании и не требует специальных навыков при работе. Для работы данного программного средства необходима предварительная установка и настройка следующих программных продуктов:

* платформа Microsoft Net Framework версии 4.5.

В состав программных входят:

* программа «RPYNOD.exe»;
* программа «Server.exe»;
* база данных «Database.db»;
* файл справочной системы «help.chm».

Объем установленного приложения не превышает 140 МБ, файлов – 60 МБ.

Программное средство требует логина и пароля для авторизации пользователей.

Приложение «RPYNOD.exe» предназначено для переписки пользователей. Программное средство предоставляет пользователю возможности для отправки текстовых сообщений, поиска пользователей, добавления, удаления отображения друзей и запросов, сохранения истории сообщений с пользователями.

**4.2 Условия применения**

Для работы с программой необходимо наличие программного обеспечения:

* операционная система, начиная с Windows 7;
* библиотека Microsoft .NET Framework 4.5.;
* наличие интернета со скоростью не ниже 8-10 Мбит/с.

На случай редактирования проекта программы необходимо наличие программного обеспечения:

* система управления базами данных SQLiteStudo;

**Заключение**

В рамках курсового проекта на тему «Программное средство для переписки пользователей – мессенджер «RPYNOD» было разработано программное средство «RPYNOD.exe».

Основными функциями приложения являются:

* ведение базы данных на сервере, в которой будет храниться вся информация;
* ведение локальной базы данных на клиенте, для сокращения количества обращений к серверу;
* авторизация пользователей;
* регистрация пользователей;
* восстановления пароля с помощью email;
* изменение пароля;
* переписка пользователей текстовыми сообщениями, с использованием текстовых сообщений и стикеров;
* отправление запросов в друзья другим пользователям;
* поиск других пользователей;
* отклонение, либо принятие запросов в друзья.

Для достижения целей курсового проекта были решены следующие задачи:

* изучена предметная область;
* разработана физическая и логическая модель данных;
* разработано программное средство;
* описана область применения созданного программное средство.

Разработка имеет интуитивно понятный графический интерфейс, позволяющий даже с минимальным знанием компьютера использовать данное программного средство.

Программа реализована в полном объеме и в соответствии с заданными требованиями, полностью отлажена и протестирована. Поставленные задачи выполнены.

К достоинствам программного средства можно отнести простоту использования, малые массивы данных. К недостаткам можно отнести …

В процессе разработки данного программного средства были применены и закреплены знания по уже изученному материалу, были отработаны навыки владения методами надежного программирования и эффективности разработки программного обеспечения в Microsoft Visual Studio 2019 с использованием языка программирования C#, разработана база данных средствами системы управления базами данных SQLiteStudio.

**Список использованных источников**

1. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск : КБП, 2023.
2. Михалевич В.Ю. Методические указания к курсовому проектированию по учебному предмету «Конструирование программ и языки программирования» / В.Ю.Михалевич. – Минск : КБП, 2023.
3. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с
4. .NET Framework [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : https://docs.microsoft.com/en-us/dotnet/framework/. – Дата доступа : 01.09.2020.
5. Dr.Explain [Электронный ресурс]. – Dr.Explain, 2021. – Режим доступа : https://www.drexplain.ru/. – Дата доступа : 04.05.2021.
6. MS Office [Электронный ресурс]. – Microsoft, 2021. – Режим доступа : https://officeapplications.net/microsoft-word/. – Дата доступа : 03.05.2021.
7. Smart Install Maker [Электронный ресурс]. – InstallBuilders, 2020. – Режим доступа : https://www.ixbt.com/news/soft/index.shtml?10/52/79/. – Дата доступа : 05.05.2021.
8. Diagrams [Электронный ресурс]. – UML, 2021. – Режим доступа : https://www.diagrams.net/. – Дата доступа : 10.05.2021.
9. Visual Studio [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : https://visualstudio.microsoft.com/vs/. – Дата доступа : 01.09.2020.
10. Что такое мессенджер и для чего он нужен + список самых популярных мессенджеров и рекомендации по выбору [Электронный ресурс]. – https://myrouble.ru/, 2020. – Режим доступа : https://myrouble.ru/messenger/. – Дата доступа : 12.12.2020.

Указываем все-все источники

**Приложение А**

(обязательное)

**Текст программных модулей**

10-15 страниц кода

using Messenger.Extentions;

using Messenger.Models;

using Messenger.ViewModels;

using System;

using System.Linq;

using System.Threading;

namespace Messenger

{

static class ServerManipulation

{

#region Properties

public static string Username { get; set; } = null;

public static string Key { get; set; } = null;

public static string Message { get; set; } = null;

public static object Data { get; set; } = null;

public static MessageModel[] NewMessages { get; set; } = null;

#endregion

#region Methods

public static bool Login(string username, string password)

{

Clear();

QueryResult data;

while (true)

{

if (!Server.ConnectToServer()) break;

if (!Server.SendMessage("[LOG]")) break;

if (!Server.ReadMessage()) break; // + {SYS}

if (!Server.SendMessage(username)) break;

if (!Server.ReadMessage()) break; // + salt

string hashPassword = Server.ComputePasswordHash(password, Server.LastQueryResult.Data);

if (!Server.SendMessage(hashPassword)) break; // - hash

if (!Server.ReadMessage()) break;

data = Server.LastQueryResult;

if (data.Type == QueryResultType.Data)

{

// {101} + ключ

Username = username;

Key = data.Data;

return true;

}

break;

}

Server.Close();

data = Server.LastQueryResult;

if (data.Type == QueryResultType.Message)

Message = data.Data;

return false;

}

**Приложение Б**

(обязательное)

**Формы выходных документов**

Номера рисунков и скриншоты, на которые ссылаетесь в пункте 3.5

**Приложение В**

(обязательное)

**Результаты работы программы**

Номера рисунков и скриншоты, на которые ссылаются тест-кейсы