Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное
учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

Кафедра системного анализа и телекоммуникаций

Руководитель:

к. т. н., доцент
кафедры САиТ
Ю.Ю. Липко

_____
(подпись)
«___» _____ 20__ г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

ПО ДИСЦИПЛИНЕ «ТВОРЧЕСКИЙ ПРОЕКТ»

на тему: Приложение поддержки распределения рабочей нагрузки для краткосрочного и среднесрочного планирования

Выполнил: Дмитрий Вадимович Горохов, КТбо3-2
(подпись, фамилия, имя, отчество, группа)

Баллы руководителя: _____

Баллы комиссии: _____

Таганрог 2020

Реферат

Отчёт 43 с. (включая приложения 20 с.), 4 рис., 2 таб., 21 ист., 3 прил.

ПРОКРАСТИНАЦИЯ, РАСПИСАНИЕ, POMIDORO, ПРОГРАММА, PYTHON, QT, MYSQL

В данной работе выполнено исследование широко распространенной проблемы связанной со сложностью концентрации внимания на работе в условиях современной среды, а также потерей эффективности при работе или учебе.

Определены возможные причины таких явлений и обозначены некоторые способы борьбы с ними. После чего произведена работа по разработке программного обеспечения помогающего противодействию данной проблемы.

Результатом выполнения является программа для операционной системы Windows, позволяющая автоматизировать работу с личным расписанием и обеспечивающая поддержку при выполнении задач.

Содержание

Введение 4

1 Исследование проблематики восприятия 5

1.1 Синдром дефицита внимания 5

1.2 Прокрастинация 5

2 Методики увеличения производительности человека 9

2.1 Методика Pomodoro 9

2.2 Методики планирования и контроля 10

3 Программная реализация 12

3.1 Средства разработки приложения 12

3.2 Алгоритм работы приложения 12

3.3 Разработка программы 16

3.4 Тестирование работы приложения 18

Выводы 19

Список использованных источников 20

Приложение А 23

Приложение Б 25

Приложение В 42

ВВЕДЕНИЕ

В современном обществе очень важно умение планирования и выполнения поставленных задач в установленные сроки. Менеджмент времени очень важен в условиях постоянно меняющегося окружения в любой области человеческой жизнедеятельности, будь то работа, учеба или отдых. Однако многие люди не способны грамотно распределить свое время, откладывают выполнение задач на поздние сроки, либо не способны сконцентрироваться во время выполнения задач.

Существует несколько причин подобного явления. Начиная от лени, и заканчивая поведенческими расстройствами, такими как прокрастинация или синдром дефицита внимания/гиперактивности.

Вне зависимости от причины проявления данной проблемы, с ней можно бороться, при условии, что человек сам готов к этому. Симптомы вызванные прокрастинацей и дефицитом внимания могут быть частично подавлены. Лень возможно побороть волевым усилием. А грамотное распределение времени приходит с опытом.

Необходимо разработать приложение, обеспечивающее простую и понятную систему автоматизированного расписания, позволяющее увеличить эффективность человека при работе с задачами, требующими дополнительной концентрации.

1 Исследование проблематики восприятия

1.1 Синдром дефицита внимания

Синдром дефицита внимания и гиперактивности (СДВГ) – неврологическо-поведенческое расстройство развития, начинающееся в детском возрасте [1]. Проявляется такими симптомами, как трудности концентрации внимания, гиперактивность и плохо управляемая импульсивность [2]. Также при склонности к СДВГ у взрослых возможны снижение интеллекта и трудности с восприятием информации.

В МКБ-10 СДВГ относился к гиперкинетическим расстройствам – группе эмоциональных расстройств и расстройств поведения, начинающихся обычно в детском возрасте [3], а в МКБ-11 относится к нейроонтогенетическим расстройствам [4]. С неврологической точки зрения СДВГ рассматривается как стойкий и хронический синдром, для которого не найдено способа излечения. Считается, что только 30 % детей «перерастают» этот синдром либо приспосабливаются к нему во взрослой жизни [2].

Исходя из определения и симптоматики, основной проблемой людей с диагнозом СДВГ является концентрация на длительных и монотонных задачах. Для того чтобы бороться с подобным явлением требуется разработать систему, позволяющую упростить концентрацию на выполнении задания и таким образом увеличить эффективность при монотонной работе.

1.2 Прокрастинация

Прокрастинация – склонность к постоянному откладыванию даже важных и срочных дел, приводящая к жизненным проблемам и болезненным психологическим эффектам [5].

Прокрастинация проявляется в том, что человек, осознавая необходимость выполнения вполне конкретных важных дел (например, своих должностных обязанностей), пренебрегает этой необходимостью и отвлекает своё внимание на бытовые мелочи или развлечения [6]. В той или

иной мере это состояние знакомо большинству людей и до определённого уровня считается нормальным. Прокрастинация становится проблемой, когда превращается в обычное состояние, в котором человек проводит большую часть времени. Такой человек откладывает важные дела, а когда оказывается, что все сроки уже прошли, либо просто отказывается от запланированного, либо пытается сделать всё отложенное разом, за невозможно короткий промежуток времени. В результате – дела не выполняются или выполняются некачественно, с опозданием и не в полном объёме, что приводит к соответствующим отрицательным эффектам в виде неприятностей по службе, упущенных возможностей, недовольства окружающих из-за невыполнения обязательств и т.д. Следствием этого может быть стресс, чувство вины, потеря производительности. Комбинация этих чувств и перерасхода сил (сначала – на второстепенные дела и борьбу с нарастающей тревогой, затем – на работу в авральном темпе) может спровоцировать дальнейшую прокрастинацию.

Вышеприведённое общее представление признаётся большинством исследователей данного феномена, но различные авторы расходятся в точном определении и конкретных формулировках. Тем не менее, можно выделить особенности откладывания дел, характерные для прокрастинации, что позволяет отделить этот феномен от других близких по содержанию явлений [7]:

- сам факт откладывания;
- наличие планов – откладываются дела, которые были предварительно ограничены определёнными сроками;
- осознанность – человек не забывает про важную задачу и физически может ею заниматься, он откладывает её выполнение умышленно;
- иррациональность – субъекту очевидно, что откладывание вызовет те или иные проблемы, то есть он действует заведомо вопреки своим интересам;
- стресс – откладывание вызывает негативные эмоции и утомление.

Исходя из этого, можно отделить прокрастинацию от лени (лень не сопровождается стрессом), отдыха (при отдыхе человек восполняет запасы энергии, а при прокрастинации — теряет) [8]. Также прокрастинация не равнозначна неумелому планированию, когда планы не исполняются и сдвигаются потому, что основываются на неверных оценках возможностей и производительности, хотя ошибки в планировании могут провоцировать или усугублять прокрастинацию [7].

До настоящего времени ни в западной, ни в российской психологии не сформировано единой теории прокрастинации. Нет даже общепринятого определения данного явления. Существуют исследования, пытающиеся связать склонность к прокрастинации с социальными, культурными, демографическими особенностями.

По Милграму, можно выделить два основных направления прокрастинации: откладывание *выполнения задач* и откладывание *принятия решений*. Милграм, Батори и Моурер [9] выделяют пять характерных типов прокрастинации:

- бытовая – откладывание регулярных повседневных дел, таких как уборка, стирка, работы по дому.
- прокрастинация принятия решений – постоянное затягивание выбора по любым, даже самым малозначительным вопросам, когда все условия и информация для принятия решения уже имеются.
- невротическая – затягивание жизненно важных решений, имеющих долговременные последствия: выбора учебного заведения, профессии, партнера, согласия или отказа от брака и так далее.
- академическая – откладывание выполнения учебных заданий, подготовки курсовых, экзаменов и т. п. (например: ночь перед экзаменом, когда за несколько часов изучается материал за весь курс).

- компульсивная – сочетание откладывания выполнения дел с откладыванием принятия решений, сложившееся в постоянное устойчивое поведение.

Есть несколько вариантов классификации склонных к прокрастинации людей. В опубликованном в 2005 г. исследовании Чу и Чой [10] предложено разделять прокрастинаторов на два основных типа: «пассивных» и «активных».

Таким образом, основная проблема прокрастинации – неспособность, а если точнее готовность откладывать дела до последнего. При создании графика работы, который не предполагает откладывание, может быть уменьшено влияние данной проблемы на человека, увеличение его показателей активности и качества работы в целом.

Следовательно требуется разработать систему, которая создавала бы постоянный размеренный график работы и отдыха с учетом заданных изначальных условий и приоритетов.

2 Методики увеличения производительности человека

2.1 Методика Pomodoro

Метод эффективного управления временем Pomodoro разработан итальянским студентом Франческо Чирилло. Она позволяет распределить рабочую нагрузку и увеличить производительность, при уменьшении времени работы.

Основными целями техники является:

- поддержка решимости достигать собственных целей;
- улучшение рабочего и учебного процесса;
- увеличение эффективности работы и учёбы;
- развитие решимости действовать в сложных ситуациях.

Суть техники проста, но, тем не менее, эффективна. Требуется определиться с задачей, которую следует выполнить, поставить таймер («помидор») на 25 минут. Работать, ни на что не отвлекаясь, пока таймер не прозвонит, после чего необходимо произвести короткий перерыв 3-5 минут. После каждого 4-го «помидора» необходимо выполнять длинный перерыв (15-30 минут) [11].

Эффективность техники доказана исследованиями Федерико Гоббо и Маттео Ваккари, которые наблюдали за группой программистов, работающих с техникой и без неё. Эффективность их работы по технике Pomodoro была выше [12]. Другое исследование, проведённое компанией Staples, показало неэффективность работников, которые трудятся без перерыва, кроме обеденного.

Как и любую другую технику, Pomodoro допускается адаптировать под собственные потребности. Время работы и отдыха можно уменьшать или увеличивать, ориентируясь на личные предпочтения и, конечно же, на род деятельности.

Рано или поздно наступит такой момент, когда задача будет полностью выполнена, а на таймере останется еще немало времени. В таком случае можно заняться планированием будущих задач, подведением текущих итогов, разбором прошлых неточностей, чтением качественной литературы или выполнением внеплановых заданий, не входящих в текущий список дел.

Жесткие временные рамки способны вызвать определенный дискомфорт у людей, занимающихся креативными профессиями. Сюда можно отнести и музыкантов, и писателей, и художников. «Pomodoro» будет агрессивно воздействовать на их вдохновение и разрушать атмосферу творчества [13].

Для реализации данной методики требуется простой алгоритм-таймер, который будет оповещать пользователя о начале и завершении циклов работы и отдыха. Также возможно изменение времени таймера для адаптации под рабочий ритм человека.

2.2 Методики планирования и контроля

План – основа самоконтроля. Он помогает структурировать свою деятельность. При желании любой график можно скорректировать: ужесточить или же, наоборот, ослабить. Зато, преодолев очередной этап, происходит эмоциональный подъем, позволяющий продолжать работать.

Нельзя хвататься за все – рабочий день не безразмерный. Человеку требуется достаточно времени и на работу, и на личную жизнь, и на отдых. Поэтому в список заданий на день лучше записывать 5-6 неотложных дел, а менее срочные и важные перенесите на конец недели или месяца.

Гораздо легче выполнять задание, разбив его на отдельные этапы. Для каждого этапа, однако, тоже необходимо установить четкие временные рамки.

Психологи рекомендуют сначала выполнять наиболее сложную и, возможно, не самую приятную работу. После чего перейти к тому, что нравится больше, что позволит быстро закончить запланированное на день.

Не следует также забывать о законе Парето (правило 80/20), который помогает оценить эффективность любой деятельности. Заключается данный закон в отношении выполненной работы к полученному результату, а именно: 20% усилий дают 80% результата, а остальные 80% усилий помогают добиться оставшихся 20%. Значит, нужно понять, что даст наибольший эффект, а какие последующие действия будут, по сути, лишними.

Проанализировав свои задачи и определив, какие из них попадают в те 20%, которые позволяют добиться 80% результатов, требуется уделить все внимание тем, которые действительно влияют на решение значительной части задач.

И последнее, но не менее важное – нужно забывать дома о работе. Мозгу, нервной системе необходим полноценный отдых, то есть переключение на книги, кинофильмы, общение с близкими и друзьями. И тогда каждый день вы будете готовы с еще большим усердием воплощать в жизнь свои планы [14].

Таким образом, правильное планирование требует распределения задач так, чтобы сложные или наиболее приоритетные задачи не превышали 5-6 в день, кроме того, оптимально использовать разбиение задач на подзадачи. Также не следует забывать об отдыхе иначе возможно быстрое уставание и угасание мотивации. Следовательно, для программной реализации требуется учитывать приоритетность и важность задачи для пользователя, рабочую нагрузку и время отдыха.

3 Программная реализация

3.1 Средства разработки приложения

Разработка программы велась с использованием языка программирования высокого уровня Python версии 3.7 [15] с помощью интегрированной среды разработки PyCharm.

Для создания пользовательского интерфейса использовался фреймворк QT для Python (PyQT) и приложение PyQt design [16]. QT позволяет создать оконное приложение с помощью удобного интерфейса и создать параллельные вычислительные процессы во время выполнения программы.

Информация пользователя хранится в базе данных SQL. Реализовать базу данных решено с помощью MySQL, поскольку данная РСУБД наиболее проста в управлении и легко масштабируема. Также MySQL имеет официальный модуль графического интерфейса, позволяющий быстро и эффективно настраивать и изменять базу данных под названием MySQL Workbench. Для работы MySQL с Python используется соответствующий модуль под названием pymysql [17]. Модель базы данных создана в программе ERwin Data Modeler.

3.2 Алгоритм работы приложения

В приложении присутствуют четыре основных реализованных элемента. Реализация выполнения методики Pomodoro, алгоритм распределяющий элементы расписания, модуль работы с базой данных и система аутентификации логин-пароль.

При запуске программы определяется текущее время и дата на компьютере пользователя, после чего производится поиск записанных событий в базе данных и выводятся на экран пользователя. Программа автоматически позиционируется на текущей задаче.

Если в записи события указано, что требуется дополнительная концентрация при работе, начинает работу алгоритм, реализующий методику

Pomodoro. Данные о времени одного помидора для данного пользователя берутся из базы данных. Функция получает текущее значение таймера помидора (*T*) и ожидает *T**60 секунд, после чего пользователь получает уведомление. Затем пользователю предлагается выбор: увеличить, уменьшить таймер, либо оставить его с текущим значением. Это необходимо для адаптивной подстройки программы под пользователя и более комфортной работы самого пользователя.

После указания пользователем соответствующих значений начинается отсчет таймера отдыха, значение которого равно $\frac{T}{5}$ минут. Время отдыха увеличивается до $\frac{T+5}{2}$ минут после каждого четвертого цикла.

При увеличении и уменьшении времени таймера, требуется реализовать ограничения, позволяющие завершить один полный цикл из четырех итераций до длительного перерыва. Поскольку стандартный рабочий день составляет 8 часов, предполагается, что длительный обеденный перерыв производится в середине рабочего дня. Соответственно, для определения наибольшего возможного значения таймера требуется вычислить максимальное допустимое значение для четырехчасового периода.

Определяем допустимое значение по формуле 1:

$$4T_{\max} + 3\frac{T_{\max}}{5} \le 240 мин. \quad (1)$$

После проведения расчетов получаем значение $T_{\max} \le 52$, следовательно максимальное время непрерывной работы будет равно $T_{\max} = 50$. Минимальное время целесообразно обозначить таким образом, чтобы время отдыха было не меньше 3 минут, таким образом определяется $T_{\min} = 15$.

Для определения важности события пользователю предлагается пройти тестирование для ранжирования событий по важности. Тестирование представляет собой попарное сравнение событий, после него определяются личные предпочтения того или иного события для пользователя.

При работе пользователь может определить новое событие в расписании, указать его наименование, периодичность, выбрать один из типов событий и указать, возможно ли сдвинуть данное мероприятие, при появлении в расписании более важных событий.

Также пользователь может создать цель, которая определяется датой – крайним сроком выполнения цели, временем, которое требуется затратить на данную цель, наименованием и типом события. Алгоритм, на основании ранжирования событий определяет в расписании время, когда пользователь должен выполнять работу по выполнению данной цели. Поскольку пользователь мог недооценить сложность задачи – программа автоматически увеличивает время, требуемое для выполнения цели.

Переопределение событий происходит следующим образом. Определяется – возможен ли перенос уже записанных событий, затем определяется его приоритет, после чего, если событие переносимое и его приоритет выше, то новое событие записывается на необходимое время и дату, а событие, находившееся в базе данных ранее, смещается на время, определяемое смещающим событием.

Программа использует базу данных SQL для хранения пользовательской информации. Для обеспечения возможности пользования программы несколькими людьми, и защиты их персональных данных – применена система аутентификации логин-пароль. Безопасность хранения в базе данных обеспечивает алгоритм рассчитывающий хэш-функцию и сравнивающий ее со строкой, хранящейся в базе данных.

Хэш-функция – функция, осуществляющая преобразование массива входных данных произвольной длины в выходную битовую строку установленной длины, выполняемое алгоритмом. Для реализации данной работы достаточно хэш-функции MD5, поскольку она имеет следующие преимущества.

- простоту вычисления;

- отсутствие таких входных данных, для которых вероятность коллизии наибольшая;
- возможность модификации в идеальную хеш-функцию [18].

Для увеличения надежности хэш-функции, к входным данным, вводимым пользователем (логин и пароль) добавляется *модификатор входа хэш-функции* (соль). Соль – строка данных, которая передаётся хеш-функции вместе с входным массивом данных для вычисления хэша [19].

В программе используется статичная соль, которая вместе с логином и паролем образует единую строку данных. В результате ее обработки алгоритмом MD5 образуется ключ доступа, который сравнивается с хранимым в базе данных. Получение одинакового результата образа при разных вводимых данных возможно из-за ограничений его длинны, однако практически нивелировано добавлением значения соли.

Для работы приложения требуется разработать базу данных, соответствующую алгоритмам работы приложения. Разработка базы данных велась в программе ERwin Data Modeler. Модель базы данных приложения представлена на рисунке 1.

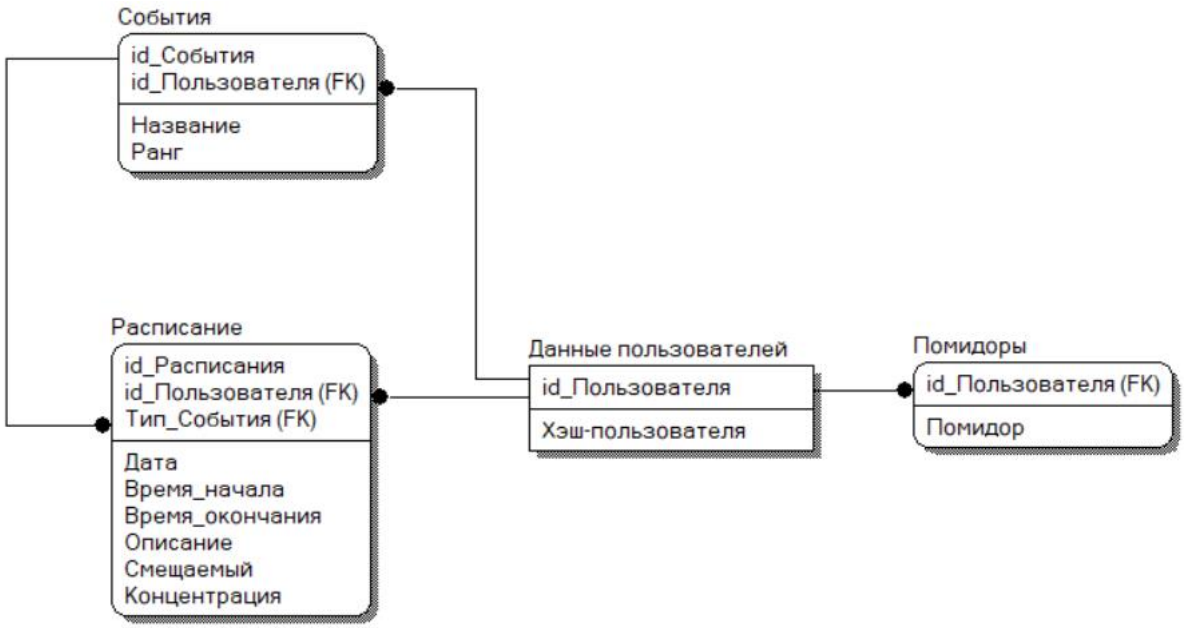

Рисунок 1 – Модель базы данных

В данной модели представлено четыре основных таблицы. Таблица «События» хранит ранги и наименования возможных событий для каждого пользователя. Таблица «Расписание» описывает элементы расписания с датой, временем начала и окончания события. Таблица «Помидоры» хранит время помидора для пользователей. Таблица «Дынные пользователей» хранит идентификатор пользователя и результат вычисления хэш-функции, для идентификации пользователя.

Обобщенная блок-схема программы представлена в приложении А.

3.3 Разработка программы

Разработка оконной части приложения велась с использованием программы PyQt design. В данной программе имеется возможность настройки вида окон, определения порядка использования элементов и частичная их настройка, перед последующей работой с основным алгоритмом. Основные окна программы имеют вид соответствующий рисунку 2.

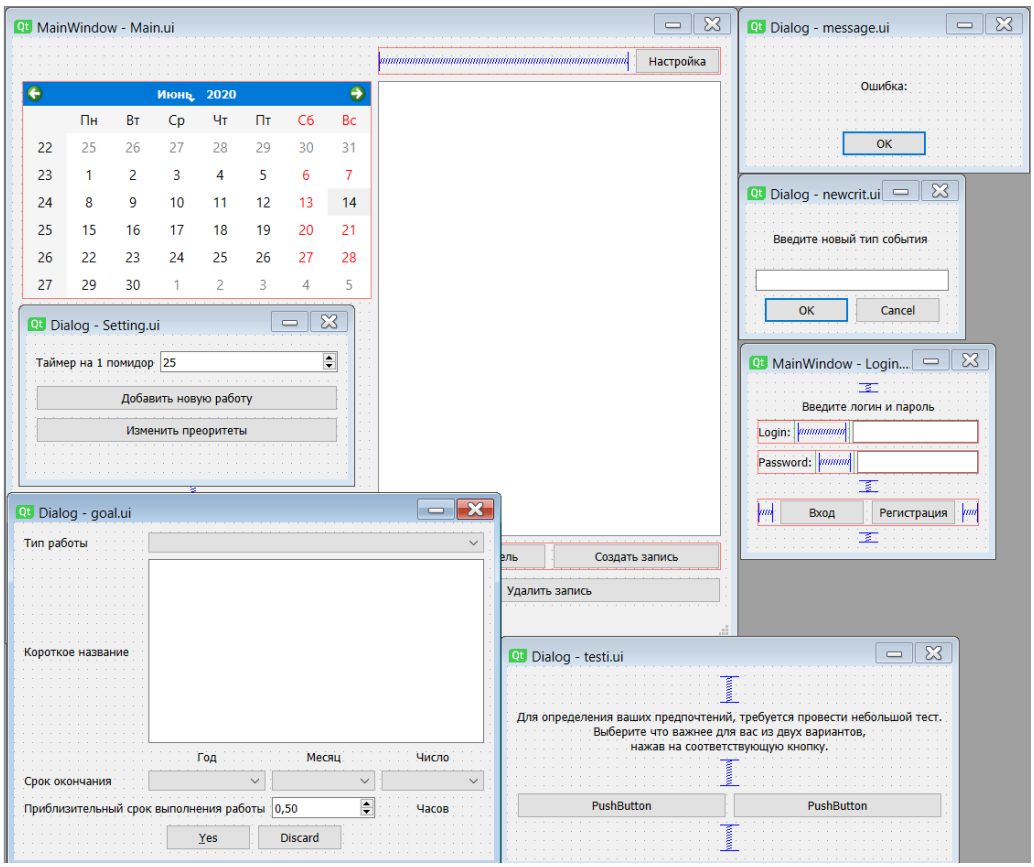

Рисунок 2 – Основные окна программы

База данных разрабатывалась с помощью графического интерфейса MySQL Workbench в соответствии с моделью базы данных представленной на рисунке 1. Работа с базой данных осуществляется с помощью отдельного файла, в котором описан класс работы с базой данных и описаны функции работы с необходимыми элементами. Для работы с базой данных был использован модуль pymysql для языка Python.

Реализация методики Pomodoro выполняется в отдельном вычислительном потоке, реализованной при помощи средств QT. Алгоритм работы представляет собой таймер, который начинает работу в случае, если указано, что текущая задача требует концентрации.

Алгоритм автоматизирующий расписание взаимодействует с графическим интерфейсом и запускается после того как пользователь определил новую цель или новое событие в расписании. Поскольку при создании масштабной цели требующей много времени на реализацию возможны длительные вычисления и работа с базой данных, данный

алгоритм также использует отдельных вычислительный поток, чтобы пользователь мог продолжать работу, пока производятся вычисления.

Листинг программного кода основного модуля программы представлен в приложении Б.

3.4 Тестирование работы приложения

При тестировании приложения были использованы тест-кейсы, разработанные с целью выявления критических ошибок работы программы.

Тест-кейс – это чёткое описание действий, которые необходимо выполнить, для того чтобы проверить работу программы (поля для ввода, кнопки и т.д.). Данное описание содержит: действия, которые надо выполнить до начала проверки — предусловия; действия, которые надо выполнить для проверки — шаги; описание того, что должно произойти, после выполнения действий для проверки — ожидаемый результат [20].

Стандартный тест-кейс состоит из заголовка, предусловия шагов проверки и ожидаемого результата. Каждый выполненный тест-кейс, оканчивается одним из следующих результатов:

- Положительный результат – фактический результат соответствует ожидаемому результату,
- Отрицательный результат – фактический результат не соответствует ожидаемому результату. В этом случае найдена ошибка.
- Выполнение теста блокировано – после одного из шагов продолжение теста невозможно. В этом случае так же, найдена ошибка.

Одним тест-кейсом проверяется одна конкретная вещь, и для этой вещи должен быть только один ожидаемый результат [21].

В ходе тестирования производится выполнение каждого шага тестового задания и проверяется получаемый результат. В результате проверки таким методом было выявлено и исправлено несколько критических ошибок. Примеры использованных тест-кейсов представлены в приложении В.

ВЫВОДЫ

В результате выполнения проектного задания были исследованы различные сферы науки, относящиеся к медицине, безопасности данным, программированию и графическому дизайну.

Разработана программа, соответствующая теме проекта и требованиям, поставленным при тестировании и выполняющая необходимые функции.

Следует отметить общую направленность созданного приложения. Несмотря на то что его основной задачей является борьба со снижением эффективности при работе, оно также пригодится людям работающим на дому, домохозяйкам и студентам.

В дальнейшем возможно улучшение программной реализации. Например, осуществление хранения данных пользователей на внешних серверах, для доступа через любой компьютер, подключенный к сети Интернет. Также возможно увеличение функционала, добавление новых методик увеличения продуктивности, отслеживания используемых программ и ограничения их запуска по соответствующим настроенным спискам.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Attention-Deficit/Hyperactivity Disorder [Electronic Resource] // National institute of mental health (NIMH). – Режим доступа: URL: https://www.nimh.nih.gov/health/topics/attention-deficit-hyperactivity-disorder-adhd/index.shtml (дата обращения: 15.05.2020).

2. Attention-Deficit/Hyperactivity Disorder [Electronic Resource] // Behavenet. – Режим доступа: URL: https://behavenet.com/attention-deficithyperactivity-disorder (дата обращения: 15.05.2020).

3. Всемирная организация здравоохранения. F90–F98 Эмоциональные расстройства и расстройства поведения, начинающиеся обычно в детском и подростковом возрасте [Текст] // Международная классификация болезней (10-й пересмотр). Класс V: Психические расстройства и расстройства поведения (F00–F99) (адаптированный для использования в Российской Федерации). – Ростов-на-Дону: Феникс, 1999. – С. 334.

4. ICD-11 for Mortality and Morbidity Statistics: 6A05 Attention deficit hyperactivity disorder [Electronic Resource] // World Health Organisation. – Режим доступа: URL: https://icd.who.int/browse11/l-m/en#/http%3a%2f%2fid.who.int%2ficd%2fentity%2f334423054 (дата обращения: 17.05.2020).

5. Тарасевич, Г.В. Прокрастинация: болезнь века [Текст] / Г.В. Тарасевич // Русский репортёр: журнал. – М.: PunaMusta Oy, 2014. – № 14 (342). – С. 20–29.

6. Ferrari, J. Procrastination and Task Avoidance: Theory, Research and Treatment [Text] / J.R. Ferrari, J.L. Johnson, M.G., William. – NY.: Plenum Press, 1995. – 266 p.

7. Чернышева, Н.А. Прокрастинация: актуальное состояние проблемы и перспективы изучения [Текст] / Н.А. Чернышева [Электронный ресурс] // Cyberleninka. – Электрон. текстовые дан. – Режим доступа: URL:

https://cyberleninka.ru/article/n/prokrastinatsiya-aktualnoe-sostoyanie-problemy-i-perspektivy-izucheniya (дата обращения: 27.05.2020).

8. Людвиг, П. Победи прокрастинацию! Как перестать откладывать дела на завтра [Текст] / П. Людвиг. – М.: Альпина Паблишер, 2014. – 263 с.

9. Milgram, N.A. Correlates of academic procrastination [Text] / N.A. Milgram, G. Batori, D. Mowrer // Journal of School Psychology. –1993. – Vol. 31. – P. 487–500.

10. Chu, A.H.C., Rethinking procrastination: Positive effects of «active» procrastination behavior on attitudes and performance [Text] / A.H.C. Chu, J.N. Choi // Journal of Social Psychology. – 2005. – Vol. 14.

11. Мураховский А. Всё, что вам нужно знать о технике Pomodoro [Электронный ресурс] // Lifehacker. – Электрон. текстовые дан. – Режим доступа: URL: https://lifehacker.ru/all-about-pomodoro/ (дата обращения: 20.05.2020).

12. Gobbo, F. The Pomodoro Technique for Sustainable Pace in Extreme Programming Teams. [Text] / F. Gobbo, M. Vaccari // Agile Processes in Software Engineering and Extreme Programming. – 2008. – P. 180–184

13. Брайт, М. «Pomodoro»: техника, приложения, плюсы и минусы [Электронный ресурс] // 4brain. – Электрон. текстовые дан. – Режим доступа: URL: https://4brain.ru/blog/pomodoro-техника (дата обращения: 22.05.2020).

14. Ильина, Н. 11 способов повышения личной эффективности [Электронный ресурс] // BBF. – Электрон. текстовые дан. – Режим доступа: URL: https://bbf.ru/magazine/26/5603/ (дата обращения: 21.05.2020).

15. Шоу, З. Легкий способ выучить Python 3 [Текст] / З.А. Шоу – М.: Эксмо, 2019 – 367 с.

16. Create User Interfaces with Qt for Python [Electronic Resource] // QT.io. – Режим доступа: URL: https://www.qt.io/qt-for-python (дата обращения: 30.05.2020).

17. PyMySQL — инструкция по использованию MySQL на примерах [Электронный ресурс] // python-scripts – Режим доступа: URL: https://python-scripts.com/pymysql (дата обращения: 02.06.2020).

18. Pearson, P.K. Fast Hashing of Variable-Length Text Strings [Text] / P.K. Pearson // Communications of the ACM. – 1990. – Vol. 33 (6). – P. 677–680

19. Кнут, Д. Искусство программирования [Текст]: Том 3. Сортировка и поиск / Д. Кнут. – М.: Вильямс, 2007. – С. 824.

20. Пишем максимально эффективный тест-кейс [Электронный ресурс] // Habr – Режим доступа: URL: https://habr.com/ru/post/246463/ (дата обращения: 06.06.2020).

21. Захаров, В. Правильно пишем тест-кейсы. Памятка начинающему специалисту по тестированию [Электронный ресурс] // victorz.ru – Режим доступа: URL: https://victorz.ru/202001101079 (дата обращения: 06.06.2020).

ПРИЛОЖЕНИЕ А

Блок-схема программы

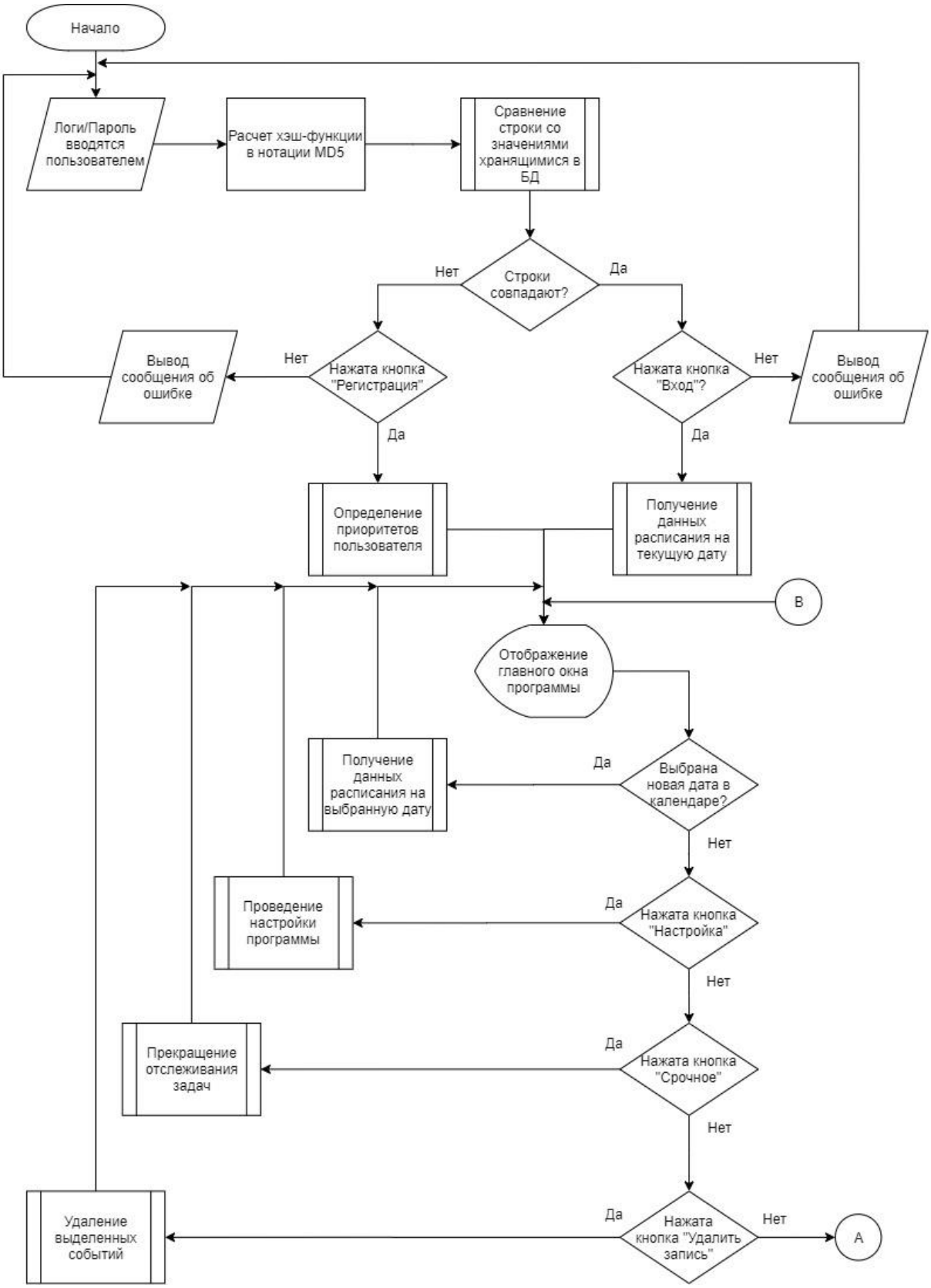

Рисунок А.1.1 - Блок-схема часть 1

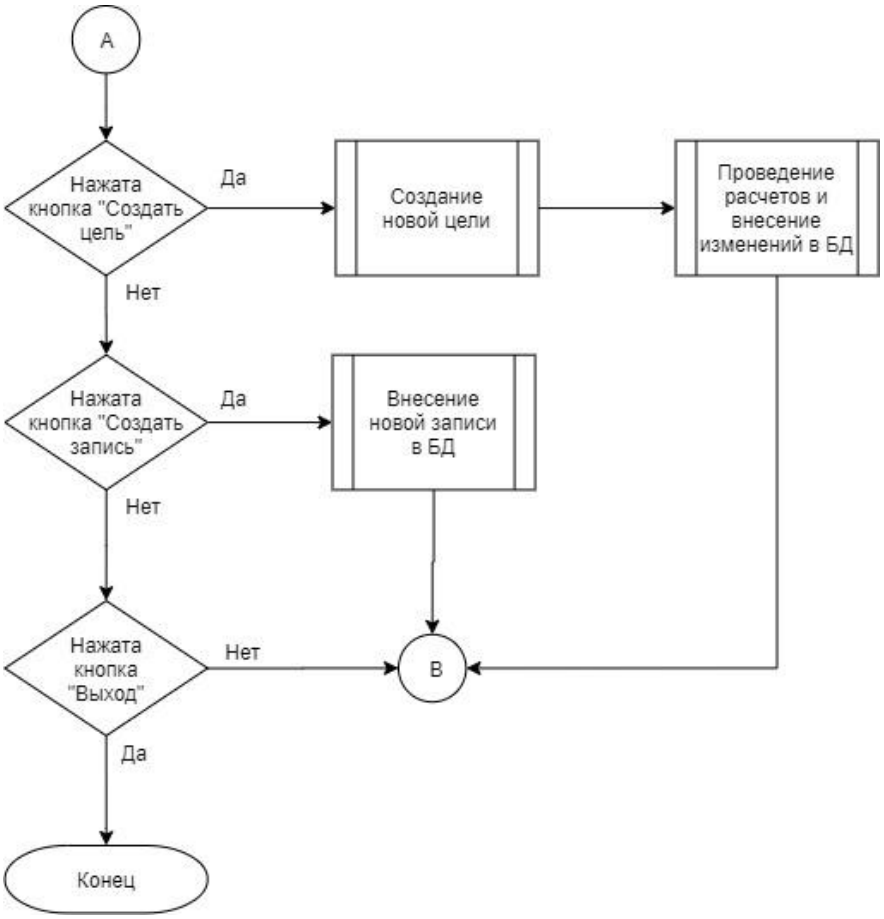

Рисунок А.1.2 - Блок-схема часть 2

ПРИЛОЖЕНИЕ Б

Листинг программы

```
from PyQt5 import QtWidgets
from PyQt5.QtCore import pyqtSignal, QDate, QObject, QThread
import DB_working
import sys
import Main
import Login
import goal
import quot
import record
import setting
import message
import testi
import newcrit
import time
import rest
#Окно выводимое после завершения рабочего таймера
class restwindow(QtWidgets.QDialog):
    Pom = pyqtSignal(int)

    def __init__(self, pomidor):
        super(restwindow, self).__init__()
        self.ui = rest.Ui_Dialog()
        self.ui.setupUi(self)
        if pomidor == 15:
            self.ui.Less.setEnabled(False)
        if pomidor == 50:
            self.ui.More.setEnabled(False)
        self.ui.Less.clicked.connect(self.changing)
        self.ui.Midlle.clicked.connect(self.changing)
        self.ui.More.clicked.connect(self.changing)

    def changing(self):
        sender = self.sender()
        if sender == self.ui.Less:
            self.Pom.emit(-5)
        if sender == self.ui.More:
            self.Pom.emit(5)
        self.close()
#Пересчет и изменение текущих значений расписания
class recalculating(QThread):
    messag = pyqtSignal(str)
```

```
def __init__(self, id, worktype, description, shedultype, date, concentration, starttime='7:00',
             endtime='23:00', sdvig=0, timetowork=0.5):
    self.id = id
    self.worktype = worktype
    self.description = description
    self.shedultipe = shedultype
    self.date = date
    self.starttime = starttime
    self.endtime = endtime
    self.concentration = concentration
    self.move = sdvig
    self.timetowork = timetowork
    self.DataBase = DB_working.DBwork()

def run(self):
    if self.shedultipe == 0:
        data = (self.id, self.date)
        shed = self.DataBase.GetShedul(data)
        flag = True
        for i in shed:
            flag = False
            tmstrt = str(i.get('StartIvent'))
            if len(tmstrt) == 7:
                tmstrt = tmstrt[:4]
            else:
                tmstrt = tmstrt[:5]
            tmend = str(i.get('EndIvent'))
            if len(tmend) == 7:
                tmend = tmend[:4]
            else:
                tmend = tmend[:5]
            if tmstrt <= self.starttime and self.endtime <= tmend:
                if i.get('movable') and self.chan:
                    self.messag.emit('В данное время уже запланировано другое мероприятие')
                    break
                elif i.get('movable') and not self.chan:
                    recal = 0
                elif self.chan:
                    recal = 0
                else:
                    recal = 0
            else:
                self.DataBase.ShedulInput(self.allin)
        if flag:
            self.DataBase.ShedulInput(self.allin)
```

```
#Таймер Помидоров  и времени простоя в отдельном потоке
class timer(QThread):
    output = pyqtSignal(str, int)
    def inicilize(self, timevalue=1, usetimer=0, circlenuber=0):
        self.value = timevalue * 60
        self.uses = usetimer
        self.circle = circlenuber

    def run(self):
        while True:
            if self.uses:
                QThread.sleep(self.value)
                self.output.emit('work', self.circle)
                if self.circle == 3:
                    self.circle = 0
                    QThread.sleep(int((self.value+300)/2))
                    self.output.emit('rest', self.circle)
                else:
                    QThread.sleep(int(self.value/5))
                    self.circle += 1
                    self.output.emit('rest', self.circle)
            else:
                QThread.sleep(int(self.value))
                self.output.emit('timepass', 0)
#Создание нового критерия
class newcritmess(QtWidgets.QDialog):
    list = pyqtSignal(str)
    def __init__(self, ID):
        super(newcritmess, self).__init__()
        self.ID = ID
        self.ui = newcrit.Ui_Dialog()
        self.ui.setupUi(self)
        self.ui.lineEdit.textChanged[str].connect(self.onChang)
        self.ui.buttonBox.accepted.connect(self.acceptin)

    def onChang(self, text):
        self.line = text

    def acceptin(self):
        self.list.emit(self.line)
        self.close()

#Сообщение об ошибке или предупреждение
class messagewindow(QtWidgets.QDialog):
    hash = pyqtSignal(int)
    def __init__(self, line):
```

```
        super(messagewindow, self).__init__()
        self.ui = message.Ui_Dialog()
        self.ui.setupUi(self)
        self.ui.label.setText(line)
        self.ui.buttonBox.accepted.connect(self.ok)

    def ok(self):
        self.hash.emit(1)
        self.close()

#Окно настройки (Помидор, новый вид события, перераспределение рангов)
class settingwindow(QtWidgets.QDialog):
    def __init__(self, ID):
        self.ID = ID
        super(settingwindow, self).__init__()
        self.DataBase = DB_working.DBwork()
        self.crit = self.DataBase.GetCrit(self.ID)
        self.ui = setting.Ui_Dialog()
        self.ui.setupUi(self)
        pom = self.DataBase.GetPomodoro(self.ID)
        self.ui.Timer.setProperty("value", pom[0].get('pomodoro'))
        self.ui.Timer.valueChanged.connect(self.changingValue)
        self.ui.Priority.clicked.connect(self.ChangingPriority)
        self.ui.NewWork.clicked.connect(self.NewPriority)

    def changingValue(self):
        data = [self.ui.Timer.value(), self.ID]
        self.DataBase.ChangingSetting(data)

    def ChangingPriority(self):
        names = []
        for i in (self.crit):
            names.append(i.get('name'))
        self.TEST = testing(names)
        self.TEST.show()
        self.TEST.criteriors[list, list].connect(self.critInput)


    def critInput(self, criteriors, line):
        data = []
        for i in range(len(line)):
            k = (line[i], self.ID, criteriors[i])
            data.append(k)
        self.DataBase.ChangingCriterior(data)
```

```
    def NewPriority(self):
        self.new = newcritmess(self.ID)
        self.new.show()
        self.new.list[str].connect(self.jin)

    def jin(self, listin):
        self.listin = listin
        names = []
        for i in (self.crit):
            names.append(i.get('name'))
        names.append(self.listin)
        self.TEST = testing(names)
        self.TEST.show()
        self.TEST.criteriors[list, list].connect(self.newcritInput)

    def newcritInput(self, criteriors, line):
        for i in range(len(line)):
            if criteriors[i] == self.listin:
                data = [(self.ID, criteriors[i], line[i])]
                self.DataBase.CritInput(data)
        self.critInput(criteriors, line)

class quotwindow(QtWidgets.QDialog):
    timing = pyqtSignal(int, bool)
    def __init__(self, flag):
        super(quotwindow, self).__init__()
        self.flag = flag
        self.ui = quot.Ui_Dialog()
        self.ui.setupUi(self)
        self.ui.buttonBox.accepted.connect(self.tm)

    def tm(self):
        if self.flag:
            self.flag = False
        else:
            self.flag = True
        self.timing.emit(time.clock(), self.flag)
        self.close()
#Окно записи нового события
class recordwindow(QtWidgets.QDialog):
    def __init__(self, ID, dates):
        super(recordwindow, self).__init__()
        self.DataBase = DB_working.DBwork()
        self.id = ID
        self.ui = record.Ui_Dialog()
        self.ui.setupUi(self)
```

```
self.crit = self.DataBase.GetCrit(ID)
for i in range(len(self.crit)):
    # Наполняем список.
    self.ui.WorkCheck.addItem(self.crit[i].get('name'))
self.ui.PeriodCheck.addItem('Без повторения')
self.ui.PeriodCheck.addItem('Раз в неделю')
self.ui.PeriodCheck.addItem('Раз в 14 дней')
self.ui.PeriodCheck.addItem('Раз в месяц')
self.ui.PeriodCheck.addItem('Будние дни')
self.ui.PeriodCheck.addItem('Выходные')
for i in range(2010, 2100):
    # Наполняем список.
    self.ui.year.addItem('%d' % i)
for i in range(1, 32):
    # Наполняем список.
    self.ui.date.addItem('%d' % i)
self.ui.mounth.addItem('Январь')
self.ui.mounth.addItem('Февраль')
self.ui.mounth.addItem('Март')
self.ui.mounth.addItem('Апрель')
self.ui.mounth.addItem('Май')
self.ui.mounth.addItem('Июнь')
self.ui.mounth.addItem('Июль')
self.ui.mounth.addItem('Август')
self.ui.mounth.addItem('Сентябрь')
self.ui.mounth.addItem('Октябрь')
self.ui.mounth.addItem('Ноябрь')
self.ui.mounth.addItem('Декабрь')
self.ui.year.setCurrentIndex(dates[0] - 2010)
self.ui.mounth.setCurrentIndex(dates[1]-1)
self.ui.date.setCurrentIndex(dates[2]-1)
self.worktipe = self.crit[0].get('idtypes')
self.period = 'Без повторения'
self.shr = 7    #start hour
self.smin = 0   #start min
self.ehr = 7    #end hour
self.emin = 30  #end min
self.year = dates[0]
self.mounth = dates[1]
self.date = dates[2]
self.descript = ''
self.chan = False
self.concent = False
self.ui.movable.stateChanged.connect(self.changing)
self.ui.Concentrate.stateChanged.connect(self.changing)
self.ui.starthour.valueChanged.connect(self.changingValue)
```

```
    self.ui.startminute.valueChanged.connect(self.changingValue)
    self.ui.endhour.valueChanged.connect(self.changingValue)
    self.ui.endminute.valueChanged.connect(self.changingValue)
    self.ui.year.currentIndexChanged.connect(self.datachange)
    self.ui.mounth.currentIndexChanged.connect(self.datachange)
    self.ui.date.currentIndexChanged.connect(self.datachange)
    self.ui.PeriodCheck.currentIndexChanged.connect(self.changingValue)
    self.ui.WorkCheck.currentIndexChanged.connect(self.workinput)
    self.ui.buttonBox.accepted.connect(self.calculating)
    self.ui.buttonBox.rejected.connect(self.close)

def workinput(self):
    sender = self.sender()
    for i in range(len(self.crit)):
        if self.crit[i].get('name') == sender.currentText():
            self.worktipe = self.crit[i].get('idtypes')
            break

def txt(self, line):
    self.descript = line

def datachange(self):
    sender = self.sender()
    if sender == self.ui.year:
        self.year = sender.currentText()
    if sender == self.ui.mounth:
        self.mounth = sender.currentIndex()+1
    if sender == self.ui.date:
        self.date = sender.currentIndex()+1


def changing(self):
    sender = self.sender()
    if sender == self.ui.movable:
        if self.chan:
            self.chan = False
        else:
            self.chan = True
    else:
        if self.concent:
            self.concent = False
        else:
            self.concent = True

def calculating(self):
    Date = str(str(self.year) + '-' + str(self.mounth) + '-' + str(self.date))
```

```
        if self.smin == 0:
            timestart = str(str(self.shr) + ':' + '00')
        else:
            timestart = str(str(self.shr) + ':' + str(self.smin))
        if self.emin == 0:
            timend = str(str(self.ehr) + ':' + '00')
        else:
            timend = str(str(self.ehr) + ':' + str(self.emin))
        self.descript = self.ui.WorkEdit.toPlainText()
        self.calculation = recalculating(self.id, self.worktipe, self.descript, self.period, Date,
self.concent, timestart, timend, self.chan)
        self.calculation.messag.connect(self.out)
        self.calculation.run()

    def out(self, line):
        mess = messagewindow(line)
        mess.show()

    def changingValue(self):
        sender = self.sender()
        if sender == self.ui.starthour:
            self.shr = sender.text()
        if sender == self.ui.startminute:
            self.smin = sender.text()
        if sender == self.ui.endhour:
            self.ehr = sender.text()
        if sender == self.ui.endminute:
            self.emin = sender.text()
        if sender == self.ui.PeriodCheck:
            self.period = sender.currentIndex()


#Создание новой цели
class goalwindow(QtWidgets.QDialog):
    def __init__(self, ID, dates):
        super(goalwindow, self).__init__()
        self.DataBase = DB_working.DBwork()
        self.ui = goal.Ui_Dialog()
        self.ui.setupUi(self)
        self.crit = self.DataBase.GetCrit(ID)
        for i in range(len(self.crit)):
            # Наполняем список.
            self.ui.WorkTipe.addItem(self.crit[i].get('name'))
        for i in range(2010, 2100):
            # Наполняем список.
            self.ui.year.addItem('%d' % i)
```

```
        for i in range(1, 32):
            # Наполняем список.
            self.ui.data.addItem('%d' % i)
        self.ui.mounth.addItem('Январь')
        self.ui.mounth.addItem('Февраль')
        self.ui.mounth.addItem('Март')
        self.ui.mounth.addItem('Апрель')
        self.ui.mounth.addItem('Май')
        self.ui.mounth.addItem('Июнь')
        self.ui.mounth.addItem('Июль')
        self.ui.mounth.addItem('Август')
        self.ui.mounth.addItem('Сентябрь')
        self.ui.mounth.addItem('Октябрь')
        self.ui.mounth.addItem('Ноябрь')
        self.ui.mounth.addItem('Декабрь')
        self.ui.year.setCurrentIndex(dates[0] - 2010)
        self.ui.mounth.setCurrentIndex(dates[1])
        self.ui.data.setCurrentIndex(dates[2] - 1)
        self.year = dates[0]
        self.mounth = dates[1]
        self.data = dates[2]
        self.worktipe = self.crit[0].get('idtypes')
        self.concent = False
        self.worktime = 0.5
        self.ui.year.currentIndexChanged.connect(self.datachange)
        self.ui.mounth.currentIndexChanged.connect(self.datachange)
        self.ui.data.currentIndexChanged.connect(self.datachange)
        self.ui.WorkLoad.valueChanged.connect(self.work)
        self.ui.WorkTipe.currentIndexChanged.connect(self.workinput)
        self.ui.Concentrate.stateChanged.connect(self.switch)
        self.ui.buttonBox.accepted.connect(self.calculating)
        self.ui.buttonBox.rejected.connect(self.close)

    def work(self):
        sender = self.sender()
        self.worktime = sender.text()

    def switch(self):
        if self.concent:
            self.concent = False
        else:
            self.concent = True

    def workinput(self):
        sender = self.sender()
        for i in range(len(self.crit)):
```

```
            if self.crit[i].get('name') == sender.currentText():
                self.worktipe = self.crit[i].get('idtypes')
                break

    def datachange(self):
        sender = self.sender()
        if sender == self.ui.year:
            self.year = sender.currentText()
        if sender == self.ui.mounth:
            self.mounth = sender.currentIndex() + 1
        if sender == self.ui.data:
            self.data = sender.currentIndex() + 1

    def calculating(self):
        Date = str(str(self.year) + '-' + str(self.mounth) + '-' + str(self.data))
        self.descript = self.ui.WorkName.toPlainText()
        self.calculation = recalculating(self.id, self.worktipe, self.descript, 6, Date, self.concent,
timetowork=self.worktime)
        self.calculation.messag.connect(self.out)
        self.calculation.run()

    def out(self, line):
        mess = messagewindow(line)
        mess.show()
#выполнение ранжирования
class testing(QtWidgets.QDialog):
    criteriors = pyqtSignal(list, list)

    def __init__(self, line):
        super(testing, self).__init__()
        self.ui = testi.Ui_Dialog()
        self.ui.setupUi(self)
        self.line = line
        self.library = []
        self.dust = []
        for i in range(len(self.line)):
            self.library.append(0)
            self.dust.append(i)
        self.j = 1
        self.minimum = []
        self.maximum = []
        self.work()

    def work(self):
        self.ui.LeftButton.setText(self.line[self.dust[0]])
        self.ui.RightButton.setText(self.line[self.dust[self.j]])
```

```
        self.ui.LeftButton.clicked.connect(self.react)
        self.ui.RightButton.clicked.connect(self.res)

    def react(self):
        self.minimum.append(self.dust[self.j])
        self.jek()

    def res(self):
        self.maximum.append(self.dust[self.j])
        self.jek()

    def jek(self):
        flag = False
        if self.j < (len(self.dust)-1):
            self.j += 1
            self.ui.RightButton.setText(self.line[self.dust[self.j]])
        else:
            self.library[self.dust[0]] += len(self.minimum)
            for i in self.maximum:
                self.library[i] += len(self.minimum)+1
            flag = True
            i = 0
            while i < (len(self.library)-1):
                N = i
                for j in range(len(self.library)-N-1):
                    if self.library[N] == self.library[j+N+1]:
                        if flag:
                            flag = False
                            self.dust.clear()
                            self.minimum.clear()
                            self.maximum.clear()
                            self.dust = [i, j+i+1]
                            i = len(self.library)-1
                        else:
                            self.dust.append(j+N+1)
                i += 1
            self.j = 1
            self.ui.LeftButton.setText(self.line[self.dust[0]])
            self.ui.RightButton.setText(self.line[self.dust[self.j]])
        if flag:
            self.criteriors.emit(self.line, self.library)
            self.close()
#Окно аутентификации
class logwindow(QtWidgets.QDialog):
    hash = pyqtSignal(str, str)
```

```
    def __init__(self):
        super(logwindow, self).__init__()
        self.ui = Login.Ui_Dialog()
        self.ui.setupUi(self)
        self.ui.Login_2.clicked.connect(self.LogClick)
        self.ui.Registration.clicked.connect(self.RegClick)

    def LogClick(self):
        self.ui.Hash()
        self.otsend('Login')

    def RegClick(self):
        self.ui.Hash()
        self.otsend('Registration')

    def otsend(self, call):
        self.hash.emit(self.ui.hashline, call)
        self.close()
#Главное окно приложения
class mainwindow (QtWidgets.QMainWindow):
    def __init__(self):
        super(mainwindow, self).__init__()
        self.ui = Main.Ui_MainWindow()
        self.ui.setupUi(self)
        self.currentData = time.time()
        # создадим объект для выполнения кода в другом потоке
        self.timethread = timer()
        self.timethread.output.connect(self.circle)
        self.DataBase = DB_working.DBwork()
        self.timing = False
        self.delay = 0
        header = ['Время', 'Описание', 'Тип']
        self.ui.ShedulWidget.setHorizontalHeaderLabels(header)
        self.rows = 48
        self.colomns = 3
        self.ui.ShedulWidget.setRowCount(self.rows)
        self.ui.ShedulWidget.setColumnCount(self.colomns)
        self.ui.ShedulWidget.setColumnWidth(0, 50)
        self.ui.ShedulWidget.setColumnWidth(1, 230)
        self.ui.ShedulWidget.setColumnWidth(2, 100)
        self.ui.ShedulWidget.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)
        self.times = []
        self.tm = time.strftime('%H:%M', time.localtime(self.currentData))
        for i in range(self.rows):
            if i % 2:
                out = QtWidgets.QTableWidgetItem(str(int(i / 20)) + str(int((i-1)/2) % 10) + ":30\n" +
```

```
str(int((i + 1) / 20)) + str(int((i+1)/2) % 10) + ":00")
            else:
                self.times.append(str(int(i / 20)) + str(int(i/2) % 10) + ":00")
                self.times.append(str(int(i / 20)) + str(int(i/2) % 10) + ":30")
                out = QtWidgets.QTableWidgetItem(str(int(i / 20)) + str(int(i/2) % 10) + ":00\n" +
str(int(i / 20)) + str(int(i/2) % 10) + ":30")
            self.ui.ShedulWidget.setItem(i, 0, out)
            self.ui.ShedulWidget.setRowHeight(i, 50)
        self.ChangingTable()
        self.ui.GoalB.clicked.connect(self.GoalClick)
        self.ui.RecordB.clicked.connect(self.RecordClick)
        self.ui.SettingB.clicked.connect(self.SettingClick)
        self.ui.SrokB.clicked.connect(self.StrokClick)
        self.ui.deleteButton.clicked.connect(self.DeleteClick)
        self.starttimer(0)
        self.ui.CalendarWidget.clicked.connect(self.ChangingTable)
        self.timethread.start()        # запустим поток

    def starttimer(self, circle):
        self.currentData = time.time()
        self.tm = time.strftime('%H:%M', time.localtime(self.currentData))
        date = time.strftime('%Y-%m-%d', time.localtime(self.currentData))
        userdata = (self.id, date)
        carrentShedul = self.DataBase.GetShedul(userdata)
        concent = True
        flag = True
        tie = time.localtime(self.currentData)
        delta = 0
        delay = 10
        for i in carrentShedul:
            SI = str(i.get('StartIvent'))
            if len(SI) == 7:
                SI = '0' + SI[:4]
            else:
                SI = SI[:5]
            EI = str(i.get('EndIvent'))
            if len(EI) == 7:
                EI = '0' + EI[:4]
            else:
                EI = EI[:5]
            if self.tm >= SI and self.tm <= EI:
                if i.get('concentration'):
                    concent = False
                    minus = time.strptime(SI, '%H:%M')
                    delta += minus.tm_hour - tie.tm_hour
                elif flag:
```

```
                flag = False
                minus = time.strptime(SI, '%H:%M')
                delay = minus.tm_hour - tie.tm_hour
        if concent:
            self.timethread.inicilize(delay, 0, 0)
        else:
            pmr = self.DataBase.GetPomodoro(self.id)
            if delta < pmr[0].get('pomodoro'):
                if delta > 15:
                    pomidor = delta
                else:
                    pomidor = 15
            else:
                pomidor = pmr[0].get('pomodoro')
            self.timethread.inicilize(pomidor, 1, circle)
            mes = messagewindow('Запущен таймер работы')
            mes.show()

    def pomidorChange(self, chang):
        pom = self.DataBase.GetPomodoro(self.id)
        kin = pom[0].get('pomodoro')
        kin += chang
        data = (kin, self.id)
        self.DataBase.ChangingSetting(data)

    def circle(self, type, count):
        pom = self.DataBase.GetPomodoro(self.id)
        if type == 'work':
            self.rest = restwindow(pom[0].get('pomodoro'))
            self.rest.show()
            self.rest.Pom.connect(self.pomidorChange)
        if type == 'rest':
            self.starttimer(count)
        if type == 'timepass':
            self.starttimer(0)

    def ChangingTable(self):
        for i in range(self.rows):
            self.ui.ShedulWidget.setItem(i, 1, QtWidgets.QTableWidgetItem(''))
            self.ui.ShedulWidget.setItem(i, 2, QtWidgets.QTableWidgetItem(''))
            if self.tm >= (str(int(i / 20)) + str(int(i/2) % 10) + ":00"):
                self.ui.ShedulWidget.setCurrentCell(i, 1)
        dtuple = QDate.getDate(self.ui.CalendarWidget.selectedDate())
        datastring = str(str(dtuple[0]) + '-' + str(dtuple[1]) + '-' + str(dtuple[2]))
        self.crit = self.DataBase.GetCrit(self.id)
        data = (self.id, datastring)
```

```
        geted = self.DataBase.GetShedul(data)
        for i in geted:
            SI = str(i.get('StartIvent'))
            if len(SI) == 7:
                SI = '0' + SI[:4]
            else:
                SI = SI[:5]
            EI = str(i.get('EndIvent'))
            if len(EI) == 7:
                EI = '0' + EI[:4]
            else:
                EI = EI[:5]
            desc = i.get('Description')
            typ = i.get('Type')
            s = 0
            for k in self.times:
                if k >= SI and k < EI:
                    self.insertintable(s, desc, typ)
                s += 1

    def insertintable(self, set, description, type):
        Str = ''
        for k in self.crit:
            if k.get('idtypes') == type:
                Str = k.get('name')
        self.ui.ShedulWidget.setItem(set, 1, QtWidgets.QTableWidgetItem(description))
        self.ui.ShedulWidget.setItem(set, 2, QtWidgets.QTableWidgetItem(Str))

    def GoalClick(self):
        self.GoalDialog = goalwindow(self.id,
QDate.getDate(self.ui.CalendarWidget.selectedDate()))
        self.GoalDialog.show()

    def RecordClick(self):
        self.RecordDialog = recordwindow(self.id,
QDate.getDate(self.ui.CalendarWidget.selectedDate()))
        self.RecordDialog.show()

    def SettingClick(self):
        self.SetingDialog = settingwindow(self.id)
        self.SetingDialog.show()

    def StrokClick(self):
        self.StrokDialog = quotwindow(self.timing)
        self.StrokDialog.show()
        self.StrokDialog.timing[int, bool].connect(self.Sroch)
```

```
def Sroch(self, timer, flag):
    self.timing = flag
    self.delay = timer - self.delay
    if self.timing:
        j = 0

def login(self):
    self.loginDialog = logwindow()
    self.loginDialog.show()
    self.loginDialog.hash[str, str].connect(self.check)

def check(self, hashline, info):
    self.DataBase = DB_working.DBwork()
    proof = self.DataBase.identity(hashline)
    if proof and info == 'Login':
        self.id = self.DataBase.getID(hashline)
        self.show()
    elif not proof and info == 'Registration':
        self.DataBase.id_input(hashline)
        self.id = self.DataBase.getID(hashline)
        criteriors = ['Работа', 'Учеба', 'Здоровье', 'Отдых', 'Саморазвитие', 'Работа по дому']
        self.TEST = testing(criteriors)
        self.TEST.show()
        self.TEST.criteriors[list, list].connect(self.critInput)
    elif info == 'Login':
        win.err('Неверное имя пользователя или пароль')
    else:
        win.err('Пользователь с таким именем уже существует')

def critInput(self, criteriors, line):
    data = []
    for i in range(len(line)):
        k = (self.id, criteriors[i], line[i])
        data.append(k)
    self.DataBase.CritInput(data)
    self.DataBase.pomodoroinput(self.id)
    self.show()

def err(self, line):
    self.mesDialog = messagewindow(line)
    self.mesDialog.show()
    self.mesDialog.hash[int].connect(self.login)

def DeleteClick(self):
    self.mesDialog = messagewindow('Удалить запись?')
```

```
        self.mesDialog.show()

app = QtWidgets.QApplication([])
win = mainwindow()
win.login()
sys.exit(app.exec())
```

ПРИЛОЖЕНИЕ В

Тест-кейсы

Таблица В.1 – Тест-кейс регистрации пользователя

Заголовок	Регистрация нового пользователя
Предусловие	Программа запущена. Открыто окно «Авторизация» с просьбой ввести Логин и Пароль
Шаг	**Ожидаемый результат**
Ввести имя пользователя состоящее из латинских букв и цифр в поле «Логин»	В поле «Логин» отображается введённое имя
Ввести случайный пароль состоящий из латинских букв и цифр в поле «Пароль»	В поле «Пароль» отображается введённый пароль
Нажать кнопку «Регистрация»	Появилось окно «Тестирование» с пояснительным текстом и двумя кнопками «Работа» (слева) и «Учеба» (справа). Окно «Авторизация» исчезло.
Нажать кнопку «Работа», находящуюся слева	Надписи на кнопках окна «Тестирование» изменились на «Работа» и «Здоровье».
Нажать кнопку «Здоровье», находящуюся справа	Надписи на кнопках окна «Тестирование» изменились на «Работа» и «Отдых».
Нажать кнопку «Отдых», находящуюся справа	Надписи на кнопках окна «Тестирование» изменились на «Работа» и «Саморазвитие».
Нажать кнопку «Работа», находящуюся слева	Надписи на кнопках окна «Тестирование» изменились на «Работа» и «Работа по дому».
Нажать кнопку «Работа», находящуюся слева	Надписи в окне «Тестирование» изменились на «Учеба» и «Саморазвитие».
Нажать кнопку «Учеба», находящуюся слева	Надписи в окне «Тестирование» изменились на «Учеба» и «Работа по дому».
Нажать кнопку «Учеба», находящуюся слева	Надписи в окне «Тестирование» изменились на «Здоровье» и «Отдых».
Нажать кнопку «Здоровье», находящуюся слева	Надписи в окне «Тестирование» изменились на «Саморазвитие» и «Работа по дому».
Нажать кнопку «Саморазвитие», находящуюся слева	Окно «Тестирование» исчезло, появилось главное окно программы с названием «Расписание».

Таблица В.2 – Тест-кейс функционала кнопок главного окна

Заголовок	Проверка функционала кнопок главного окна приложения
Предусловие	Программа запущена. Открыто главное окно программы с названием «Расписание».
Шаг	**Ожидаемый результат**
Нажать кнопку «Настройки» в правом верхнем углу главного окна	Появляется окно с заголовком «Настройки»
Закрыть окно «Настройки» нажав на кнопку закрытия окна в правом верхнем углу	Окно «Настройки» закрылось
Нажать кнопку «Срочное» в левом нижнем углу главного окна	Появляется окно с предупреждением о том, что все дела будут отложены и двумя кнопками «ОК» и «Cancel»
Нажать кнопку «Cancel»	Окно с предупреждением закрылось.
Выбрать случайную строку в таблице расписания в главном окне справа	Выбранная строка выделилась синим цветом
Нажать кнопку «Создать запись», находящуюся снизу справа в главном окне программы	Появилось окно с настройками новой записи.
Нажать кнопку «Cancel»	Окно с настройками новой записи закрылось.
Нажать кнопку «Создать новую цель», находящуюся снизу справа в главном окне программы	Появилось окно с настройками новой цели.
Нажать кнопку «Cancel»	Окно с настройками новой цели закрылось.