

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**   
**«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»**

Факультет Физико-технический

Кафедра Компьютерных технологий (КТ)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | КТ |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | Т. В. Ермоленко |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе бакалавра 3 курса

на тему:

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ СИСТЕМЫ ПЛАНИРОВАНИЯ ЗАДАЧ И ЦЕЛЕЙ

|  |  |  |  |
| --- | --- | --- | --- |
| Автор работы |  |  | Е.С. Ермоленко |

подпись

|  |  |  |
| --- | --- | --- |
| Направление | 09.03.01 | Информатика и вычислительная техника |

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель Котенко В.Н.

подпись

Консультанты по разделам:

Техническое задание \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент Т.В. Шарий

подпись

Нормоконтроль \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. лаборант В.Г. Медведева

подпись

Курсовая работа защищена \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата итоговая оценка комиссия

Подписи членов комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Донецк  
2020

ГОУ ВПО «Донецкий национальный университет»

кафедра Компьютерных технологий

Утверждаю

Зав. кафедрой КТ

\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

**ЗАДАНИЕ**

на курсовую работу студента 3 курса Ермоленко Е.С.

*Тема курсовой работы*: Проектирование и реализация системы планирования задач и целей

*Краткая постановка задачи*: 1. Изучить и проанализировать предметную область по реализации систем планирования задач и целей. 2. Оценить актуальность и состояние вопроса на данный момент, а также выполнить моделирование существующих бизнес-процессов. 3. Ознакомиться с программными продуктами для создания задач и целей.   
4. Разработать техническое задание на создание приложения. 5. Разработать проект программного обеспечения системы: спроектировать концептуальную, логическую и физическую модели базы данных, создать базу данных для хранения данных о пользователях, направлениях, целях, задач, расписаниях, достижениях и данных для авторизации в сервисах. 6. Разработать средствами языка C# приложение, которое позволяет создавать цели, направления, расписания, хранить данные для авторизации в сервисах и вести личную статистику эффективности выполнения задач.   
7. Протестировать программное обеспечение. 8. Оформить отчёт.

*Исходные данные*: 1. Документация о реализации систем планирования задач и целей. 2. Документация по языку программирования C#. 3. Документация по встраиваемой кроссплатформенная базе данных SQLite3.

*Ожидаемые результаты*: система планирования задач и целей

*Календарный план работы:*

|  |  |  |
| --- | --- | --- |
| Даты консультаций | Этапы выполнения работы | Отметки о выполнении |
| 03.02.2020 | Постановка задачи и обсуждение литературы | выполнено |
| 07.02.2020 | Предварительное утверждение содержания отчёта | выполнено |
| 17.02.2020 | Утверждение проекта, алгоритмов, методов, технологий | выполнено |
| 03.03.2020 | Ход реализации проекта | выполнено |
| 23.03.2020 | Обсуждения организации тестирования программы | выполнено |
| 10.04.2020 | Демонстрация программного продукта руководителю | выполнено |
| 17.04.2020 | Оформление отчёта | выполнено |
| 24.04.2020 | Предоставление отчёта руководителю | выполнено |

Дата выдачи задания 03.02.2020 года

Студент Е.С. Ермоленко

Руководитель В.Н. Котенко

**АННОТАЦИЯ**

Отчет о курсовой работе: 79 с., 40 рис., 8 табл., 2 приложения,   
12 источников.

Объект исследования – цели и задачи.

Предмет исследования – программный продукт для планирования задач и целей.

Цель работы – разработка прикладного программного приложения для нескольких пользователей, которая позволяла бы создавать направления и цели для каждого из направлений. Каждая цель имела бы несколько задач и расписаний.

Метод исследования – анализ возможностей языка С# и встраиваемой базы данных SQLite3.

В работе были использованы возможности таких программных продуктов как:

1. Visual Studio 2019;
2. DB Browser for SQLite;
3. Adobe Photoshop 2019;
4. Figma.

В результате решения задачи было разработано приложение для планирования задач и целей. Приложение позволяет создавать задачи, которые относятся к какой-либо цели и направлению, также предоставляет возможность создавать расписания, получать достижения по ходу пользования приложением и хранить данные для авторизации в сервисах.

Главный модуль приложения работает с объектами классов. Главными операциями приложения являются:

1. Регистрация пользователя;
2. Авторизация пользователя;
3. Вывод общей эффективности пользователя по задачам;
4. Вывод статистики пользователя по задачам за день, неделю, месяц;
5. Ведение прогресса получения достижения;
6. Хранение данных пользователя для авторизации в интернет ресурсе;
7. Изменение цветового оформления приложения;
8. Изменение данных пользователя;
9. Создание, изменение, удаление задачи;
10. Создание, изменение, удаление направления;
11. Создание, изменение, удаление цели;
12. Создание, удаление расписания;
13. Создание, удаление, заметки.

Приложение Upgrade может использоваться любыми пользователями для планирования целей и ведения личной статистики по задачам, а также для хранения данных для авторизации в интернет ресурсе.

С#, ПРОГРАММНЫЙ ПРОДУКТ, ФИЗИЧЕСКА ЛАБОРАТОРИЯ, ЛАБОРАТОРНАЯ РАБОТА , СИЛА ТОКА, ПРОВОДНИК, НАПРЯЖЕНИЕ

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 7](#_Toc56632523)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 9](#_Toc56632524)

[1.1 Состояние вопроса 9](#_Toc56632525)

[1.2 Актуальность и цель работы 15](#_Toc56632526)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 18](#_Toc56632527)

[2.1 Описание области применения и исходных данных приложения 18](#_Toc56632528)

[2.2 Требования к пользовательским интерфейсам 18](#_Toc56632529)

[2.3 Требования к аппаратным, программным интерфейсам 25](#_Toc56632530)

[2.4 Требования к пользователям продукта 25](#_Toc56632531)

[2.5 Функции продукта 25](#_Toc56632532)

[2.6 Ограничения 26](#_Toc56632533)

[3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ 27](#_Toc56632534)

[4 РАЗРАБОТКА БАЗЫ ДАННЫХ ПРИЛОЖЕНИЯ 34](#_Toc56632535)

[4.1 Концептуальное проектирование базы данных 34](#_Toc56632536)

[4.2 Логическое проектирование базы данных 34](#_Toc56632537)

[4.3 Физическое проектирование базы данных 34](#_Toc56632538)

[5 РАЗРАБОТКА СИСТЕМЫ ПЛАНИРОВАНИЯ ЗАДАЧ И ЦЕЛЕЙ 35](#_Toc56632539)

[5.1 Входные и выходные данные приложения 35](#_Toc56632540)

[5.2 Проектирование структуры приложения 37](#_Toc56632541)

[5.3 Описание алгоритма работы объектов и SQL-запросов системы 40](#_Toc56632542)

[6 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 47](#_Toc56632543)

[6.1 Аппаратные и программные средства создания и эксплуатации программного обеспечения 47](#_Toc56632544)

[6.2 Руководство пользователя 47](#_Toc56632545)

[6.3 Описание контрольных примеров 50](#_Toc56632546)

[ЗАКЛЮЧЕНИЕ 53](#_Toc56632547)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 54](#_Toc56632548)

[ПРИЛОЖЕНИЕ А Экранные формы 56](#_Toc56632549)

[ПРИЛОЖЕНИЕ Б Фрагменты листинга 59](#_Toc56632550)

# ВВЕДЕНИЕ

В последнее время темп жизни настолько увеличился, что удержать все в голове практически невозможно. Специально для тех, кто не хочет держать в голове несколько задач подряд были созданы планировщики задач. Планирование – это одна из составных частей управления (менеджмента), заключающаяся в разработке и практическом осуществлении планов, определяющих будущее состояние какой–либо структуры или системы. Планирование можно трактовать как умение предвидеть цели организации, результаты ее деятельности и ресурсы, необходимые для достижения поставленных целей.

В то же время, планирование – это систематическая подготовка принятия решений о целях, средствах и действиях путем целенаправленной сравнительной оценки различных альтернативных действий в ожидаемых условиях.

В свою очередь план – это упорядоченный список дел и задач, которые необходимо выполнить в определенное время (в определенном месте). Планирование помогает выполнять задачи вовремя, качественно и не терять при этом время. Без планирования в современном обществе не решаются ни производственные ни бытовые задачи. Современный человек вынужден планировать свою жизнь, начиная с распорядка дня и заканчивая долгосрочными планами. Качественное планирование имеет массу преимуществ: поможет стать более ответственным, собранным; позволит концентрироваться, ставить перед собой задачи, цели; расширяет диапазоны мышления; дает возможность самореализации; повышает уровень осознанности; развивает гибкость мышления. Тем, кто смутно представляет, какие бывают планы, и еще не научился планировать, следует приступить немедленно [1].

В данной курсовой работе будет реализовано приложение, которое позволит систематизировать задачи пользователя, а также контролировать развитие какого-либо жизненного направления путем анализа статистики и будет отличным помошником в повседневной жизни.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Состояние вопроса

Правильно составленный план на бумаге – наполовину выполненное дело. Каждодневное планирование необходимо для четкого выполнения поставленных задач в фиксированные сроки. Сложные задачи дробятся на мелкие подзадачи по принципу «дерева решений». План корректируется в течение дня по мере выполнения задач. Жизнь по плану – смелое и разумное решение. Не каждый решится на такое. Именно поэтому в нашем обществе так мало успешных и уверенных в себе людей. Жизненный план дает возможность не плыть по течению событий, не жить планами и решениями других людей. Необязательно все запланированные события сбудутся. Но они подтолкнут к новым решениям и стремлениям.

Распланированная жизнь имеет следующие преимущества:

1. Жизнь по плану интересна. В жизни каждого человека происходит множество разнообразных событий. Для праздной лени просто не остается времени.
2. Личностный рост и развитие человека. Всестороннее развитие от множества пережитых событий и возможности уделять любимым занятиям достаточно времени.
3. Свобода творчества. Когда вы планируете свою жизнь самостоятельно, запускается творческое мышление, развивается креативность.
4. Жизнь здесь и сейчас. Планирование не дает возможности «застревать» в прошлом, стимулирует к движению вперед, открытому взгляду в завтрашний день [1].

На данный момент существует несколько приложений-аналогов, такие как:

Приложение «Any.do». приложение с простым и понятным интерфейсом, которое обеспечивает быстрое и простое управление задачами и даже интегрируется с приложением iOS Reminders. Таким образом, вы можете сообщить Siri напоминание, и оно появится в Any.do. Однако синхронизация работает только в одном направлении: удаление задач из Any.do не приведет к его удалению из приложения напоминаний для iPhone, но, если вы используете Any.do в качестве основного приложения, это не будет проблемой. Есть также дополнительные удобные функции, такие как автоматическая сортировка списков продуктов и «планировка дня», чтобы помочь расставить приоритеты для задач. Синхронизация между ПК, планшетом и телефоном - еще одно приятное дополнение к поддержке iOS и Android [2].

Преимущества:

1. **Хорошая интеграция со смартфоном;**
2. **Кросс-платформенность.**

**Недостатки:**

1. **Цветовая палитра только в платной версии.**

Данное приложение является условно бесплатным [2]. Окно работы в приложении представлено на рисунке 1.1.

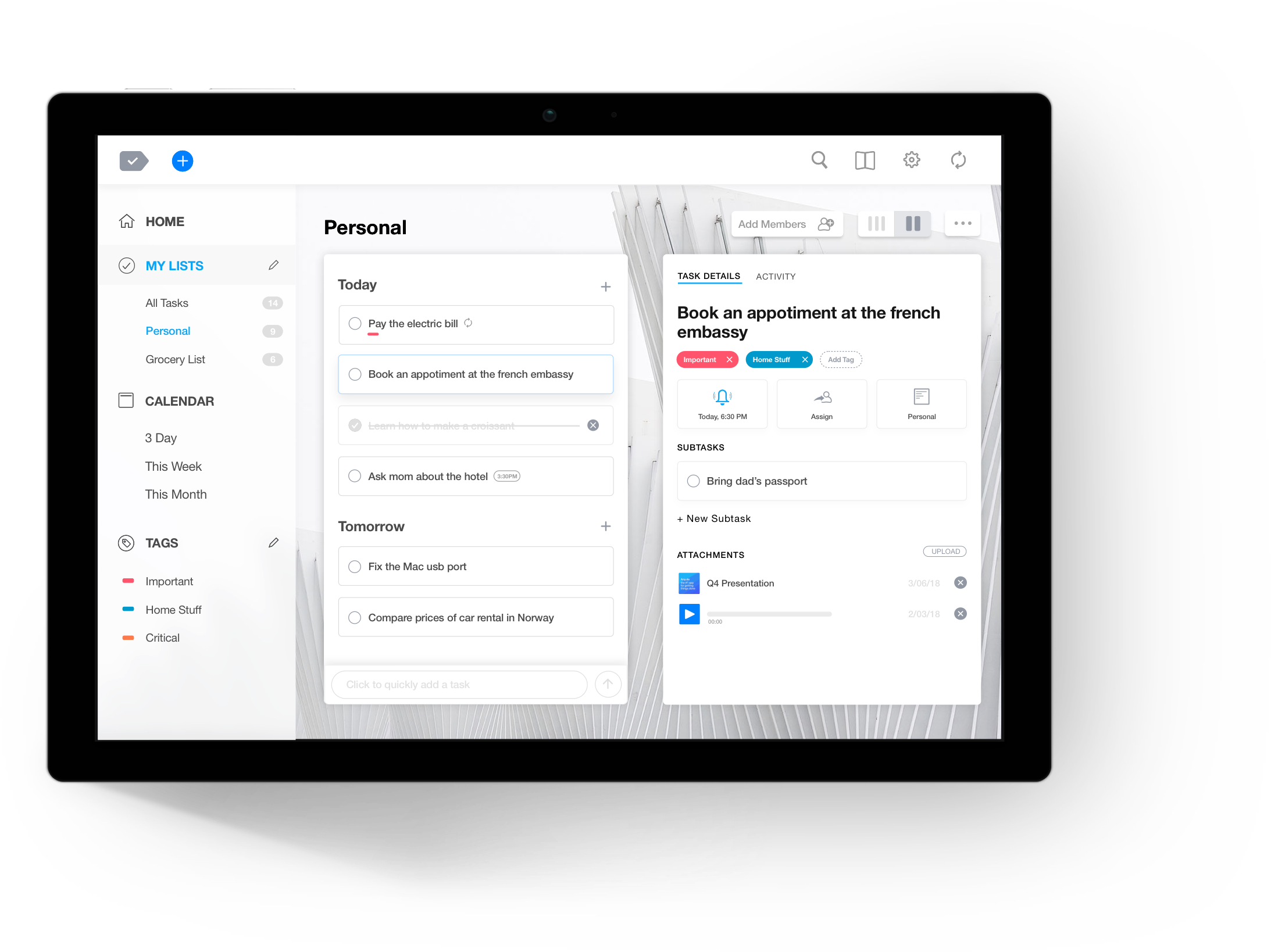
****

Рисунок 1.1 – Окно работы в приложении

Приложение «Google Tasks». Простой, как и список дел на бумаге. Это потрясающе минималистичное и хорошо разработанное приложение, которое делает именно то, что должно, и не более того. Вы можете создавать задачи, составлять их описание, а затем добавлять подзадачи. Они появляются в маркированном списке, и можно пометить каждую подзадачу завершенной, когда придет время. Каждая задача находится под списком, и количество списков, которые вы можете создать, не ограничено. Вы можете иметь список покупок, список дел и многое другое. В обмен на простоту Google Tasks теряем некоторые более глубокие теги и организационные функции, которые вы можете найти в других приложениях. Задачи Google доступны на iOS и Android. Если вы используете Gmail в Интернете, вы можете увидеть обзор своих задач в правом краю интерфейса, рядом с приложениями Календарь и Google Keep [2].

Преимущества:

1. Бесплатность**;**
2. Простой интерфейс**.**

**Недостатки:**

1. Ограниченные возможности детализации и планирования**.**

Данное приложение является условно бесплатным [2]. Окно работы в приложении представлено на рисунке 1.2.

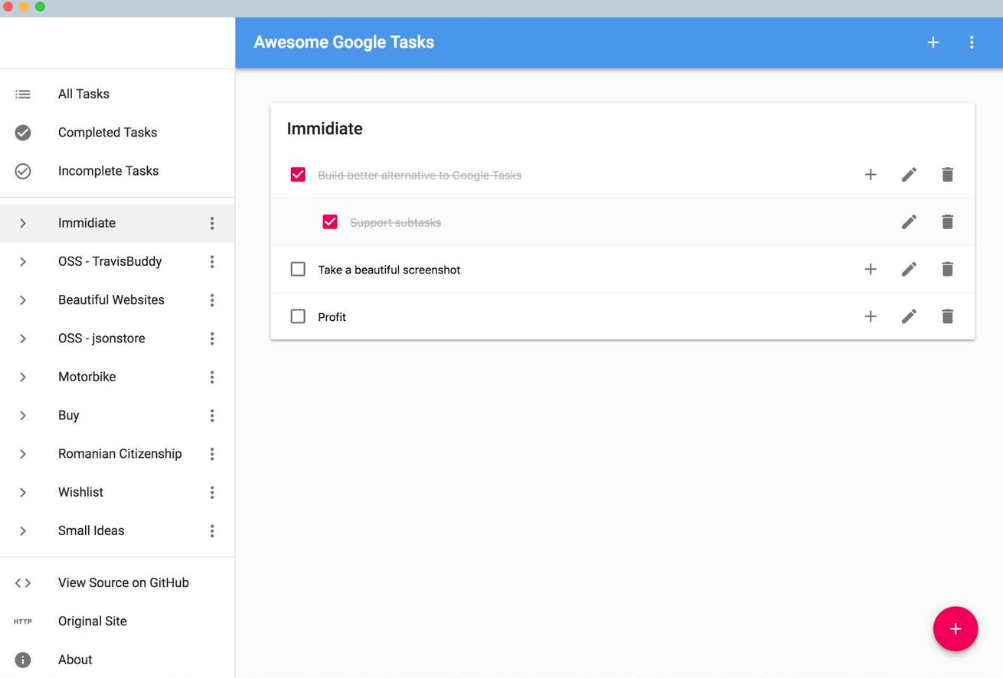


Рисунок 1.2 – Окно работы в приложении

Приложение «Todoist». Данное приложение одно из крупнейших приложений с огромным количеством пользователей и проверенной годами эффективностью. Вы можете зарегистрироваться с помощью своего профиля в Facebook или учетной записи Google, и начать работу так же просто, как ввести свою первую задачу и нажать «Отправить». Можно установить крайний срок выполнения, а также назначить приоритет задаче, или положить ее в группу схожих задач. Выполнение включает в себя установку галочки рядом с задачей, и есть определенное удовлетворение от этого действия и его анимации. Настройка напоминаний о задачах, дополнительные активные проекты, комментарии к задачам и автоматическое резервное копирование - все это дополнительные функции.

Преимущества:

1. Популярность**;**
2. Большое количество настроек**.**

**Недостатки:**

1. **Ограниченная функциональность в бесплатной версии.**

Данное web-приложение является условно бесплатным [2]. Окно для работы представлено на рисунке 1.3.

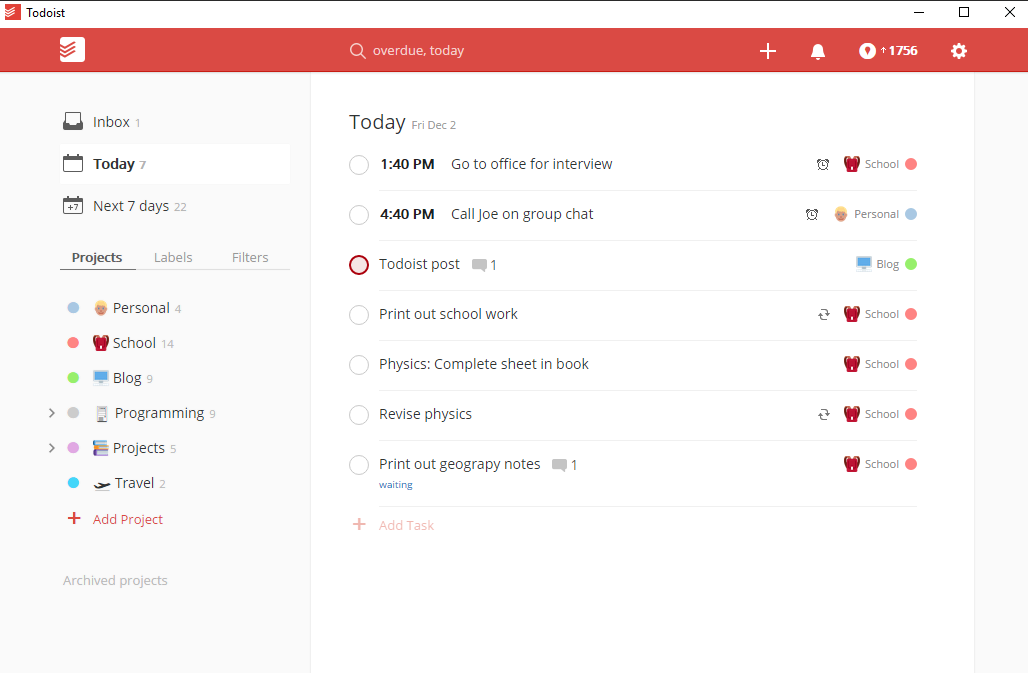


Рисунок 1.3 – Окно для работы в приложении

Приложение «Things». Менеджер задач Things предназначен для личного использования. Предлагается с постоянной лицензией, но на каждое устройство её нужно приобретать отдельно. Русский язык поддерживается.

По времени задачи в Things распределяются по категориям «Сегодня», «Планы», «В любое время», «Когда-нибудь». Кроме того, списки задач можно разделить по виду деятельности. Для больших задач можно создавать проекты, которые будут состоять из нескольких этапов. Сервис позволяет добавить в карточку задачи чек-лист, описание, прикрепить ссылку на файл. Интеграции с другими сервисами не предусмотрены [3].

**Недостатки:**

1. **Только для macOS и IOS;**
2. **Платный.**

Окно для работы представлено на рисунке 1.4.

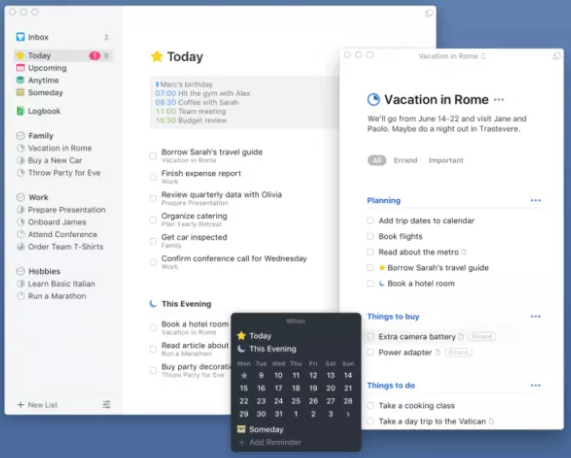


Рисунок 1.4 – Окно для работы в приложении

1.2 Актуальность и цель работы

Цель курсового проекта – создание прикладного программного обеспечения для планирования целей и ведении статистики по прогрессу выполнения задач.

Для того, чтобы примерно понимать насколько важно планирование и создание целей можно создать приблизительный план по некоторым жизненным направлениям. Направление - учеба и карьера. Цель – поступить в университет, магистратуру, аспирантуру. Для этого надо подтянуть успеваемость по определенным предметам. Умеющий составлять планы сумеет спланировать решение любой задачи. Если нет возможности пользоваться услугами репетитора, необходимо составить план ежедневных самостоятельных занятий по определенным темам с использованием литературы и интернет-ресурсов. При этом необходимо составлять план каждого занятия, отмечать каждую пройденную и закрепленную тему, выполнять план строго по пунктам. В конце каждого занятия-начале следующего необходимо проводить тест на проверку усвоенного материала.

Цель – получить работу. Для этого нужно составить резюме, посмотреть требования компании к профессии.

Жизненное направдение – здоровье. Цель – похудеть или придать рельефность телу. Для этого нужно определиться, где будут проходить тренировки: дома или в фитнес-клубе. Заниматься в тренажерном центре всегда приоритетнее, так как график занятий всегда организует и обязывает, а профессиональный тренер ответит на любой волнующий вопрос. Если заниматься дома, то нужно самостоятельно составить программу домашних тренировок, вести дневник питания и тренировок, фиксировать все результаты.

Отдых, свободное время, хобби. В этой сфере жизни нужно определиться, сколько времени уделять хобби, а сколько отдыху. Здесь можно дать волю фантазии, потому что проводить свободное время можно разными способами в зависимости от предпочтения.

Духовность. Каждый сам решает, как развивать духовность. Для кого-то это развитие личностных качеств путем чтения различных книг, прививания полезных привычек, самообразования в духовной сфере, изучение культуры народов, истории и многого другого [1].

Исходя из разнообразия и количества жизненных направлений целесообрахно использовать дополнительный ресурс, чтобы не запутаться и видеть конкретные шаги для достижения цели. Таким ресурсом может выступить система планирования задач и целей.

Проведя анализ приложений аналогов, сделал вывод, что актуально будет разработать программу, которая будет иметь приятный и понятный интерфейс, а также предоставлять возможность вручную создавать неограничеенное количество направлений и целей к ним, также реализовать функцию расписаний, для повторяющихся задач. Также приложение должно иметь и функционал для мониторинга эффективности конкретного пользователя в общем виде и на определенный промежуток времени, чем не могут похвастаться приложения аналоги. И дополнительным функционалом реализовать хранение данных для авторизации пользователя в интернет ресурсах и создание заметок. Данное приложение будет полностью бесплатным, что выделит его среди конкурентных приложений.

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

2.1 Описание области применения и исходных данных приложения

Название программного продукта: «Upgrade».

Программный продукт предназначен для планирования целей и задач, ведения статистики по задачам и хранению дополнительных пользовательских данных. Программа предназначена для использования любыми типа пользователей.

Программный продукт является самодостаточным и автономным, а также данные для работы приложения находятся в базе данных.

Программный продукт должен реализовывать создание направлений, целей, задач, расписаний и вывод статистических данных по задачам пользователя. Также необходимо предусмотреть настройку внешнего вида приложения и создание, изменение данных пользователя.

Исходными данными приложения являются созданные пользователем задачи.

2.2 Требования к пользовательским интерфейсам

Интерфейс должен быть реализован для отображения на мониторах минимум 1366х768 пикселей. Кнопки и другие элементы управления должны иметь стандартные размеры. Цветовое оформление окон должно быть светлым, с акцентным и вторичным цветами. Предпочтений в оттенках нет, должна быть реализована функция изменения акцентного цвета.

Окно входа в систему должно предполагать несколько полей для ввода данных для авторизации и кнопки для входа в систему. Макет окна входа в систему представлен на рисунке 2.1.

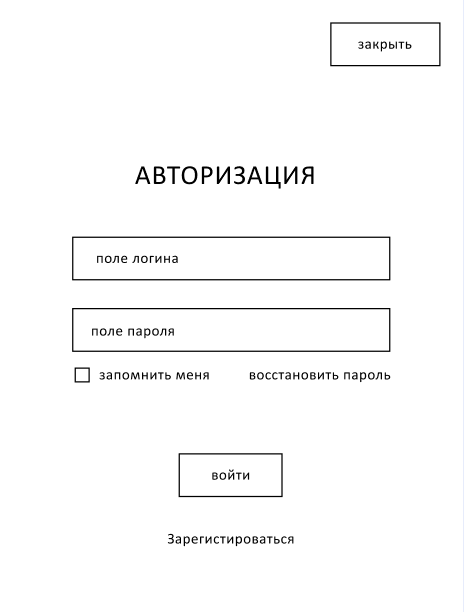


Рисунок 2.1 – Макет окна входа в систему

Окно регистрации нового пользователя должно предполагать несколько полей для ввода данных для регистрации, макет окна для регистрации представлено на рисунке 2.2.

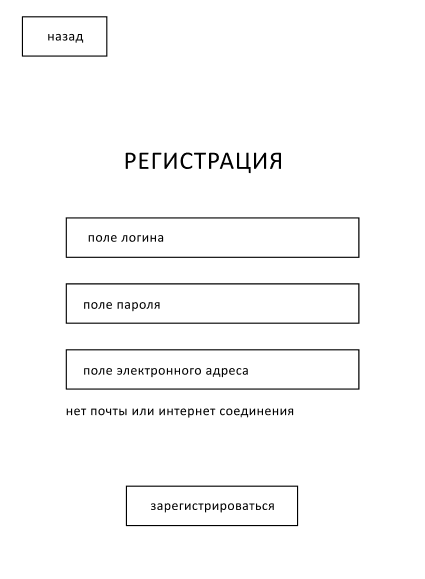


Рисунок 2.2 – Макет окна регистрации

Окно для отображения задач, заметок и недельной статистики должно предполагать меню приложения, панель основной информации пользователя, календарь, несколько панелей для вывода данных и кнопки для добавления новых данных. Макет окна задач, заметок и недельной статистики представлен на рисунке 2.3.



Рисунок 2.3 – Макет окна задач, заметок и недельной статистики

Окно для отображения направлений, целей и задач цели должно предполагать меню приложения, панель основной информации пользователя, календарь, несколько панелей для вывода данных и кнопки для добавления новых данных. Макет окна направлений, целей и задач целей представлен на рисунке 2.4.



Рисунок 2.4 – Макет окна задач, заметок и недельной статистики

Окно для отображения достижений и отображеня статистики должно предполагать меню приложения, панель основной информации пользователя, календарь, несколько панелей для вывода данных и переключатель для периода вывода. Макет окна достижений и статистики представлен на   
рисунке 2.5.



Рисунок 2.5 – Макет окна достижений и статистики

Окно для отображения расписаний и вывода паролей сервисов должно предполагать меню приложения, панель основной информации пользователя, календарь и несколько панелей для вывода данных и кнопку для добавления новых данных сервиса. Макет окна распианий, заметок и и вывода паролей сервисов представлен на рисунке 2.6.

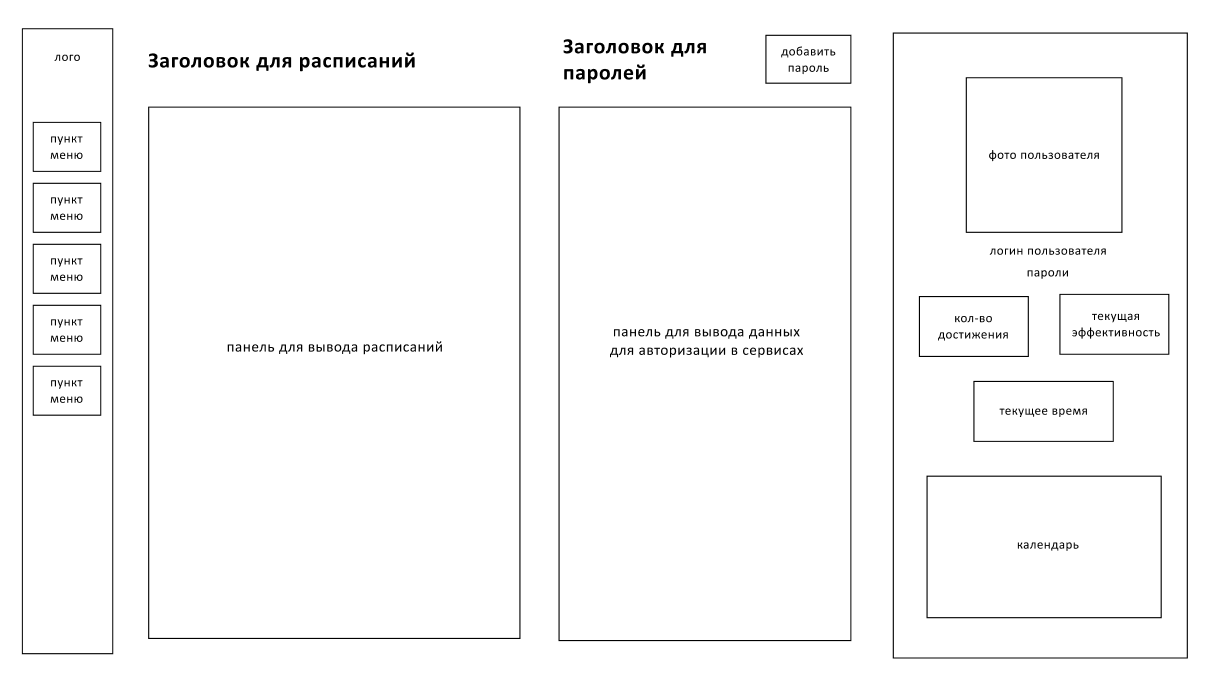


Рисунок 2.6 – Макет окна распианий, заметок и и вывода паролей сервисов

Окно для настроек приложения должно предполагать меню приложения, панель основной информации пользователя, календарь, панель для изменения данных пользователя и панель для изменения цветового оформления приложения. Макет окна настроек приложения представлен на рисунке 2.7.

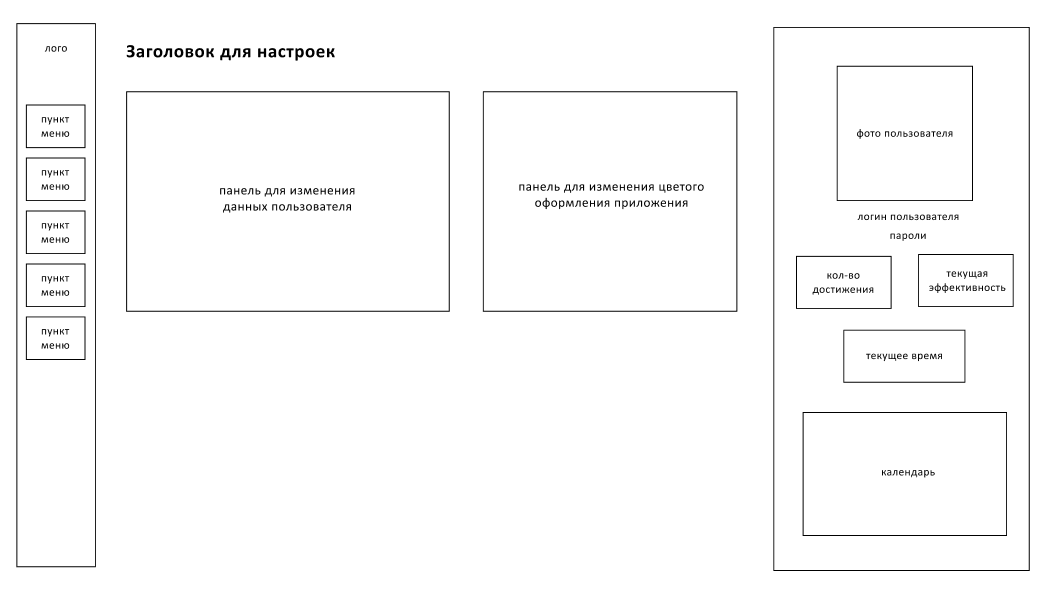


Рисунок 2.7 – Макет окна настроек приложения

Окно для создания задачи должно предполагать поле для выбора цели, поле для ввода текста задачи, описания, панель для добавления подзадач, поле для названия расписания, панель для выбора дней в расписании, поле для выбора даты, времени и несколько полей для установки начала и завершения выполнения задачи. Макет окна для создания задачи представлен на   
рисунке 2.8.

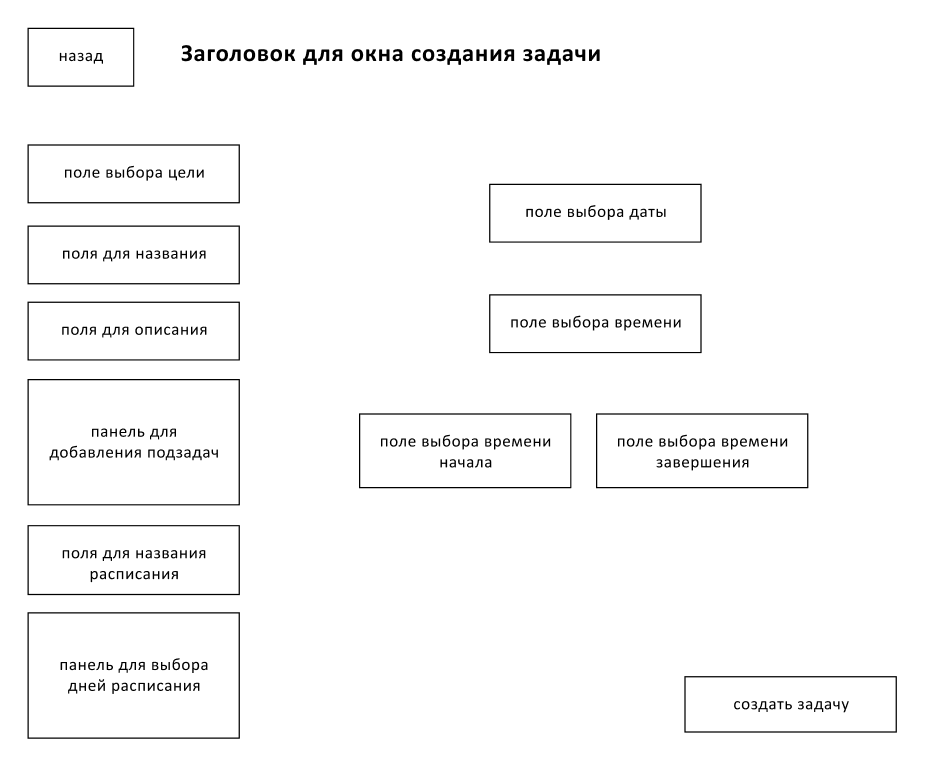


Рисунок 2.8 – Макет окна для создания задачи

Окно для создания направления должно предполагать поле для выбора категории, поле для ввода текста направления и поле для установки цветовой метки направления. Макет окна для создания направления представлен на   
рисунке 2.9.

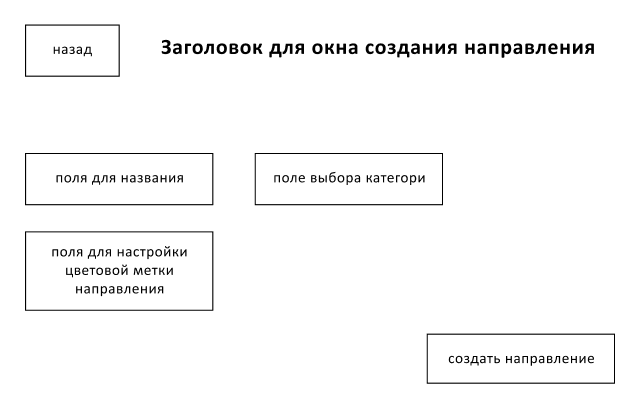


Рисунок 2.9 – Макет окна для создания направления

Окно для создания цели должно предполагать поле для выбора направления, поле для ввода текста цели. Макет окна для создания цели представлен на рисунке 2.10.

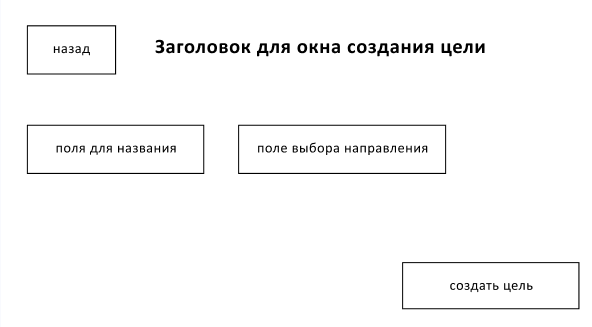


Рисунок 2.10 – Макет окна для создания цели

Окно для создания данных сервиса должно предполагать поле для ввода логина, пароли и электронного адреса или телефона. Макет окна для создания данных сервиса представлен на рисунке 2.11.

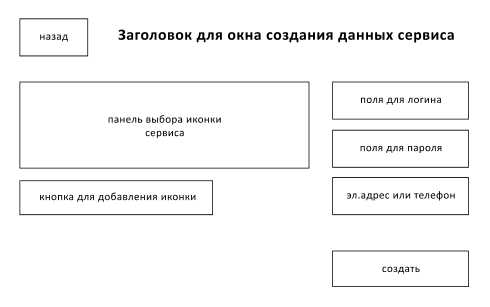


Рисунок 2.11 – Макет окна для создания данных сервиса

2.3 Требования к аппаратным, программным интерфейсам

Данное приложение должно быть использовано на операционной системе не ниже Windows 7 и иметь характеристики не ниже данной аппаратной конфигурации.

1. процессор: 2.4 ГГц;
2. оперативная память: 512 Мб;
3. свободное место на жёстком диске: 512 Мб;
4. видеоадаптер: 256 Мб;
5. монитор с разрешением: 1366х768;
6. наличие основных средств ввода-вывода (мышь, клавиатура).

2.4 Требования к пользователям продукта

Системой должны иметь возможность пользоваться любые типы пользователей. Требования к пользователям:

1. Понимание структуры файловой системы ОС Windows;
2. Понимание принципов регистрации и авторизации.

2.5 Функции продукта

Основной функционал продукта:

1. Регистрация пользователя;
2. Авторизация пользователя;
3. Вывод общей эффективности пользователя по задачам;
4. Вывод статистики пользователя по задачам за день, неделю, месяц;
5. Ведение прогресса получения достижения;
6. Хранение данных пользователя для авторизации в интернет ресурсе;
7. Изменение цветового оформления приложения;
8. Изменение данных пользователя;
9. Создание, изменение, удаление задачи;
10. Создание, изменение, удаление направления;
11. Создание, изменение, удаление цели;
12. Создание, удаление расписания;
13. Создание, удаление, заметки.

2.6 Ограничения

1. Продукт не является кроссплатформенным;
2. Продукт будет поддерживать только русский язык интерфейса;
3. Все текстовые файлы должны иметь кодировку utf-8.

# ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ

Visual Studio 2019. 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.

Среда Visual Studio доступна для Windows и Mac. Функции [Visual Studio для Mac](https://docs.microsoft.com/ru-ru/visualstudio/mac/) во многом аналогичны возможностям Visual Studio 2019 и оптимизированы для разработки кроссплатформенных и мобильных приложений. Эта статья посвящена версии Visual Studio 2019 для Windows.

Ниже перечислены некоторые популярные возможности Visual Studio, которые помогут вам повысить продуктивность разработки программного обеспечения.

Волнистые линии и обозначают ошибки или потенциальные проблемы кода прямо во время ввода. Эти визуальные подсказки позволяют устранять проблемы немедленно и не ждать, пока ошибка будет обнаружена во время сборки или запуска программы. Если навести указатель мыши на волнистую линию, на экран будут выведены дополнительные сведения об ошибке. Кроме того, в поле слева может появляться значок лампочки с быстрыми действиями по устранению ошибки.

Очистка кода. Вы можете одним нажатием кнопки отформатировать код и применить к нему исправления, предложенные [параметрами стиля кода](https://docs.microsoft.com/ru-ru/visualstudio/ide/reference/options-text-editor-csharp-formatting?view=vs-2019), [соглашениями в файле EditorConfig](https://docs.microsoft.com/ru-ru/visualstudio/ide/create-portable-custom-editor-options?view=vs-2019) и (или) [анализаторами Roslyn](https://docs.microsoft.com/ru-ru/visualstudio/code-quality/roslyn-analyzers-overview?view=vs-2019). Очистка кода помогает устранить многие проблемы в коде еще до проверки кода. (Сейчас эта возможность доступна только для кода на C#.)

Рефакторинг включает в себя такие операции, как интеллектуальное переименование переменных, извлечение одной или нескольких строк кода в новый метод, изменение порядка параметров методов и многое другое.

IntelliSense – это набор функций, отображающих сведения о коде непосредственно в редакторе и, в некоторых случаях, автоматически создающих небольшие отрывки кода. По сути, это базовая документация, встроенная в редактор, с которой вам не приходится искать информацию где-то еще. Функции IntelliSense зависят от языка. Дополнительные сведения см. в руководствах по [IntelliSense для C#](https://docs.microsoft.com/ru-ru/visualstudio/ide/visual-csharp-intellisense?view=vs-2019), [IntelliSense для Visual C++](https://docs.microsoft.com/ru-ru/visualstudio/ide/visual-cpp-intellisense?view=vs-2019), [IntelliSense для JavaScript](https://docs.microsoft.com/ru-ru/visualstudio/ide/javascript-intellisense?view=vs-2019) и [IntelliSense для Visual Basic](https://docs.microsoft.com/ru-ru/visualstudio/ide/visual-basic-specific-intellisense?view=vs-2019).

Поле поиска. Среда Visual Studio может показаться сложной, ведь там столько разных меню, параметров и свойств. Поле поиска позволяет быстро найти нужное содержимое в Visual Studio. Когда вы начнете вводить в поле то, что вы ищете, Visual Studio представит результаты, один из которых точно вам подойдет. Если вам нужно добавить функциональные возможности в Visual Studio, например, поддержку дополнительных языков программирования, поле поиска предоставляет результаты, которые открывают Visual Studio Installer для установки рабочей нагрузки или отдельного компонента.

[Live Share](https://docs.microsoft.com/ru-ru/visualstudio/liveshare/). Предоставляет возможности совместного редактирования и отладки в реальном времени независимо от типа приложения или языка программирования. Вы можете мгновенно и безопасно поделиться своим проектом и, при необходимости, сеансами отладки, экземплярами терминалов, веб-приложениями localhost, голосовыми звонками и многим другим.

[Иерархия вызовов](https://docs.microsoft.com/ru-ru/visualstudio/ide/reference/call-hierarchy?view=vs-2019). В окне показаны методы, вызывающие выбранный метод. Это может быть полезно, если вы собираетесь изменить или удалить метод, или хотите отследить ошибку.

[CodeLens](https://docs.microsoft.com/ru-ru/visualstudio/ide/find-code-changes-and-other-history-with-codelens?view=vs-2019) помогает находить ссылки на код, изменения кода, связанные ошибки, рабочие элементы, проверки кода и модульные тесты – все это, не выходя из редактора.

[Перейти к определению](https://docs.microsoft.com/ru-ru/visualstudio/ide/go-to-and-peek-definition?view=vs-2019). С функцией "Перейти к определению" вы напрямую переходите туда, где определена функция или тип.

[Показать определение](https://docs.microsoft.com/ru-ru/visualstudio/ide/how-to-view-and-edit-code-by-using-peek-definition-alt-plus-f12?view=vs-2019). В окне показано определение метода или типа, при этом не нужно открывать отдельный файл [4].

DB Browser for SQLite. Представляет собой визуальный инструмент с открытым исходным кодом для создания, разработка и редактирование файлов баз данных, совместимых с SQLite. Программа была ранее известна как "SQLite Database Browser" и "Database Browser for SQLite". Полностью на русском языке.

Средства управления позволяют:

1. Создавать файлы базы данных;
2. Создавать, определять, изменять и удалять таблицы;
3. Создавать, определять и удалять индексы;
4. Просматривать, редактировать, добавлять и удалять записи;
5. Искать записи;
6. Импортировать и экспортировать записи как текст;
7. Импортировать и экспортировать таблицы из/в файлы CSV;
8. Импортировать и экспортировать базы данных как SQL дамп;
9. Выполнять SQL-запросы и проверять результаты;
10. Изучать журнал всех SQL команд.

Инструмент для пользователей и разработчиков, которым нужно создавать базы данных, искать и редактировать данные в них. Он использует привычный табличный интерфейс, поэтому не нужно изучать сложные SQL-команды [5].

Adobe Photoshop. Предоставляет работу с типографикой, цветом, вектором и фильтрами позволит создать нечто абсолютно новое: эффектные коллажи, имиджевые и рекламные материалы. Photoshop позволяет легко отредактировать цвет, контраст, освещение, баланс цветов, убрать лишние объекты, применить фильтры и сделать многое другое. Автоматизация таких процессов, как выделение сложных форм, удаление объектов, ретуширование фотографий и цветовая коррекция, позволит быстрее и проще реализовать задуманное [6].

Figma. Figma — кроссплатформенный онлайн-сервис для дизайнеров интерфейсов и веб-разработчиков.

Figma позволяет разрабатывать интерфейсы в онлайн-приложении. У Figma две ключевые особенности: доступ к макету прямо из окна браузера и возможность совместной работы над документами. До появления Figma нескольким дизайнерам было сложно работать над одним проектом, а также передавать макеты разработчикам. Вспомните, как Photoshop отказывался корректно открывать макет, пока вы не установите нужные шрифты. Или как возникали сложности с парной работой — коллега вносил изменения в свою копию и забывал сказать вам об этом или по ошибке присылал не ту версию макета.

Команда Figma учла все эти проблемы и создала продукт, который из коробки включает в себя одновременную работу нескольких дизайнеров, версионность документа и много других возможностей. Они упрощают жизнь как дизайнеру, так и разработчику.

Для того, чтобы сотрудничать с другими дизайнерами, в Figma предусмотрена опция создания команды. Облачный сервер хранения файлов

Все файлы хранятся в облаке Figma. Не надо вспоминать, где лежат ваши макеты, скачивать и после того, как внесли изменения, заливать обратно. Изменения в Figma тоже сохраняются автоматически.

Это элементы пользовательского интерфейса, которым можно задавать общие стили и затем быстро менять их во всем макете сразу.

Например, после того как вы закончили разработку 50 макетов в Photoshop, заказчик попросил вас поменять пункт меню в шапке и прислать все макеты на утверждение. Версии автоматически сохраняются, если в течение 30 минут в файле не было никаких изменений.

Вы можете видеть, когда файл был отредактирован и кто сделал изменения. Также можно вручную добавлять новые версии при необходимости. Чтобы не запутаться в дальнейшем, осмысленно называйте версии и желательно добавляйте описание. Например, детали проекта: итерацию, этап, какие изменения внесли. У Figma низкий порог вхождения и мощный потенциал при кажущейся простоте [7].

# РАЗРАБОТКА БАЗЫ ДАННЫХ ПРИЛОЖЕНИЯ

4.1 Концептуальное проектирование базы данных

При концептуальном проектировании составил список всех основных элементов, которые могут лежат в границах моделируемой системы.

1. Логин пользователя;
2. Пароль пользователя;
3. Электронный адрес пользователя;
4. Фото пользователя;
5. Текст задачи;
6. Дата выполнения задачи;
7. Время выполнения задачи;
8. Статус задачи;
9. Текст подзадачи;
10. Статус подзадачи;
11. Название цели;
12. Статус цели;
13. Название расписания;
14. Название дня недели;
15. Название направления;
16. Цветовая метка направления;
17. Название категории;
18. Название достижения;
19. Описание достижения;
20. Очки для достижения;
21. Статус достижения;
22. Текст заметки;
23. Логин для сервиса;
24. Пароль для сервиса;
25. Электронный адрес или телефон для сервиса.

Сформировал список объектов на основе установленных элементов моделируемой системы (табл. 4.1).

Таблица 4.1 - Список объектов моделируемой системы

|  |  |
| --- | --- |
| **Наименование объекта** | **Краткое описание** |
| Пользователь | Основная информация о пользователе |
| Направление | Информация о направлении |
| Цель | Информация о целе пользователя |
| Задача | Основная информация о задаче |
| Подзадача | Информация о подзадаче для какой-лио задачи |
| Расписание | Информация о расписании пользователя |
| День недели | Служебный объект для формирования расписания пользователя |
| Категория | Основная информация о категории для какого-либо направления |
| Достижение | Основная информация о достижении какой-либо категории |
| Данные сервиса | Информация о сервисе, хранит данные для автиризации в сервисе |
| Заметка | Информация о заметке |

Сформировал список атрибутов для каждого объекта.

Пользователь (табл. 4.2):

Таблица 4.2 - Атрибуты для объекта «Пользователь»

|  |  |
| --- | --- |
| **Наименование атрибута** | **Краткое описание** |
| Логин | Логин пользователя для авторизации |
| Пароль | Пароль пользователя для авторизации |
| Электронный адрес | Электронный адрес пользователя |
| Фото | Фото пользователя |

Направление (табл. 4.3):

Таблица 4.3 - Атрибуты для объекта «Направление»

|  |  |
| --- | --- |
| **Наименование атрибута** | **Краткое описание** |
| Название | Название направления |
| Цветовая метка | HEX-код для цветовой идентификации направления |

Цель (табл. 4.4):

Таблица 4.4 - Атрибуты для объекта «Цель»

|  |  |
| --- | --- |
| **Наименование атрибута** | **Краткое описание** |
| Название | Название цели |
| Статус | Логический атрибут указывающий на статус завершения |

Задача (табл. 4.5):

Таблица 4.5 - Атрибуты для объекта «Задача»

|  |  |
| --- | --- |
| **Наименование атрибута** | **Краткое описание** |
| Текст | Текст задачи |
| Описание | Описание задачи |
| Дата | Дата выполнения задачи |
| Время | Время выполнения задачи |
| Статус | Логический атрибут указывающий на статус завершения |

Подзадача (табл. 4.6):

Таблица 4.6 - Атрибуты для объекта «Подзадача»

|  |  |
| --- | --- |
| **Наименование атрибута** | **Краткое описание** |
| Текст | Текст подзадачи |
| Статус | Логический атрибут указывающий на статус завершения |

Расписание (табл. 4.7):

Таблица 4.7 - Атрибуты для объекта «Подзадача»

|  |  |
| --- | --- |
| **Наименование атрибута** | **Краткое описание** |
| Название | Название расписания |

День недели (табл. 4.8):

Таблица 4.8 - Атрибуты для объекта «День недели»

|  |  |
| --- | --- |
| **Наименование атрибута** | **Краткое описание** |
| Название | Название дня недели |

Категория (табл. 4.9):

Таблица 4.9 - Атрибуты для объекта «Категория»

|  |  |
| --- | --- |
| **Наименование атрибута** | **Краткое описание** |
| Название | Название категории |

Достижение (табл. 4.10):

Таблица 4.10 - Атрибуты для объекта «Достижение»

|  |  |
| --- | --- |
| **Наименование атрибута** | **Краткое описание** |
| Название | Название достижения |
| Описание | Описание достижения |
| Финальные очки | Финальные очки достижения |
| Статус | Логический атрибут указывающий на статус завершения |

Данные сервиса (табл. 4.11):

Таблица 4.11 - Атрибуты для объекта «Данные сервиса»

|  |  |
| --- | --- |
| **Наименование атрибута** | **Краткое описание** |
| Логин | Логин для сервиса |
| Пароль | Пароль для сервиса |
| Электронный адрес или телефон | Электронный адрес или телефон для сервиса |

Заметка (табл. 4.12):

Таблица 4.12 - Атрибуты для объекта «Заметка»

|  |  |
| --- | --- |
| **Наименование атрибута** | **Краткое описание** |
| Текст | Текст заметки |

После установки основных атрибутов объектов установил между ними связь (табл. 4.13).

Таблица 4.13 - Связи между объектами базы данных

|  |  |  |
| --- | --- | --- |
| **Наименование связи** | **Объекты, участвующие в связи** | **Краткое описание** |
| 1:M | Пользователь – Заметка | Один пользователь может иметь несколько заметок |
| 1:M | Пользователь – Данные сервиса | Один пользователь может иметь несколько сервисов с данными для авторизации в них |
| M:N | Пользователь – Направление | Несколько пользователей могут иметь несколько направлений |
| 1:M | Категория – Направление | Одна категория соотвествует нескольким направлениям |
| M:N | Категория – Достижение | Несколько категорий могут иметь несколько достижений |
| 1:M | Направление – Цель | Одно направление может иметь несколько целей |
| 1:M | Цель – Задача | Одна цель может иметь несколько задач |
| 1:M | Задача – Подзадача | Одна задача может иметь несколько подзадач |
| M:N | Задача – Расписание | Несколько задач могут быть включены в несколько расписаний |
| M:N | Расписание – День недели | Несколько расписаний могут иметь несколько дней для повторения задач |

Концептуальная схема базы данных представлена на рисунке 4.1.

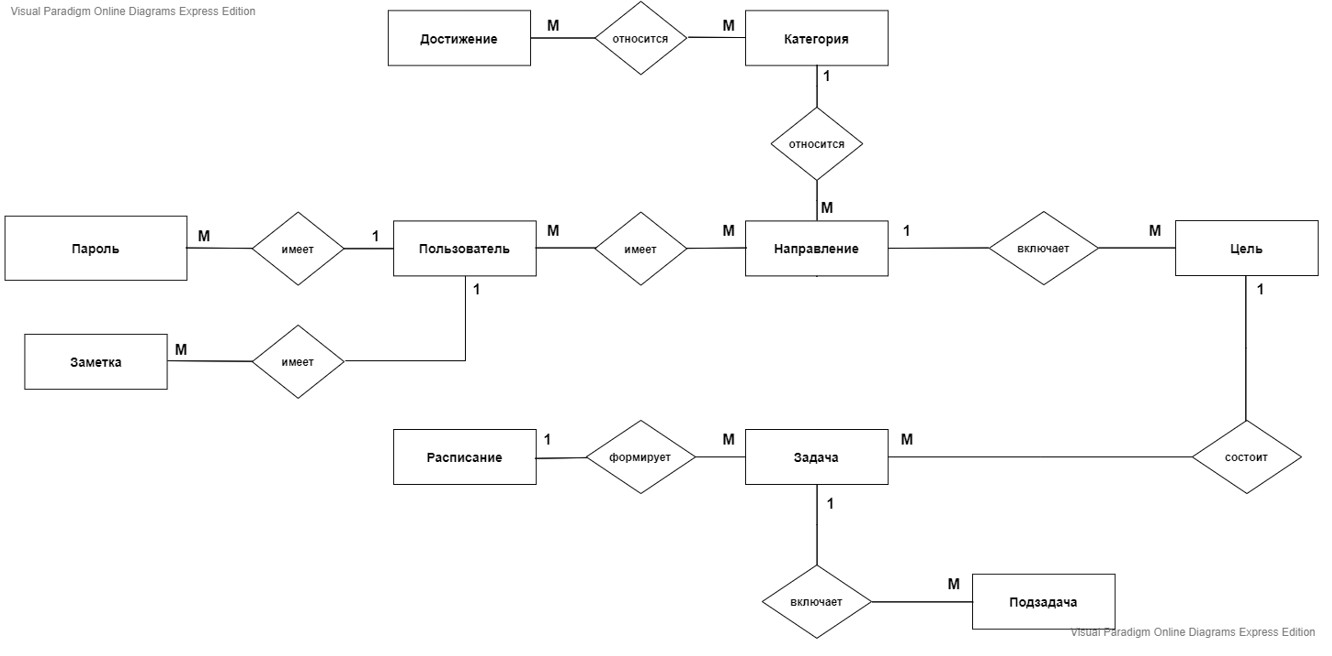


Рисунок 4.1 – Концептуальная схема базы данных

4.2 Логическое проектирование базы данных

При логическом проектировании были установлены ключи для каждого объекта, что позволяет правильно нормализовать информацию в базе данных. Для объекта «Пользователь» был установлен ключ – Номер пользователя, для объекта «Направление» был установлен ключ – Номер направления, как первичный, что позволит связать объекты «Пользователь» и «Направление», а также Номер категории, как внешний, что позволит связать объекты «Категория» и «Направление». Для объекта «Цель» был установлен ключ – Номер цели, как внешний, что позволит связать объекты «Цель» и «Направление», а также Номер цели как первичный ключ. Для объекта «Задача» был установлен первичный ключ – Номер задачи и Номер цели, как внешний ключ для связи объектов «Задача» и «Цель». Для объекта «Подзадача» был установлен первичный ключ – Номер подзадачи и Номер задачи, как внешний ключ, для связи объектов «Задача» и «Подзадача». Для объекта «Категория» был установлен первичный ключ – Номер категории. Для объекта «Достижение» был установлен первичный ключ – Номер достижения. Для объектов «Данные сервиса» и «Заметка» были установлены первичные ключи – Номер сервиса и Номер заметки, соответсвенно и внешние ключи – Номер пользователя для связи объектов «Пользователь» и «Данные сервиса», а также «Пользователь» и «Заметка».

На основе концептуальной схемы создал логическую схему, определив для сущностей атрибуты и ключевые поля, по которым эти сущности будут связаны между собой, а также более конкретно установил тип связей между сущностями. Были установлены идентифицирующие и не идентифицирующие связи между сущностями.

В результате разработки сформировал данную логическую схему (рис.4.2).

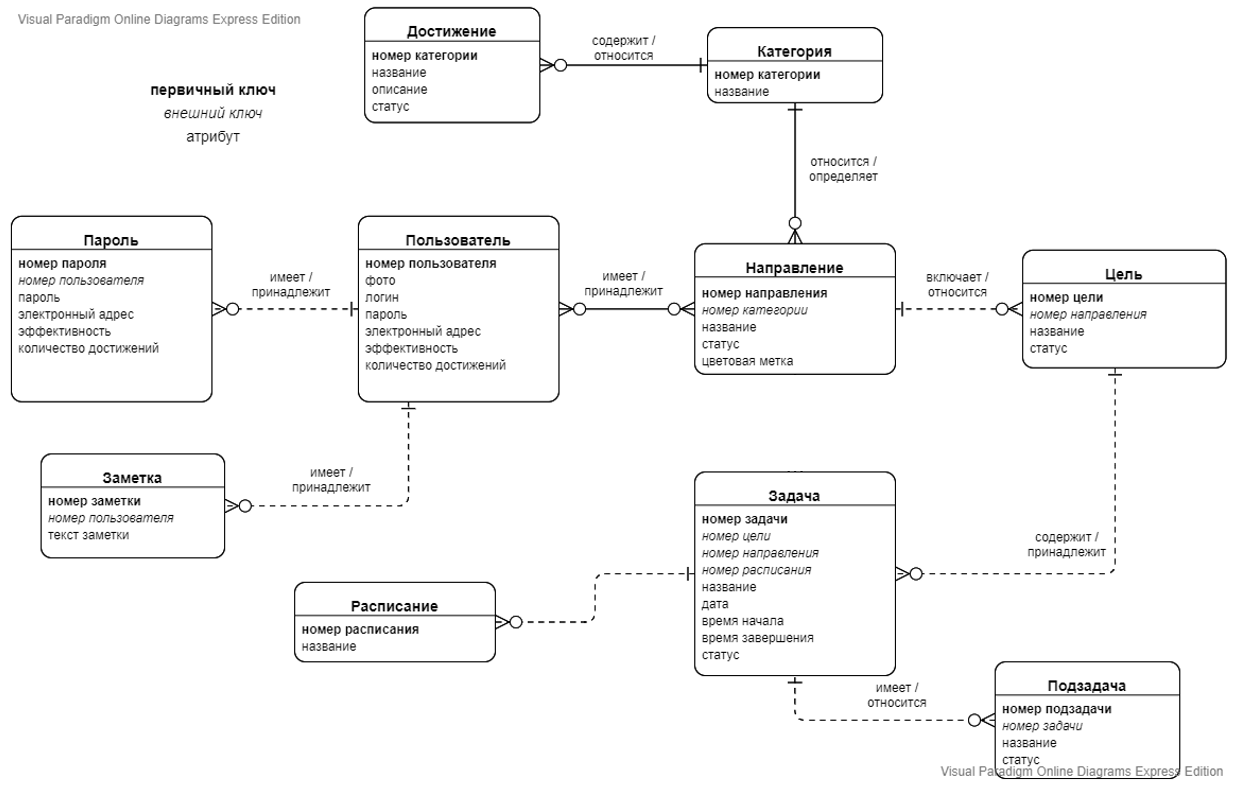


Рисунок 4.2 – Логическая схема базы данных

4.3 Физическое проектирование базы данных

Для разрабатываемой базы данных установил даталогическую модель, которая представляет собой логические взаимосвязи между элементами данных безотносительно их содержания и физической организации. Были установлены названия полей таблиц, которые будут созданы в СУБД DB Browser for SQLite.

Пользователь (табл. 4.14):

Таблица 4.14 – Поля и их свойства таблицы «Пользователь»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код пользователя | id\_user | числовой | Первичный ключ, уникальный |
| Фото | photo | текстовый |  |
| Логин | login | текстовый |  |
| Пароль | password | текстовый |  |
| Электронный адрес | email | текстовый |  |
| Регистрационный код | reg\_code | текстовый |  |

Направление (табл. 4.15):

Таблица 4.15 – Поля и их свойства таблицы «Направление»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код направления | id\_direct | числовой | Первичный ключ, уникальный |
| Код категории | id\_categ | числовой | Внешний ключ |
| Название | name | текстовый |  |
| Цветовая метка | color\_mark | текстовый |  |

Пользователь – Направление (табл. 4.16):

Таблица 4.16 – Поля и их свойства таблицы «Пользователь – Направление»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код пользователя | id\_direct | числовой | Внешний ключ |
| Код направления | id\_user | числовой | Внешний ключ |

Цель (табл. 4.17):

Таблица 4.17 – Поля и их свойства таблицы «Цель»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код цели | id\_target | числовой | Первичный ключ, уникальный |
| Код направления | id\_direct | числовой | Внешний ключ |
| Название | name | текстовый |  |
| Завершенность | compliting | числовой |  |

Задача (табл. 4.18):

Таблица 4.18 – Поля и их свойства таблицы «Задача»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код задачи | id\_task | числовой | Первичный ключ, уникальный |
| Код цели | id\_target | числовой | Внешний ключ |
| Текст задачи | text | текстовый |  |
| Описание | descr | текстовый |  |
| Дата | date | текстовый |  |
| Время | time | текстовый |  |
| Время окончания | time\_finish | текстовый |  |
| Провалено | failed | числовой |  |
| Статус | status | числовой |  |

Подзадача (табл. 4.19):

Таблица 4.19 – Поля и их свойства таблицы «Подзадача»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код подзадачи | id\_target | числовой | Первичный ключ, уникальный |
| Код задачи | id\_task | числовой | Внешний ключ |
| Текст подзадачи | text | текстовый |  |
| Статус | status | числовой |  |

Расписание (табл. 4.20):

Таблица 4.20 – Поля и их свойства таблицы «Расписание»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код расписания | id\_sched | числовой | Первичный ключ, уникальный |
| Название | name | текстовый |  |

День недели (табл. 4.21):

Таблица 4.21 – Поля и их свойства таблицы «День недели»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код дня | id\_day | числовой | Первичный ключ, уникальный |
| Название | name | текстовый |  |

Расписание – Задача (табл. 4.22):

Таблица 4.22 – Поля и их свойства таблицы «Расписание – Задача»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код расписания | id\_sched | числовой | Внешний ключ |
| Код задачи | id\_task | числовой | Внешний ключ |
| Код дня | id\_day | числовой | Внешний ключ |

Категория (табл. 4.23):

Таблица 4.23 – Поля и их свойства таблицы «Категория»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код категории | id\_categ | числовой | Первичный ключ, уникальный |
| Название | name | текстовый |  |

Достижение (табл. 4.24):

Таблица 4.24 – Поля и их свойства таблицы «Достижение»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код достижения | id\_achiev | числовой | Первичный ключ, уникальный |
| Название | name | текстовый |  |
| Описание | descr | текстовый |  |
| Текущие очки | current\_score | числовой |  |
| Конечные очки | final\_score | числовой |  |
| Статус | status | числовой |  |

Достижение – Категория (табл. 4.25):

Таблица 4.25 – Поля и их свойства таблицы «Достижение – Категория»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код достижения | id\_achiev | числовой | Внешний ключ |
| Код категории | id\_categ | числовой | Внешний ключ |

Данные сервиса (табл. 4.26):

Таблица 4.26 – Поля и их свойства таблицы «Данные сервиса»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код сервиса | id\_service | числовой |  |
| Код пользователя | id\_user | числовой | Внешний ключ |
| Логин сервиса | login | текстовый |  |
| Пароль сервиса | password | текстовый |  |
| Электронный адрес или телефон сервиса | em\_ph | текстовый |  |

Заметка (табл. 4.26):

Таблица 4.26 – Поля и их свойства таблицы «Заметка»

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Идентификатор** | **Тип** | **Ограничения** |
| Код заметки | id\_note | числовой | Первичный ключ, уникальный |
| Код пользователя | id\_user | числовой | Внешний ключ |
| Текст заметки | text | числовой | Внешний ключ |

Для этой предметной области была разработана реляционная (физическая) схема базы данных (рис. 4.3). Запросы на создание базы данных представлены в приложении Б.

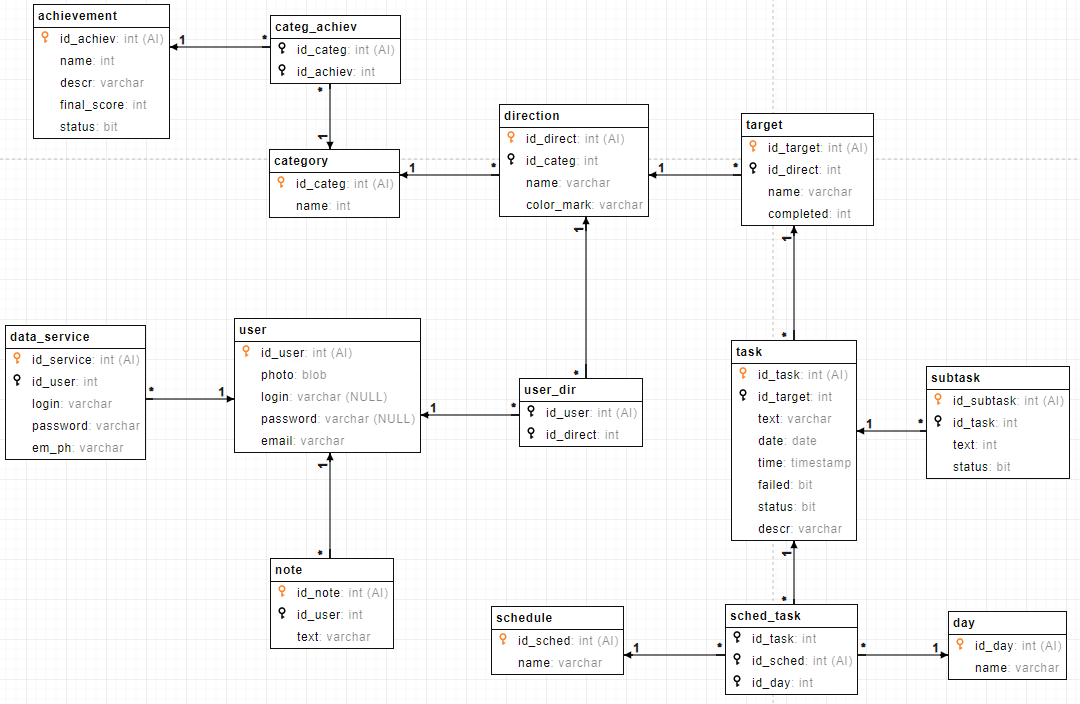


Рисунок 4.3 – Физическая схема базы данных

# РАЗРАБОТКА СИСТЕМЫ ПЛАНИРОВАНИЯ ЗАДАЧ И ЦЕЛЕЙ

5.1 Входные и выходные данные приложения

Входными данными являются задачи, цели и направления пользователя, а также достижения, которые храняться в базе данных. Работа с достижениями, направлениями, целями, задачами и т.д. происходит через приложение в соответсвующих окнах при выборе пункта меню. К данным приложения также относится файл с расширением .ini, который хранит данные пользователей для авторизации и служебные данные для корректной работы приложения.

Также входными данными приложения являются иконки для достижений и сервисов, а также фото пользователей, которые храняться в отдельных папках в корневой папке приложения. Содержимое папки с иконками для достижений представлено на рисунке 5.1. Содержимое папки с иконками для сервисов представлено на рисунке 5.2. Содержимое папки с фото пользователей представлено на рисунке 5.3.

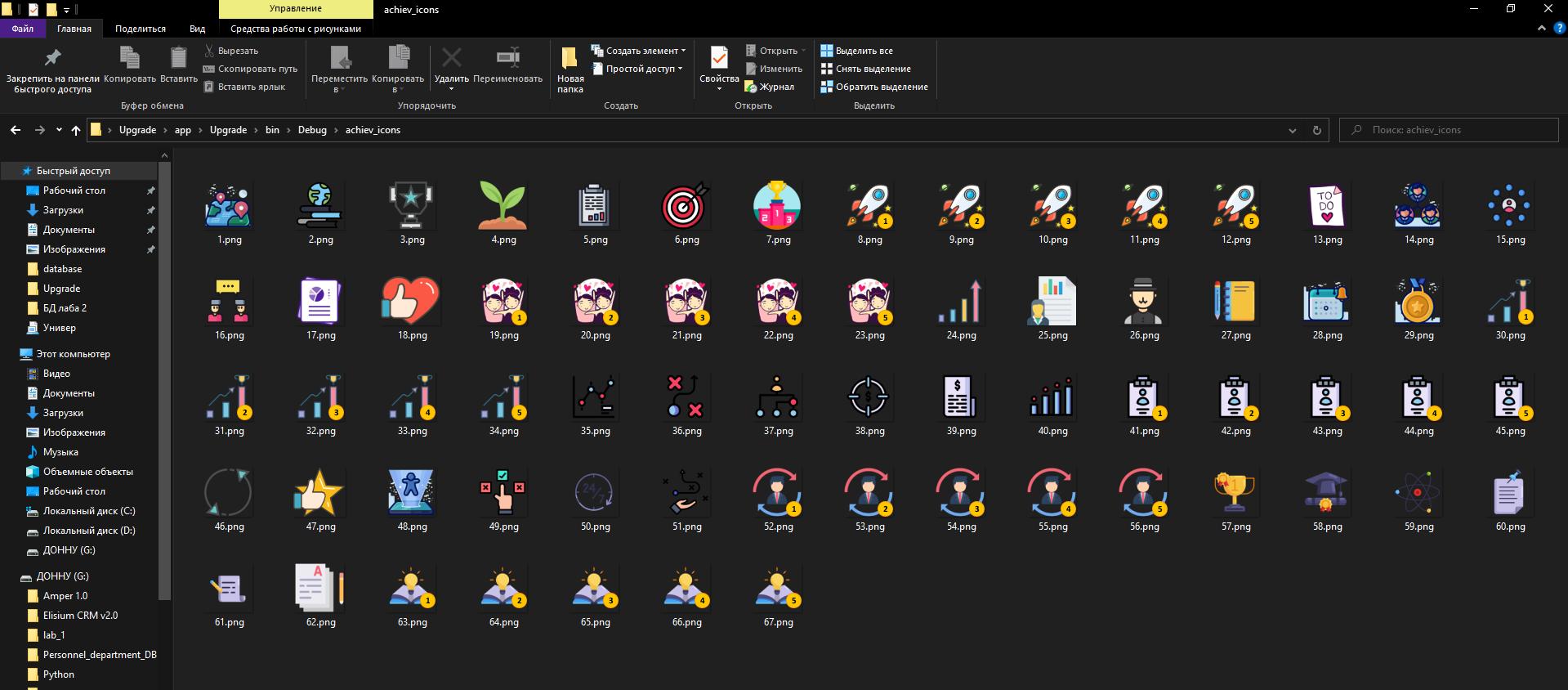


Рисунок 5.1 – Содержимое папки с иконками для достижений

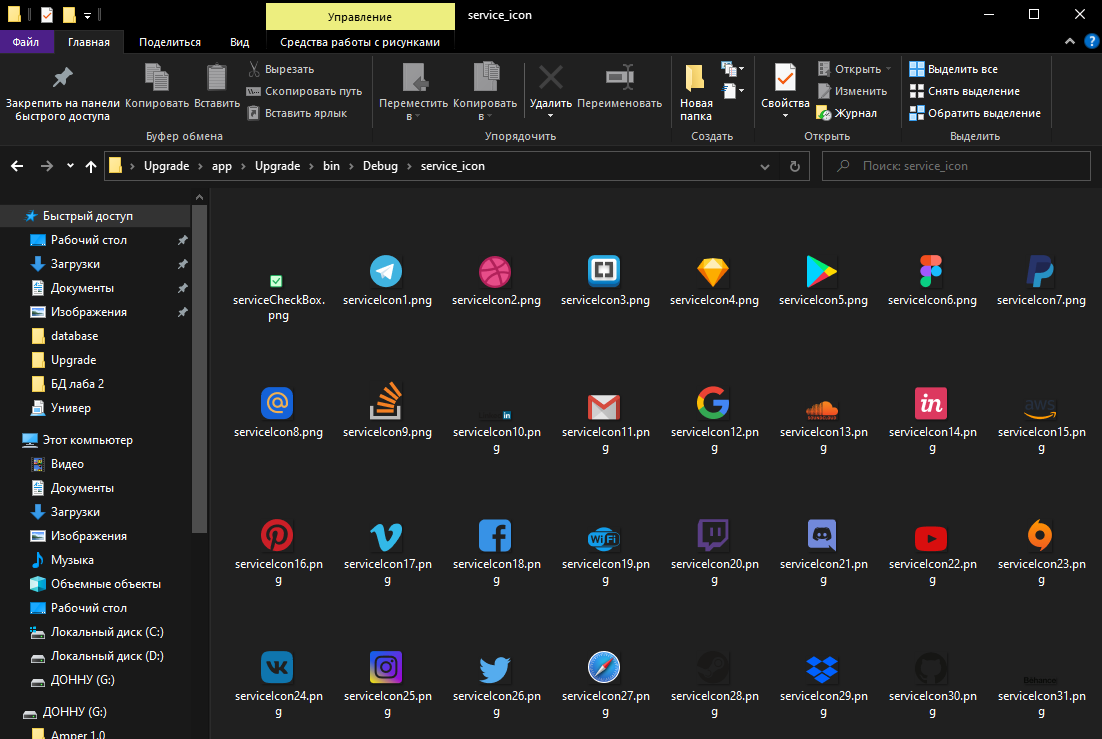


Рисунок 5.2 – Содержимое папки с иконками для сервисов

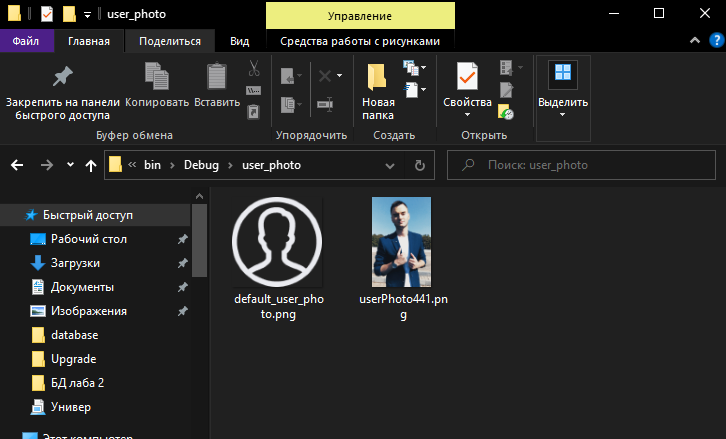


Рисунок 5.3 – Содержимое папки с фото пользователей

Выходными данными являются добавленные пользователями задачи, направления, цели, расписания, данные сервисов и заметки, которые сохраняются в базе данных. Также выходными данными являются иконки сервисов, измененный конфигурационный файл settings.ini (рис.5.4) и фото пользователей, добавленные в ходе пользования приложением.

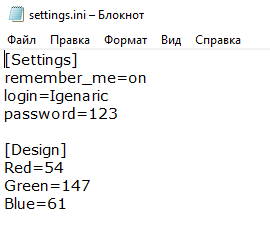


Рисунок 5.4 – Конфигурационный файл settings.ini

5.2 Проектирование структуры приложения

Данное программное приложение состоит из нескольких форм. Структурно приложение состоит из следующих классов – форм:

1. Класс-форма авторизации и регистрации (Reg\_AuthForm.cs) – содержит название приложения и кнопку выхода изприложения;
2. Класс-форма главного окна приложения (MainWorkingForm.cs) – содержит статистическую информацию по работе студента и его личные данные;
3. Класс-форма добавления новой задачи (AddTaskForm.cs) – содержит номер лабораторной работы её цель и тему;
4. Класс-форма добавления нового направления (AddDirectionForm.cs) – содержит панель выбора элементов, панель значений элементов, панель открытия тенкста лабораторной работы и алгоритм её выполнения;
5. Класс-форма добавления новой цели (AddTargetForm.cs) – предполагает одно поле для ввода текста вывода о проделанной работе;
6. Класс-форма добавления новой заметки (AddNoteForm.cs) – предполагает вывод текста вопросов и полей для ввода ответов на них.
7. Класс-форма добавления нового сервиса (AddDataServiceForm.cs) – предполагает вывод текста вопросов и полей для ввода ответов на них.

Структура форм представлена на рисунке 5.5.

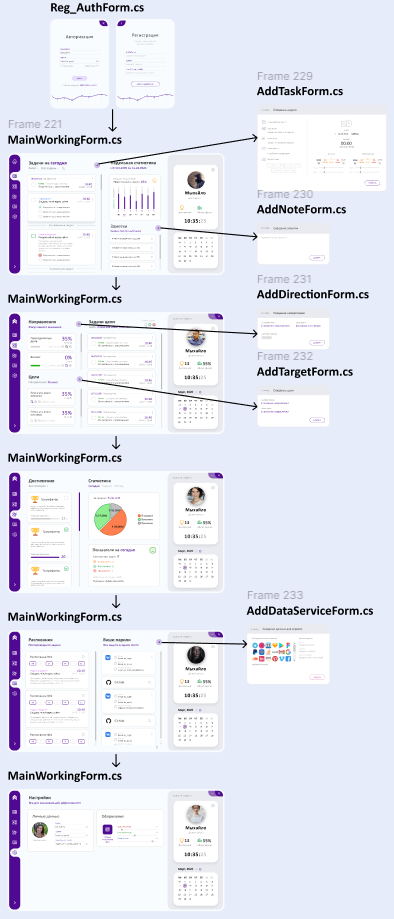


Рисунок 5.5 – Структура форм приложения

В результате проведенного анализа были сформированы классы для работы работы с базой данных и приложения в целом.

Классы для создания информационных блоков (.cs)

|  |  |  |
| --- | --- | --- |
| **Название класса** | | **Описание** |
| Block | | Содержит общие переменные и объекты стандартных классов для создания информационного блока |
|  | AchievBlock | Содержит переменные, методы и объекты стандартных классов для создания блока достижения |
| NoteBlock | Содержит переменные, методы и объекты стандартных классов для создания блока заметки |
| DataServiceBlock | Содержит переменные, методы и объекты стандартных классов для создания блока данных сервиса |
| TaskBlock | Содержит переменные, методы и объекты стандартных классов для создания блока задачи |
| SubTaskBlock | Содержит переменные, методы и объекты стандартных классов для создания блока подзадачи |
| DirectionBlock | Содержит переменные, методы и объекты стандартных классов для создания блока направления |
| ScheduleBlock | Содержит переменные, методы и объекты стандартных классов для создания блока расписания |
| TargetBlock | Содержит переменные, методы и объекты стандартных классов для создания блока цели |

Классы для реализации UI-компонентов (.cs)

|  |  |  |
| --- | --- | --- |
| **Название класса** | | **Описание** |
| UIComponent | | Содержит общие переменные и объекты стандартных классов для создания визуального компонента |
|  | UITrackBar | Содержит переменные, методы и объекты стандартных классов для создания визуального компонента TrackBar |
| UIComboBox | Содержит переменные, методы и объекты стандартных классов для создания визуального компонента ComboBox |
| Chart | | Содержит общие переменные и объекты стандартных классов для создания графика |
|  | StatisticChart | Содержит переменные, методы и объекты стандартных классов для создания диаграммы с областями |
| PieChart | Содержит переменные, методы и объекты стандартных классов для создания круговой диаграммы |
| Service | | Содержит переменные, методы и объекты стандартных классов для создания визуального компонента иконки сервиса |
| Subtask | | Содержит переменные, методы и объекты стандартных классов для создания визуального компонента подзадачи |
| Filter | | Содержит переменные, методы и объекты стандартных классов для создания блока фильтрации задач |
| WeeklyStatistic | | Содержит переменные, методы и объекты стандартных классов для создания блока недельной статистики |

Классы глобальных объектов классов, переменных, методов и перечислений(.cs)

|  |  |
| --- | --- |
| **Название класса** | **Описание** |
| Design | Содержит глабальные переменные для реализации визуального оформления приложения |
| GlobalComponent | Содержит глабальные компоненты для работы приложения |
| GlobalData | Содержит глабальные переменные, методы, структуры и объекты классов форм |
| WindowManager | Содержит глабальные переменные, методы и списки информационных блоков для отображения анных из базы данных |

Классы для реализации дополнительных функций приложения (.cs)

|  |  |
| --- | --- |
| **Название класса** | **Описание** |
| Enums | Содержит глобальные перечисления для работы приложения |
| INIManager | Содержит переменные и глобальные методы для работы с .ini файлами |
| MailSenler | Содержит глабальные метод для отправки регистрационного кода электронную почту пользователя |

Классы для работы с базой данных (.cs)

|  |  |
| --- | --- |
| **Название класса** | **Описание** |
| DBService | Содержит глобальные методы регистрации и авторизации, а также метод подключения к базе данных |
| User | Содержит глобальные методы и переменные для хранения данных авторизованного пользователя |
| ServiceData | Содержит глабальные переменные для реализации SQL запросов к базе данных |

Диаграмма классов представлена на рисунке 5.6.

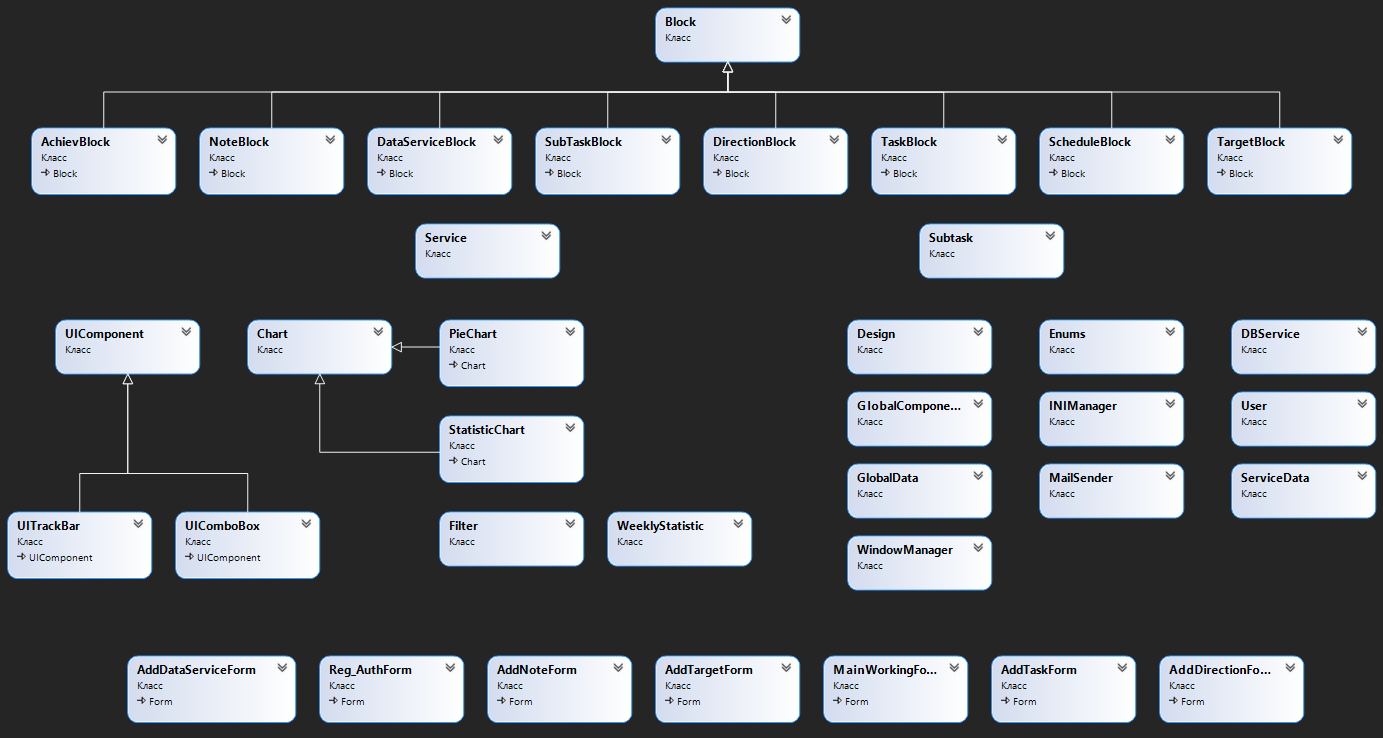


Рисунок 5.6 – Диаграмма классов

5.3 Описание алгоритма работы объектов и SQL-запросов системы

Метод авторизации пользователя класса DataService.

static public bool Authorization(string login, string password)

{

SQL-запрос дл получения всех данных пользователя

ServiceData.commandText = @"SELECT \* FROM user WHERE login = @login AND password = @password";

ServiceData.command = new SQLiteCommand(ServiceData.commandText, ServiceData.connect);

ServiceData.command.Parameters.AddWithValue("@login", login);

ServiceData.command.Parameters.AddWithValue("@password", GetMD5Hash(password));

ServiceData.reader = ServiceData.command.ExecuteReader();

if (ServiceData.reader.HasRows)

{

Если хотя бы одна запись найдена, то заполняем переменные класcа User данными из базы данных и открываем главную рабочую форму

ServiceData.reader.ReadAsync();

User.userId=ServiceData.reader.GetInt32(ServiceData.reader.GetOrdinal("id\_user"));

User.userLogin=ServiceData.reader.GetString(ServiceData.reader.GetOrdinal("login"));

User.userPassword=ServiceData.reader.GetString(ServiceData.reader.GetOrdinal("password"));

User.userEmail=ServiceData.reader.GetString(ServiceData.reader.GetOrdinal("email"));

User.SetPhoto(ServiceData.reader.GetValue(ServiceData.reader.GetOrdinal("photo")).ToString());

GlobalData.mainWorkingForm.Show();

return true;

}

else

{

Если данные пользователя не найдены, то выводим сообщение о том, что пользователю не удалось авторизоваться

MessageBox.Show(

"Не удалось авторизоваться...\n\nПроверьте правильность введенных данных или восстановите пароль.",

"Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

}

Полный листинг кода приведен в приложении Б.

# ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

6.1 Аппаратные и программные средства создания и эксплуатации программного обеспечения

Аппаратные требования для работы данного программного обеспечения:

1. процессор с тактовой частотой не менее 2.0 ГГц;
2. оперативная память 256 Мб и более;
3. видеокарта с объёмом памяти 128 Мб и выше;
4. монитор 1366х768 или с более высоким разрешением.

Программные требования к программному обеспечению, установленные на компьютере:

* + 1. операционная система —Windows 7 или выше.

6.2 Руководство пользователя

Для работы начала работы в данном программном обеспечении необходимо:

Исталировать данное программное обеспечение*,* открыв файл SetupElectronVPL.exe и следовать инструкциям установщика. При открытии вам необходимо выбрать желаемый язык установки, окно выбора языка представлено на рисунке 5.1.

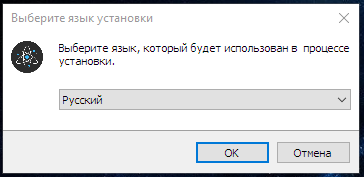


Рисунок 6.1 – Окно выбора языка установки

После того, как вы выбрали язык установки, откроется окно для выбора директории, куда будет установлено приложение. Окно представлено на рисунке 5.2.

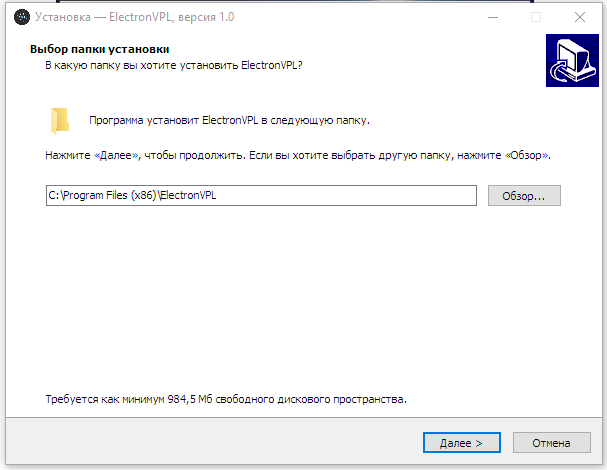


Рисунок 6.2 – Окно выбора директории для установки приложения

После выбора директории вам откроется окно для выбора создания иконки приложения на рабочем столе, окно представлено на рисунке 5.3.

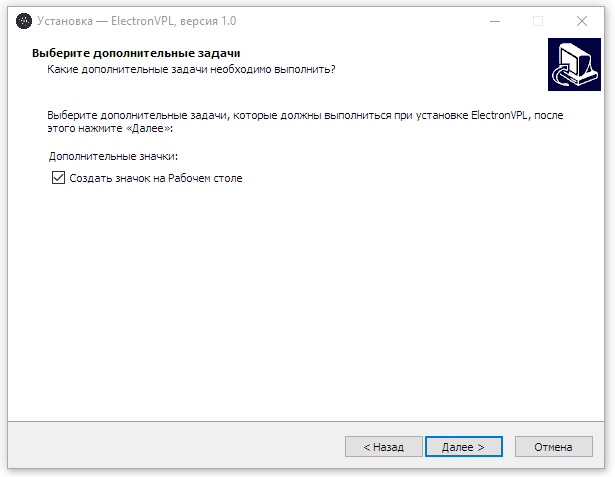


Рисунок 6.3 – Окно выбора создания иконки на рабочем столе

После выбора сохранения иконки вам откроется окно процесса установки приложения, окно представлено на рисунке 5.4.

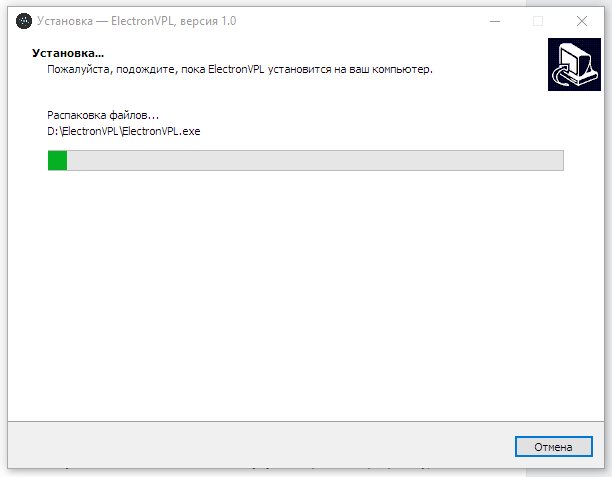


Рисунок 6.4 – Окно процесса установки приложения

После завершения установки вы сможете работать в приложении открыв его с места установки, которое вы выбрали в процессе установки.

6.3 Описание контрольных примеров

Для запуска данного приложения необходимо запустить его, дважды щелкнув по иконке приложения.

При запуске программы откроется форма приветствия  
(рисунок А.1). Чтобы продолжить, необходимо сделать мышкой скролл вниз, после чего откроется форма авторизации учащегося, на которой необходимо будет ввести данные для входа в аккаунт (рисунок А.2). Нажав на кнопку «Зарегистрироваться» откроется форма регистрации учащегося (рисунок А.3), на которой необходимо будет ввести данные учащегося.

При нажатии на форме авторизации на кнопку «Войти» откроется форма профиля учащегося (рисунок А.4), на которой будет отображаться его личная информация и количество выполненных лабораторных работ. Для начала выполнения работы необходимо нажать на кнопку «Начать работу», которая находится на форме профиля учащегося. После этого откроется форма выбора лабораторной работы (рисунок А.5), где указывается ее номер, тема и цель. Для начала выполнения лабораторной работы необходимо нажать кнопку «Начать работу».

После этого откроется главная форма приложения. При её открытии появится диалоговое окно для выбора места сохранения отчета (рисунок А.6). На главной форме будет происходить выполнение лабораторной работы. На ней отображаются кнопки для открытия панелей элементов и панели значений элементов, а также кнопка для открытия файла текста лабораторной работы.

На форме предоставляется возможность открытия панели элементов. Для этого необходимо кликнуть на кнопке слева, отображающей электрический элемент (рисунок А.7). Добавление элементов происходит путем клика по иконке элемента на панеле элементов и установки места для добавления (рисунок А.8), перемещения, а также их подключения. Чтобы подключить элементы, необходимо добавить все необходимые для выполнения работы элементы цепи, а затем кликнуть на плюсовом контакте на элементе, затем на минусовом контакте подключаемого элемента (рисунок А.9 – А.11). Чтобы параллельно подключить вольтметр необходимо в панели элементов выбрать «Параллельное соединение», после чего отобразиться подсказка в левом верхнем углу (рисунок А.12). Затем требуется кликнуть по проводу откуда необходимо начать подключение (рисунок А.13) и кликнуть по контакту вольтметра, этот алгоритм проделать и для второго контакта (рисунок А.14).

Также есть возможность просмотреть значения элементов цепи в панели значений (рисунок А.15). Для завершения выполнения лабораторной работы необходимо нажать кнопку в правом верхнем углу главной формы. После этого пользователю откроется форма для ввода текста вывода о проделанной лабораторной работе (рисунок А.16). Для сохранения текста вывода необходимо нажать на кнопку «Сохранить».

После сохранения вывода пользователю откроется форма для ответов на контрольные вопросы лабораторной работы (рисунок А.17). Для завершения выполнения лабораторной работы необходимо нажать на кнопку «Завершить работу», после чего пользователю откроется форма профиля.

# ЗАКЛЮЧЕНИЕ

В результате написания курсовой работы спроектировано приложение для выполнения лабораторных работ по физике по разделу «Электричество» с целью изучения учебного материала без наличия оборудованной физической лаборатории.

Приложение позволяет проводить лабораторные работы по физике по разделу «Электричество», предоставлять возможность производить регистрацию и авторизацию учащихся, собирать электрические схемы, наглядно демонстрировать работу элементов цепи, формировать отчет по выполненной работе в формате .docx.

Для решения задачи был использован язык разработки С#.

Приложение «ElectronVPL» может использоваться учащимися для изучения учебного материала по физике по разделу «Электричество».

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Какие бывают планы и как их составлять правильно // Сайт Выучить.WORK

URL: <https://vyuchit.work/samorazvitie/kakie-byvayut-plany.html> (дата обращения: 03.10.2020)

1. Лучшие плановщики задач 2020 // Сайт Комсомольская правд

URL: <https://www.kp.ru/putevoditel/tekhnologii/luchshie-planirovshhiki-zadach/> (дата обращения: 04.10.2020)

1. 40 сервисов для управления задачами и проектами // Сайт Vc.ru

URL: <https://vc.ru/services/50333-40-servisov-dlya-upravleniya-zadachami-i-proektami> (дата обращения: 05.10.2020)

1. Документация по Visual Studio // Сайт Microsoft

URL: https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019 (дата обращения: 10.10.2020)

1. DB Browser for SQLite - Браузер баз данных для // Сайт Континент свободы

URL: <https://xn--90abhbolvbbfgb9aje4m.xn--p1ai/%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%87%D0%B8%D0%BA%D1%83/%D0%BF%D1%80%D0%BE%D1%87%D0%B5%D0%B5/db-browser-for-sqlite-%D0%B1%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80-%D0%B1%D0%B0%D0%B7-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-%D0%B4%D0%BB%D1%8F-sqlite.html> (дата обращения: 12.10.2020)

1. Adobe Photoshop // Сайт Adobe

URL: https://www.adobe.com/ru/covid-19-response.html (дата обращения: 16.10.2020)

1. Что такое Figma и как ей пользоваться // Сайт Skillbox

URL:https://skillbox.ru/media/design/chto\_takoe\_figma/?advcake\_params=8deacff4dae128ade3290fad7248a9c7&utm\_source=advcake&utm\_medium=cpa&utm\_campaign=admitad&utm\_content=1010045 (дата обращения: 18.10.2020)

# ПРИЛОЖЕНИЕ А Экранные формы

Рисунок A.1 – Приветствующая форма приложения

Рисунок A.2 – Форма авторизации учащегося

Рисунок A.3 – Форма регистрации учащегося

Рисунок A.4 – Форма профиля учащегося

Рисунок A.5 – Форма выбора лабораторной работы

Рисунок A.6 – Форма выбора лабораторной работы

Рисунок A.7 – Форма для выполнения ЛР с открытой панелью элементов

Рисунок A.8 – Форма для выполнения ЛР с отображением места добавления источника напряжения

Рисунок A.9 – Форма для выполнения ЛР с добавленым источником напряжения

Рисунок A.10 – Форма для выполнения ЛР с добавлеными элементами цепи

Рисунок A.11 – Форма для выполнения ЛР с подключенным источником напряжения и амперметром

Рисунок A.12 – Форма для выполнения ЛР с подключенными элементами цепи, а также выводом подсказки для параллельного подключения вольтметра

Рисунок A.13 – Форма для выполнения ЛР с подключенными элементами цепи, а также выводом подсказки для параллельного подключения вольтметра

Рисунок A.14 – Форма для выполнения ЛР с подключенными элементами цепи

Рисунок A.15 – Форма выполнения ЛР с открытой панелью значений элементов цепи

Рисунок A.16 – Форма написания вывода

Рисунок A.17 – Форма для ответа на контрольные вопросы

# ПРИЛОЖЕНИЕ Б Фрагменты листинга

Листинг Б.1 – SQL-запрос на создание базы данных

DROP TABLE IF EXISTS sched\_task;

DROP TABLE IF EXISTS user\_dir;

DROP TABLE IF EXISTS achiev\_categ;

DROP TABLE IF EXISTS subtask;

DROP TABLE IF EXISTS schedule;

DROP TABLE IF EXISTS day;

DROP TABLE IF EXISTS task;

DROP TABLE IF EXISTS target;

DROP TABLE IF EXISTS direction;

DROP TABLE IF EXISTS note;

DROP TABLE IF EXISTS data\_service;

DROP TABLE IF EXISTS achievement;

DROP TABLE IF EXISTS category;

DROP TABLE IF EXISTS user;

CREATE TABLE "note"

(

id\_note INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,

id\_user INTEGER NOT NULL,

text TEXT NOT NULL CHECK(length(text) != 0),

FOREIGN KEY (id\_user) REFERENCES user (id\_user)

);

CREATE TABLE "data\_service"

(

id\_service INTEGER NOT NULL,

id\_user INTEGER NOT NULL,

login TEXT NOT NULL CHECK(length(login) > 0),

password TEXT NOT NULL CHECK(length(password) > 0),

em\_ph TEXT NULL,

FOREIGN KEY (id\_user) REFERENCES user (id\_user)

);

CREATE TABLE "user"

(

id\_user INTEGER PRIMARY KEY AUTOINCREMENT,

photo TEXT NULL,

login TEXT NOT NULL CHECK(length(login) != 0),

password TEXT NOT NULL CHECK(length(password) > 0),

email TEXT NULL,

reg\_code TEXT NOT NULL CHECK(length(reg\_code) == 4)

);

CREATE TABLE "direction"

(

id\_direct INTEGER PRIMARY KEY AUTOINCREMENT,

id\_categ INTEGER NOT NULL,

name TEXT NOT NULL CHECK(length(name) > 0),

color\_mark TEXT NOT NULL,

FOREIGN KEY (id\_categ) REFERENCES category(id\_categ)

);

/\*таблица категорий с фиксированными данными, данные не меняются в процессе работы приложения\*/

CREATE TABLE "category"

(

id\_categ INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL

);

CREATE TABLE "achievement"

(

id\_achiev INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

descr TEXT NOT NULL,

current\_score INTEGER NOT NULL CHECK(current\_score >= 0),

final\_score INTEGER NOT NULL,

status INTEGER NOT NULL CHECK(status == 0 OR status == 1)

);

CREATE TABLE "target"

(

id\_target INTEGER PRIMARY KEY AUTOINCREMENT,

id\_direct INTEGER NOT NUll,

name TEXT NOT NULL,

FOREIGN KEY (id\_direct) REFERENCES direction(id\_direct)

);

CREATE TABLE "task"

(

id\_task INTEGER PRIMARY KEY AUTOINCREMENT,

id\_target INTEGER NOT NUll,

text TEXT NOT NULL,

descr TEXT NULL,

date TEXT NOT NULL DEFAULT CURRENT\_DATE,

time TEXT NULL DEFAULT CURRENT\_TIME,

time\_finish TEXT NULL,

failed INTEGER NOT NULL CHECK(failed == 0 OR failed == 1),

status INTEGER NOT NULL CHECK(status == 0 OR status == 1),

FOREIGN KEY (id\_target) REFERENCES target(id\_target)

);

CREATE TABLE "subtask"

(

id\_subtask INTEGER PRIMARY KEY AUTOINCREMENT,

id\_task INTEGER NOT NUll,

text TEXT NOT NULL CHECK(length(text) > 0),

status INTEGER NOT NULL DEFAULT 0 CHECK(status == 0 OR status == 1),

FOREIGN KEY (id\_task) REFERENCES task (id\_task) ON DELETE CASCADE

);

CREATE TABLE "day"

(

id\_day INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL CHECK(length(name) > 0)

);

CREATE TABLE "schedule"

(

id\_sched INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL CHECK(length(name) > 0)

);

/\* промежуточные таблицы для связей M:N \*/

CREATE TABLE "user\_dir"

(

id\_user INTEGER,

id\_direct INTEGER,

FOREIGN KEY (id\_user) REFERENCES user(id\_user),

FOREIGN KEY (id\_direct) REFERENCES direction(id\_direct)

);

CREATE TABLE "achiev\_categ"

(

id\_categ INTEGER,

id\_achiev INTEGER,

FOREIGN KEY (id\_categ) REFERENCES category(id\_categ),

FOREIGN KEY (id\_achiev) REFERENCES achievement(id\_achiev)

);

CREATE TABLE "sched\_task"

(

id\_sched INTEGER NOT NULL,

id\_task INTEGER NOT NULL,

id\_day INTEGER NOT NULL,

FOREIGN KEY (id\_sched) REFERENCES schedule(id\_sched),

FOREIGN KEY (id\_task) REFERENCES task(id\_task),

FOREIGN KEY (id\_day) REFERENCES day(id\_day)

);

Листинг Б.2 – Файл «DBService.cs»

using System;

using System.Collections.Generic;

using System.Data.SQLite;

using System.IO;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Upgrade.Classes;

using Upgrade.Forms;

namespace Upgrade

{

class DBService

{

static public void ConnectToDB(string databaseName)

{

if (File.Exists(databaseName))

{

ServiceData.connect = new SQLiteConnection(@"Data Source="+ databaseName + "; Version=3;");

ServiceData.connect.Open(); // открытие соединения

}

else

{

if (MessageBox.Show(

"Не удалось подключиться к базе данных.\nФайл базы данных отсутствует.\nДля подключения будет создан новый файл базы данных.",

"Ошибка подключения к базе данных", MessageBoxButtons.OK, MessageBoxIcon.Information) == DialogResult.OK)

{

CreateDB(databaseName);

}

}

}

static public void CloseConnectionWithDB()

{

ServiceData.connect.Close(); // закрытие соединения

}

static private void CreateDB(string databaseName)

{

SQLiteConnection.CreateFile(databaseName);

ConnectToDB(databaseName);

ServiceData.commandText = File.ReadAllText(@"database\create\_db.sql");

ServiceData.command = new SQLiteCommand(ServiceData.commandText, ServiceData.connect);

ServiceData.command.ExecuteNonQuery();

}

static public string GetMD5Hash(string text)

{

using (var hashAlg = MD5.Create()) // Создаем экземпляр класса реализующего алгоритм MD5

{

byte[] hash = hashAlg.ComputeHash(Encoding.UTF8.GetBytes(text)); // Хешируем байты строки text

var builder = new StringBuilder(hash.Length \* 2); // Создаем экземпляр StringBuilder. Этот класс предназначен для эффективного конструирования строк

for (int i = 0; i < hash.Length; i++)

{

builder.Append(hash[i].ToString("X2")); // Добавляем к строке очередной байт в виде строки в 16-й системе счисления

}

return builder.ToString(); // Возвращаем значение хеша

}

}

static public bool Registration(string login,

string password,

string email)

{

ServiceData.commandText = @"SELECT \* FROM user WHERE login = @login AND password = @password";

ServiceData.command = new SQLiteCommand(ServiceData.commandText, ServiceData.connect);

ServiceData.command.Parameters.AddWithValue("@login", login);

ServiceData.command.Parameters.AddWithValue("@password", GetMD5Hash(password));

ServiceData.reader = ServiceData.command.ExecuteReader();

if (!ServiceData.reader.HasRows)

{

// регистрация

if (GlobalData.RegistrationCode == null)

{

GlobalData.RegistrationCode = GlobalData.GererateCode();

}

if (email != "")

{

if (MailSender.SendMail(email,

"Подтверждение регистрации",

"Это ваш регистрационный код, который необходим для " +

"подтверждения вашей электронной почты и восстановления данных аккаунта",

GlobalData.RegistrationCode))

{

try

{

ServiceData.commandText = @"INSERT INTO user ('login', 'password', 'email', 'reg\_code')

VALUES(@login, @password, @email, @regCode); ";

ServiceData.command = new SQLiteCommand(ServiceData.commandText, ServiceData.connect);

ServiceData.command.Parameters.AddWithValue("@login", login);

ServiceData.command.Parameters.AddWithValue("@password", GetMD5Hash(password));

ServiceData.command.Parameters.AddWithValue("@email", email);

ServiceData.command.Parameters.AddWithValue("@regCode", GlobalData.RegistrationCode);

ServiceData.command.ExecuteNonQuery();

ServiceData.commandText = @"INSERT INTO user\_dir ('id\_user', 'id\_direct')

VALUES(@idUser, @idDirect); ";

ServiceData.command = new SQLiteCommand(ServiceData.commandText, ServiceData.connect);

ServiceData.command.Parameters.AddWithValue("@idUser", User.userId);

ServiceData.command.Parameters.AddWithValue("@idDirect", 1);

ServiceData.command.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

MessageBox.Show(

"Не удалось зарегистрироваться...\n-\nОшибка: " + ex.Message,

"Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

}

else

{

MessageBox.Show(

"Не удалось зарегистрироваться...\n\nПроверьте правильность указанной электронной почты",

"Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

}

else

{

try

{

ServiceData.commandText = @"INSERT INTO user ('login', 'password', 'email', 'reg\_code')

VALUES(@login, @password, @email, @regCode); ";

ServiceData.command = new SQLiteCommand(ServiceData.commandText, ServiceData.connect);

ServiceData.command.Parameters.AddWithValue("@login", login);

ServiceData.command.Parameters.AddWithValue("@password", GetMD5Hash(password));

ServiceData.command.Parameters.AddWithValue("@email", "отсутствует");

ServiceData.command.Parameters.AddWithValue("@regCode", GlobalData.RegistrationCode);

ServiceData.command.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

MessageBox.Show(

"Не удалось зарегистрироваться...\n-\nОшибка: " + ex.Message,

"Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

}

}

else

{

// Данные уже есть

return false;

}

}

static public bool Authorization(string login,

string password)

{

ServiceData.commandText = @"SELECT \* FROM user WHERE login = @login AND password = @password";

ServiceData.command = new SQLiteCommand(ServiceData.commandText, ServiceData.connect);

ServiceData.command.Parameters.AddWithValue("@login", login);

ServiceData.command.Parameters.AddWithValue("@password", GetMD5Hash(password));

ServiceData.reader = ServiceData.command.ExecuteReader();

if (ServiceData.reader.HasRows)

{

ServiceData.reader.ReadAsync();

User.userId = ServiceData.reader.GetInt32(ServiceData.reader.GetOrdinal("id\_user"));

User.userLogin = ServiceData.reader.GetString(ServiceData.reader.GetOrdinal("login"));

User.userPassword = ServiceData.reader.GetString(ServiceData.reader.GetOrdinal("password"));

User.userEmail = ServiceData.reader.GetString(ServiceData.reader.GetOrdinal("email"));

User.SetPhoto(ServiceData.reader.GetValue(ServiceData.reader.GetOrdinal("photo")).ToString());

GlobalData.mainWorkingForm.Show();

return true;

}

else

{

MessageBox.Show(

"Не удалось авторизоваться...\n\nПроверьте правильность введенных данных или восстановите пароль.",

"Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

}

}

}