TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Informaatikainstituut

Tarkvaratehnika õppetool

Aljona Gvozdeva

093760IAPB

Isikliku aja korraldamise programmi väljatöötamine.

Bakalaureusetöö

Juhendaja: Kaarel Allik

Tallinn 2015

Autorideklaratsioon.

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

_____ _____

(kuupäev) (allkiri)

Isikliku aja korraldamise programmi väljatöötamine.

Annotatsioon.

Käesoleva diplomitöö eesmärk on „Time Control“ isikliku aja korraldamise programmi prototüübi väljatöötamine.

Käesolev prototüüp pakub funktsioonirida inimeste isikliku aja effektiivseks kontrollimiseks ja on mõeldud inimestele, kes kasutab tihti arvutit. See prototüüp on lihtne kasutamises ning omab vaistlikult arusaadava kasutajaliidesega. Töö pakub põhjalikku infosüsteemi kirjeldust ja ka selle elluviimist.

Lõputöö on kirjutatud vene keeles ning tema põhiosa sisaldab teksti 57 leheküljel, 23 joonist, 30 tabelit ning taotlus, mis sisaldab teksti 67 leheküljel

Development of personal time management program.

Abstract.

The main purpose of this thesis is prototype development of personal time management program called „Time Controll“.

This prototype offers a number of functions for effective controling of personal time for people, who often use their computers. It is easy to use and has user friendly interface. In this thesis will be covered the detailed description of infosystem and its realization.

The thesis is written in Russian language and contains 57 pages of the main part, 23 pictures, 30 tables and supplement of 67 pages.

Разработка программы для организации личного времени.

Аннотация.

Целью данной дипломной работы является разработка прототипа программы для организации личного времени «Time Control».

Данный прототип предлагает ряд функций для эффективного контроля личного времени людей, часто использующих компьютер. Он лёгок в использовании и обладает интуитивно понятным интерфейсом. В работе будет представлено детальное описание инфосистемы, а также её реализации.

Дипломная работа на степень бакалавра написана на русском языке и содержит 57 страниц основной части, 23 рисунка, 30 таблиц и приложение объёмом в 67 страниц.

Терминология.

Компиляция	трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду.
Байт	единица хранения и обработки цифровой информации; совокупность битов, обрабатываемая компьютером одномоментно.
Кроссплатформенность программного обеспечения	возможность запуска исполняемого файла на платформах различных ОС (операционных систем)
Java Virtual Machine (JVM)	**виртуальная машина Java** основная часть исполняющей системы Java, так называемой *Java Runtime Environment* (*JRE*).
Лямбда выражения	специальный синтаксис для объявления анонимных функторов по месту их использования.
Pomodoro	**Метод «Помидора»** техника управления временем, предложенная Франческо Чирилло в конце 1980-х.[1] Техника предполагает разбиение задач на 25-минутные периоды, называемые «помидоры», сопровождаемые короткими перерывами.

Учётная запись	хранимая в компьютерной системе совокупность данных о пользователе, необходимая для его аутентификации и предоставления доступа к его личным данным и настройкам.
Чекбокс	элемент графического пользовательского интерфейса, позволяющий пользователю управлять параметром с двумя состояниями — ☑ включено и ☐ выключено.
Прецедент	спецификация последовательностей действий в Унифицированном языке моделирования (UML), которые может осуществлять система, подсистема или класс, взаимодействуя с внешними действующими лицами.
Валидация	процесс приведения доказательств того, что требования конкретного внешнего потребителя или пользователя продукта, услуги или системы удовлетворены.
Аватар	графическое представление пользователя.
Маппинг	процесс составления схемы того, какими данными следует обмениваться и как они будут использоваться.
Android	операционная система для смартфонов, планшетных компьютеров, электронных книг и других устройств.
Синхронизация	процесс приведения к одному значению одного или нескольких параметров разных объектов.
Bluetooth	производственная спецификация беспроводных персональных сетей с радиусом действия 10 метров

Список рисунков:

Список таблиц:

Оглавление:

Введение

Предыстория и проблемы

Сегодня, благодаря развитию информационных технологий, компьютеры прочно вошли в нашу жизнь. Казалось бы, что повсеместная компьютеризация должна была бы привести к облегчению жизни человека, ведь теперь многие задачи автоматизированы и более не требуют долгого трудоёмкого решения. Однако вместо этого наш жизненный темп только увеличивается и чем он выше, тем больше различных задач нам необходимо уметь быстро выполнять, чтобы не выбиваться из общего ритма. Однако практика показывает, что умело распланировать своё время может далеко не каждый. Именно для того, чтобы помочь таким людям и был создан прототип программы *Time Control.*

Главная целевая аудитория программы – это люди, часто использующие компьютер, которые хотят более эффективно распоряжаться своим временем. Она предоставляет пользователю все функции, необходимые для успешного контролирования своего времени. Кроме того, она обладает приятным, интуитивно понятным интерфейсом, что позволяет быстро освоить её без особого труда. Одной из главных причин траты времени как правило становиться то, что пользователь часто во время игры просто не замечает, как быстро летит время. *Time Control* решает эту проблему: теперь пользователь сам заранее может решить сколько времени может проработать та или иная программа. Для людей, склонных к забывчивости, есть также функция напоминания, которая будет извещать пользователя о важных для него событиях. Для планирования расписания в программе предоставляется ежедневник, куда пользователь может занести все необходимые к выполнению задания. Задания будут выводится на главной странице программы по мере их необходимого времени выполнения, чтобы пользователь быстро и легко мог получать информацию о том, что и во сколько ему необходимо выполнить. Более того, для того, чтобы повысить эффективность работы пользователя для заданий была добавлена возможность создания таймеров. Каждый человек сам может выбрать какой из трёх представляемых программой таймеров ему больше подходит. Также в приложении имеется возможность для создания долгосрочных целей. Для этого пользователь просто вводит название цели и дату к которой она должна быть достигнута. Программа сама будет высчитывать сколько времени осталось до конца срока и выводить приоритетнее всех цели, для достижения которых осталось

менее всего времени. И наконец программа предоставляет пользователю полный отчет о работе его контролируемых программ и назначенных задач, для того, чтобы он мог проанализировать качество планирования своего времени.

В заключение хочу сказать, что на данный момент на рынке программных обеспечений предоставляется огромное количество программ для родительского контроля. Однако они больше нацелены на контроль времени использования отдельных учётных записей, полную блокировку программ или отдельных сайтов. В свою очередь программа *Time Control* предоставляет пользователю полный набор возможностей для успешного контроля своего времени, что делает её по-своему уникальной в своём роде.

Поставленные задачи.

В данной работе ставятся следующие цели:

- Создать функцию, которая могла бы помочь пользователю контролировать время, которое он тратит на те, или иные программы, путём установления ограничений на время использования.
- Предоставить удобный интерфейс для быстрого планирования целей пользователя.
- Дать возможность пользователю создавать напоминания о важных для него событиях.
- Вести статистику о контролируемых программах и выполненных задачах для того, чтобы дать пользователю возможность проанализировать свои действия и более эффективно распределять своё время.

Методика.

Выбор языка программирования.

Для разработки данного прототипа был выбран язык Java, ввиду его некоторых преимуществ.

В первую очередь это кроссплатформенность: одно и то же приложение можно запускать под управлением разных операционных систем. Для достижения этого программа компилируется в байт код, чей формат является независимым от платформы. Затем, содержащаяся в операционной системе программа под названием «Виртуальная Машина Java» (*JVM – Java Virtual Machine*), преобразует этот байт код в исполняемый код с учётом

платформы, на которой запускается программа. Таким образом одно и то же приложение может быть запущено на любой операционной системе, будь то Windows, UNIX, Linux или Macintosh.

Кроме того, Java достаточно хорошо справляется с ошибками. Всего в этом языке существует три вида ошибок:

- Ошибки во время компиляции (*Compile-time error*)
- Ошибки выполнения (*Runtime error*)
- Логические ошибки (*Logic error*)

Ошибки во время компиляции выявляются достаточно легко. Они возникают из-за неправильного использования синтаксиса языка и сразу же отображаются в компиляторе. Программа с такой ошибкой не будет скомпилирована в байт код до тех пор, пока ошибка не будет исправлена.

Ошибки выполнения возникают во время использования программы. Как правило их причиной является создание невыполнимых условий. Когда возникает такая ошибка, программа сама может с ней справится. Java не допускает чрезмерное использование или потерю памяти за счет встроенного механизма по замещению или освобождению памяти, так называемому *«сбору мусора»*. Благодаря этому программист оказывается застрахован от неправильного использования памяти приложением.

Логические ошибки сложнее всего выявить. Они возникают в результате неправильного использования функции. Как правило, поиском этих ошибок занимаются *«тестеры»*, сравнивая реальное поведение программы с её ожидаемым поведением.

Благодаря такой системе выявления ошибок, можно свести к минимуму возможность появления неполадок в работе готового приложения.

Для упрощения кода в данном проекте были использованы лямбда выражения, которые стали доступны начиная с 8 версии Java.

Выбор графического интерфейса программирования.

В данной работе для разработки графического интерфейса пользователя используется JavaFX. Это достаточно молодая технология, на которую возлагают большие надежды. По задумке разработчиков JavaFX в будущем сможет затмить собой всем известную

библиотеку Swing. В ней возможно относительно легко создавать привлекательный для пользователя графический интерфейс, путём создания анимации, переходов и других эффектов. Кроме того, использование JavaFX позволяет хорошо организовать свой код, так как, главным образом, код разметки графического интерфейса пишется в FXML файлах, стили в CSS файлах, а все управляющие действия в Java классах.

Выбор базы данных.

Для хранения информации в данной работе была использована бесплатная встроенная база данных SQLite.

Выпущенная в 2000 году, эта база данных достаточно хорошо зарекомендовала себя за счет своей производительности, компактности и высокой ёмкости. Большим преимуществом встроенной базы данных является то, что для работы программы нет необходимости подключаться к Интернету. Пользователь может быть уверен, что введенные им данные никем не будут использованы, так как вся информация будет храниться на его компьютере, непосредственно на локальном сервере базы данных внутри самой программы.

1. Стратегический этап.

1.1. Общий обзор системы.

Далее будет представлен краткий обзор программы для организации личного времени *Time Control.*

1.1.1. Цели организации.

Главной целью создания прототипа программы *Time Control* является предоставление пользователю возможности контролировать время работы своих программ, учит его анализировать свою деятельность и распределять задачи с целью достижения максимальной производительности.

1.1.2 Цели инфосистемы.

- Предоставлять обзор всех контролируемых программ.
- Предоставлять обзор всех запланированных ежедневных задач.
- Предоставлять обзор всех выполненных ежедневных задач.
- Предоставлять обзор всех частично выполненных ежедневных задач.
- Предоставлять обзор всех невыполненных ежедневных задач.
- Предоставлять обзор всех запланированных долгосрочных целей.
- Предоставлять обзор всех выполненных долгосрочных целей.
- Предоставлять обзор всех невыполненных долгосрочных целей.
- Предоставлять обзор статистики работы контролируемых программ.
- Предоставлять обзор статистики ежедневных задач.
- Предоставлять обзор статистики долгосрочных целей.
- Предоставлять пользователю возможность создавать новые правила контроля для своих программ.
- Предоставлять пользователю возможность ограничивать доступ для других пользователей к настройкам контроля своих программ.
- Предоставлять пользователю возможность изменять время работы программ.

- Предупреждать пользователя о скором прекращении работы программы.
- Предоставлять пользователю возможность создавать новые напоминания.
- Предоставлять пользователю возможность создавать новые долгосрочные цели.
- Предоставлять пользователю возможность создавать новые ежедневные задания.

1.1.3. Предложения.

- Пользователь - это личность.
- Пользователь может создавать правила контроля программ.
- Пользователь может изменять правила контроля программ.
- Пользователь может создавать правила контроля программ с ограниченным доступом.
- Пользователь не может изменять чужие правила контроля программ с ограниченным доступом.
- Для каждой программы не может быть создано более одного постоянного правила контроля.
- Однодневные правила контроля имеют больший приоритет, чем постоянные правила контроля.
- Однодневные правила контроля не могут быть созданы для программ с ограниченным доступом.
- В программе могут быть зарегистрированы несколько пользователей.
- Правила контроля программ распространяются на всех пользователей.
- Пользователь может удалять правила контроля программ.

1.1.4. Главные объекты.

- Пользователь.
- Программа.
- Правило контроля.
- Статистика.
- Ежедневник.
- Напоминание.
- Время работы.
- Таймер.

- Сеть таймеров.
- Pomodoro.
- Прогресс.

1.1.5. Главные процессы.

- Вход в учётную запись пользователя.
- Регистрация учётной записи пользователя.
- Создание ежедневного правила контроля программы.
- Создание однодневного правила контроля программы.
- Создание ежедневного правила контроля программы с ограниченным доступом.
- Продление времени работы программы.
- Изменение настроек правила контроля работы программы.
- Удаление правила контроля программы.
- Создание новой долгосрочной цели.
- Создание нового задания.
- Создание нового ежедневного задания.
- Создание нового таймера.
- Создание новой сети таймеров.
- Создание нового таймера Pomodoro.
- Изменение состояния задания.
- Запуск таймера.
- Запуск сети таймеров.
- Запуск таймера Pomodoro.
- Создание нового напоминания.
- Перенесение задач на новый день.
- Удаление задач.
- Удаление напоминания.
- Просмотр статистики контроля программ.
- Просмотр статистики своих заданий.
- Фиксирование своего процесса работы.
- Фиксирование своего результата работы.
- Восстановление забытого пароля.
- Смена пароля.

- Смена пользователя.

1.1.6. Главные события.

- Регистрация нового пользователя.
- Создание нового правила контроля программы.
- Создание нового напоминания.
- Создание новой задачи.
- После достижения максимально разрешенного времени работы, программа выключается.
- За установленное пользователем время до конца работы программы, появляется предупреждение о скором отключении программы.
- *Time Control* блокирует включение программы, которая достигла своего суточного лимита работы и извещает об этом пользователя.
- Пользователь добавляет время работы программы.
- Пользователь удаляет правило контроля программы.

1.1.7. Действующие лица.

- Пользователь.

1.1.8. Местоположение.

- У каждого пользователя есть свой личный кабинет, откуда он может управлять своими правилами контроля программ, задачами, напоминаниями и статистикой.

1.1.9. Разделение целой системы на подсистемы.

Действие происходит с системой программы *Time Control.*

1.1.9.1. Функциональные подсистемы.

Основные подсистемы:

- Функциональная подсистема пользователей.
- Функциональная подсистема статистического отчёта.
- Функциональная подсистема напоминаний.

- Функциональная подсистема программ.
- Функциональная подсистема задач.

Административные подсистемы:

- Функциональная подсистема классификатора.

1.1.9.2. Регистры.

Основные регистры:

- Регистр пользователей.
- Регистр контроля программ.
- Регистр напоминаний.
- Регистр задач.
- Регистр вопросов.
- Регистр дней работы.

Административные регистры:

- Регистр классификатора.

1.1.10. Бизнес правила

- У каждого пользователя есть своя учётная запись.
- У каждого пользователя может быть ноль или более правил контроля программ.
- У каждого пользователя может быть ноль или более задач.
- У каждого пользователя может быть ноль или более напоминаний.
- У каждой задачи может быть ноль или более таймеров.
- Если пользователь забыл свой пароль, то он может восстановить его, ответив на контрольный вопрос или отправив себе на электронную почту письмо с паролем.
- Задачи имеют состояние «не выполнено» по умолчанию, пока пользователь не изменит их состояние.
- При контроле программы будет приниматься во внимание правило с более высоким приоритетом.
- У каждого правила контроля может быть ноль или одно предупреждение об окончании работы.
- У каждой задачи есть три вида состояния.

- Если пользователь не успел выполнить поставленную задачу, он может перенести её на другой день.
- Нельзя создать правило контроля для программы, на которую уже установлено правило с наивысшим приоритетом.

1.1.11. Диаграмма активности главного процесса.

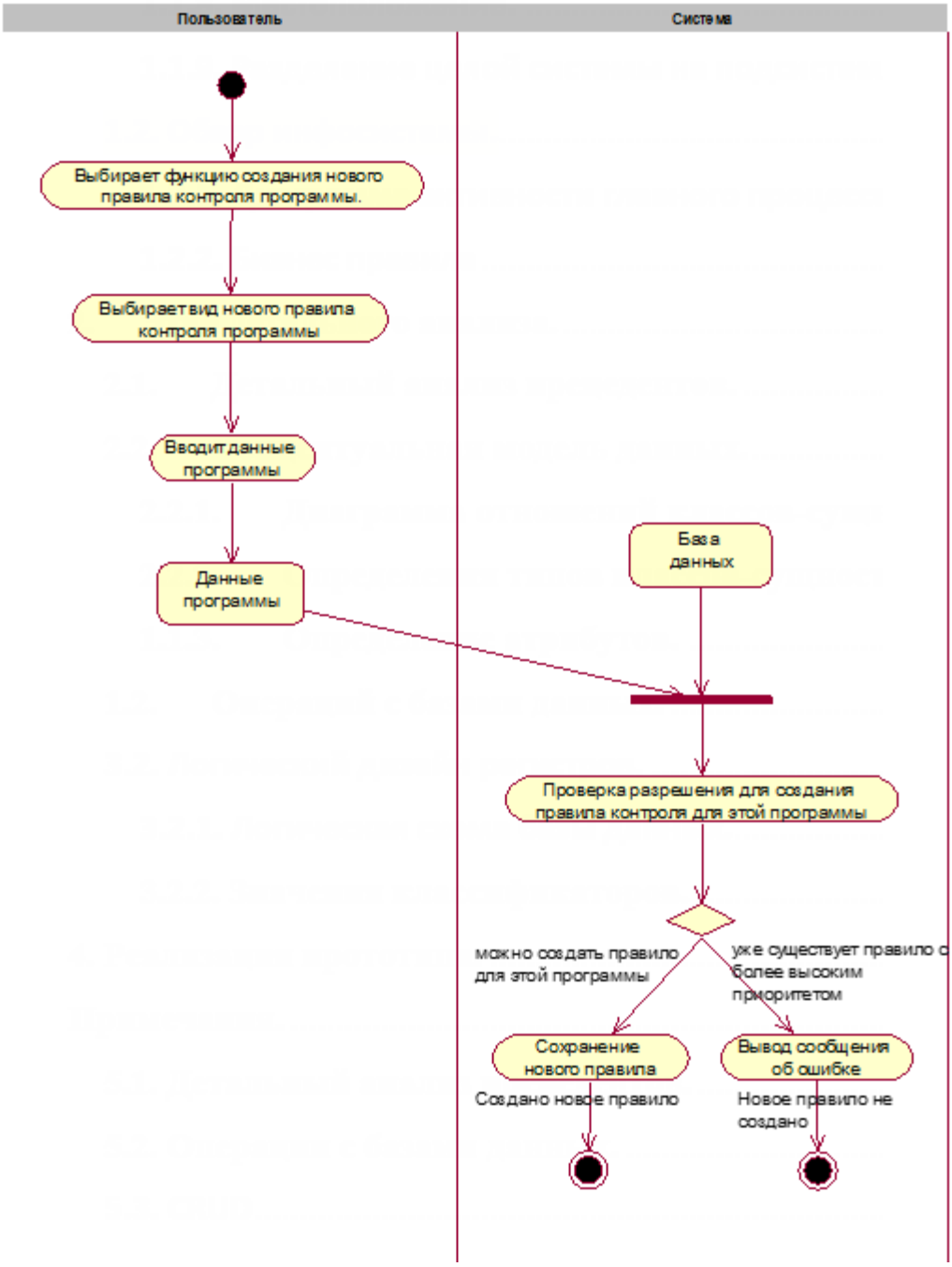

Рисунок 1. Диаграмма активности создания правила контроля работы программы.

2. Этап детального анализа.

2.1. Детальный анализ прецедентов.

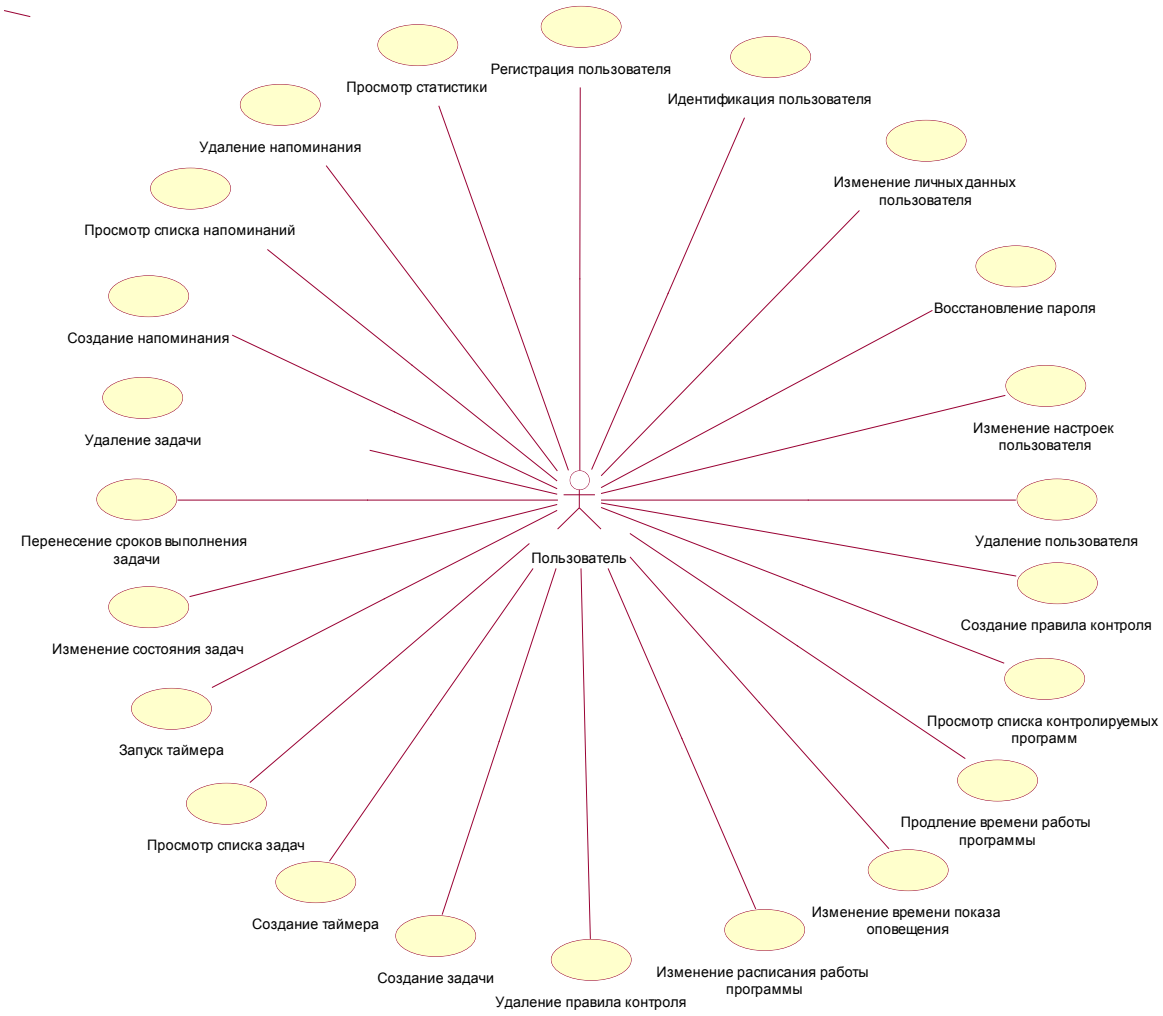

Рисунок 2. Диаграмма прецедентов.

Детальное описание прецедентов см. в приложении 5.1.

2.2. Концептуальная модель данных.

2.2.1. Диаграмма отношений классов-сущностей.

Светло-жёлтым окрашены объекты, относящиеся к *регистру пользователей*.

Голубым окрашены объекты, относящиеся к *регистру задач*.

Оранжевым окрашены объекты, относящиеся к *регистру напоминаний*.

Красным окрашены объекты, относящиеся к *регистру контроля программ*.

Зелёным окрашены объекты, относящиеся к *регистру классификатора*.

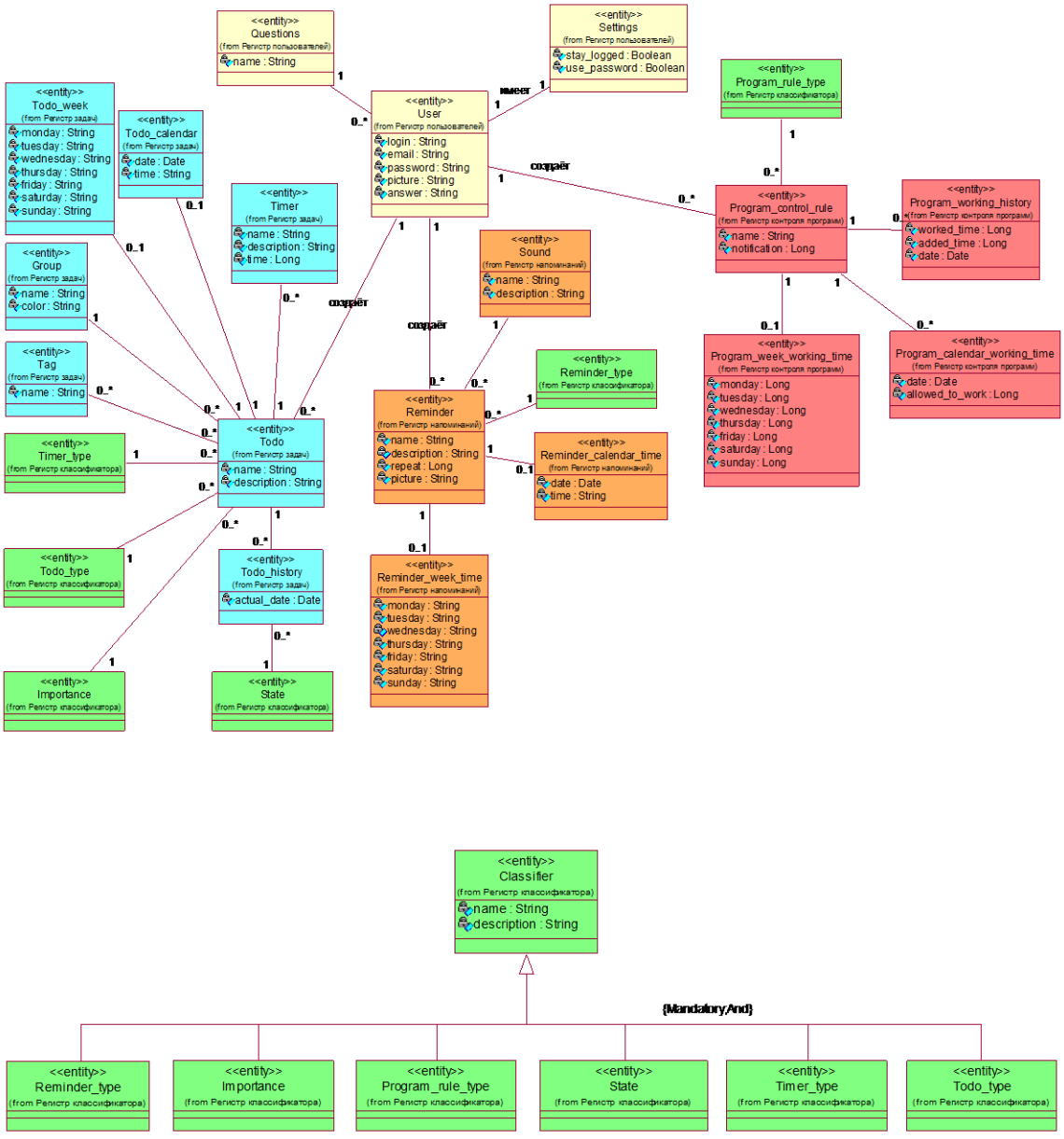

Рисунок 3. Расширенная диаграмма отношений классов-сущностей.

2.2.2. Определения классов-сущностей.

Название класса-сущности	*Принадлежность к регистру*	*Определение*
User	Регистр пользователей	«Пользователь - лицо или организация, которое использует действующую систему для выполнения конкретной функции». («Процессы жизненного цикла программных средств», 2000)
Settings	Регистр пользователей	Место, где хранятся настройки пользователя, которые отвечают за личные предпочтения пользователя в работе программы.
Questions	Регистр пользователей	Вопросы для самоконтроля пользователя, они необходимы если пользователь хочет восстановить свой пароль.
Group	Регистр задач	Группы для распределения задач пользователя по категориям, для более удобного управления задачами.
Timer	Регистр задач	Таймер – устройство, которое отмеряет заданный интервал времени, при помощи обратного отсчёта, после его запуска.
Timer_type	Регистр классификатора	Определяет вид используемого таймера. Например: простой таймер, сеть таймеров или таймер Pomodoro.

Название класса-сущности	*Принадлежность к регистру*	*Определение*
State	Регистр классификатора	Состояние определяет степень завершенности задачи. Всего у задачи могут быть 3 состояния: не выполнено, частично выполнено, выполнено.
Todo_type	Регистр классификатора	Определяет тип зарегистрированной задачи. Всего 3 типа задач: ежедневная, однодневная и долгосрочная.
Importance	Регистр классификатора	Степень демонстрирующая важность задания.
Tag	Регистр задач	Ключевое слово для тематического группирования информации.
Todo	Регистр задач	Задачи, которые пользователь запланировал для себя сделать.
Todo_history	Регистр задач	Используется статистикой для отображения качества работы пользователя. Хранит в себе даты всех запланированных задач и ссылки на их состояние (выполнено, частично выполнено, не выполнено).
Todo_calendar	Регистр задач	Хранит в себе даты и время запланированных на будущее задач. Используется для одноразовых задач.
Todo_week	Регистр задач	Хранит в себе время запланированных на будущее задач, распределённое по дням недели. Используется для ежедневных задач.

Название класса-сущности	*Принадлежность к регистру*	*Определение*
Reminder	Регистр напоминаний	Хранит данные напоминаний о важных для пользователя событий.
Reminder_type	Регистр классификатора	Демонстрирует к какому типу относится выбранное напоминание. Всего может быть 2 типа напоминаний: ежедневое напоминание и одноразовое напоминание.
Reminder_calendar_time	Регистр напоминаний	Хранит в себе даты и время созданных напоминаний. Используется для одноразовых напоминаний.
Sound	Регистр напоминаний	Содержит список названий возможных сигналов напоминаний.
Reminder_week_time	Регистр напоминаний	Хранит в себе время созданных напоминаний, распределённое по дням недели. Используется для постоянных напоминаний.
Program_control_rule	Регистр контроля программ	Хранит наименования программ, для которых было создано правило контроля.
Program_rule_type	Регистр классификатора	Демонстрирует к какому типу контроля относится выбранное правило. Всего может быть 3 типа контроля: ежедневный контроль, одноразовый контроль и родительский контроль.
Program_working_history	Регистр контроля программ	Хранит в себе полную историю времени работы контролируемой программы.

Название класса-сущности	*Принадлежность к регистру*	*Определение*
Program_week _working_time	Регистр контроля программ	Хранит в себе расписание работы программы на неделю. Используется при постоянном контроле.
Program_calendar _working_time	Регистр дней работы	Хранит в себе даты и соответствующие им ограничения работы программы. Используется при одноразовом контроле.
Classifier	Регистр классификатора	«Классификатор описывает предопределённый набор данных или концепций.» («Data Mining 3rd Ed.», 2012)

Таблица 1. Определения классов-сущностей.

2.2.3. Определение атрибутов.

См приложение 5.4.

2.3. Операций с базами данных.

См. приложение 5.2.

2.4. Диаграмма состояния регистра задач.

Главный объект регистра – Задача (Todo).

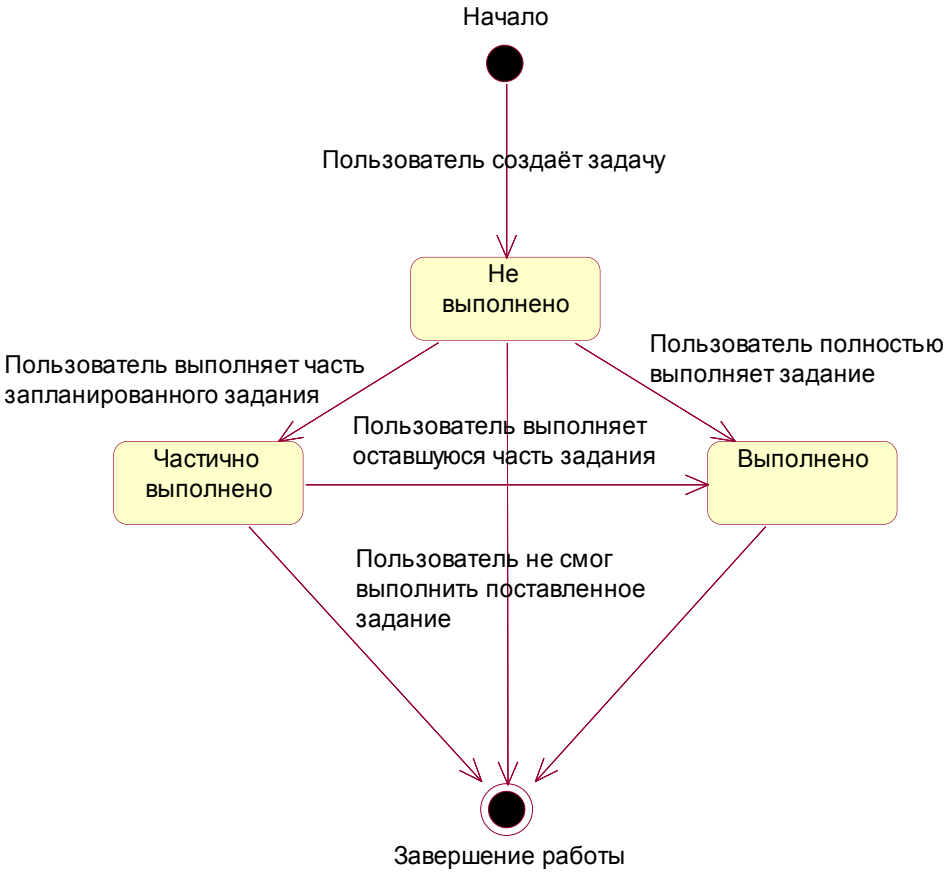

Рисунок 4. Диаграмма состояния задачи.

2.5. Таблица CRUD

См. приложение 5.3.

3. Логический дизайн.

3.1. Реализация системы.

3.1.1. Место реализации.

Личный кабинет пользователя.

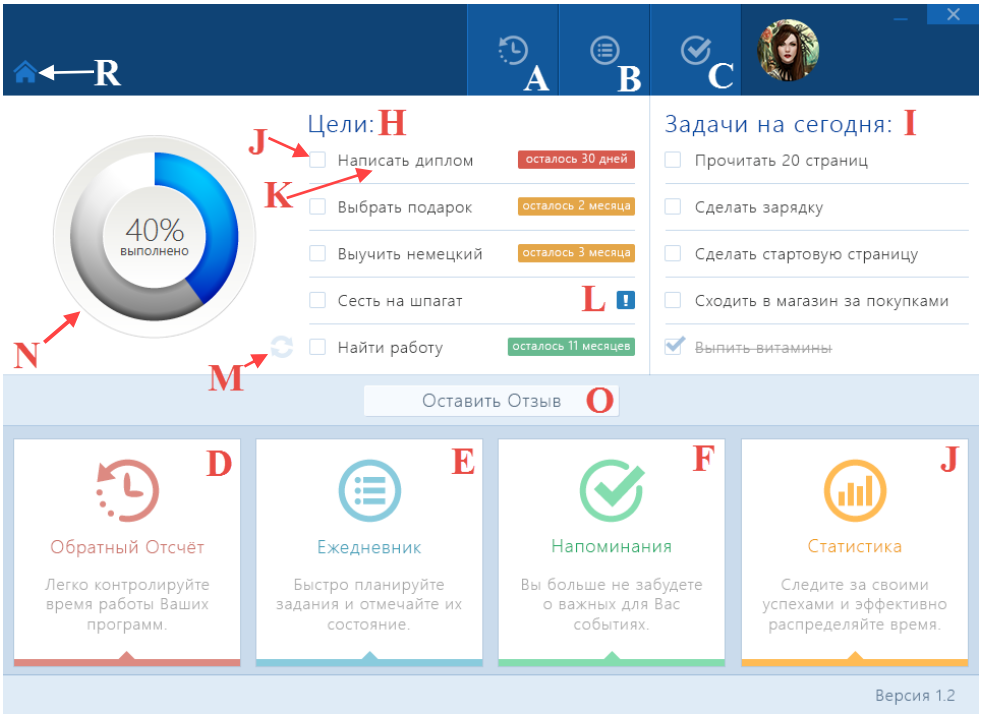

Рисунок 5. Главный экран прототипа.

Список объектов главной страницы:

A – Функция отображения всех созданных правил контроля.

B – Функция отображения всех созданных задач.

C – Функция отображения всех созданных напоминаний.

D – Функция создания нового правила контроля программы.

E – Функция создания новой задачи.

F – Функция создания нового напоминания.

G – Функция отображения статистики.

H – Краткий список долгосрочных целей.

I – Краткий список задач на текущий день.

J – Чекбокс для быстрой смены состояния задачи. (Возможные значения: выполнено, не выполнено)

K – Названия задачи, кликнув на которое пользователь переходит на страницу полной информации о данной задаче.

L – Показатель оставшегося времени до истечении срока долгосрочной задачи.

M – Кнопка обновления списков.

N – Прогресс, показывает процент выполненных задач, запланированных на текущий день.

O – Кнопка для перехода на экран обратной связи.

R – Кнопка возврата на главную страницу.

3.1.2. Описание реализации прецедентов.

Далее представлена реализация 5 самых важных прецедентов:

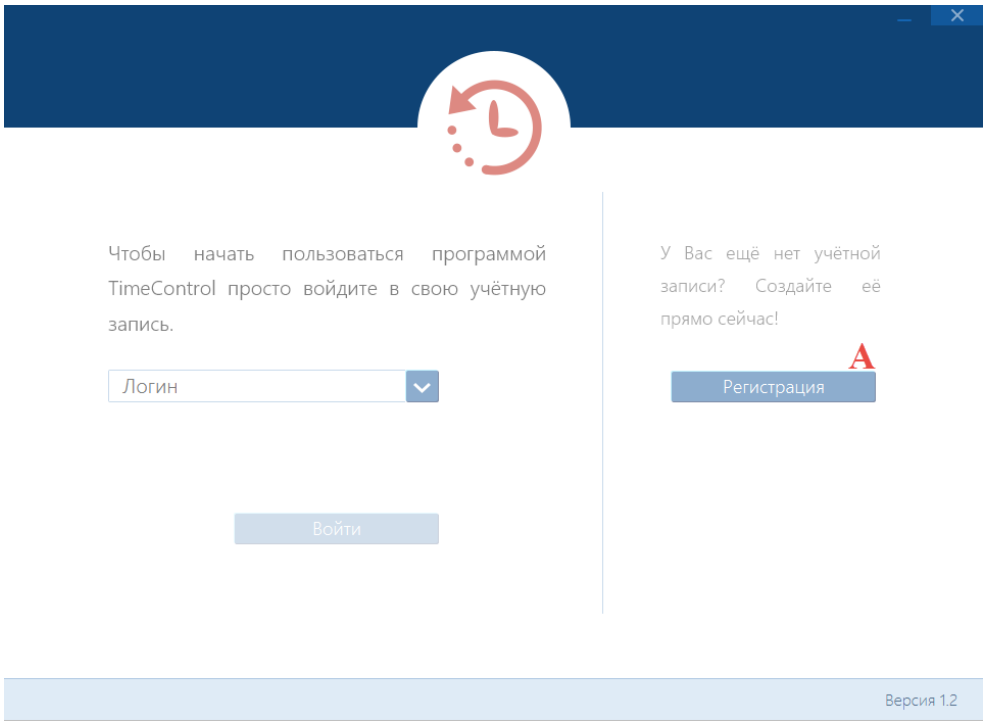

Рисунок 6. Регистрация пользователя. Часть 1.

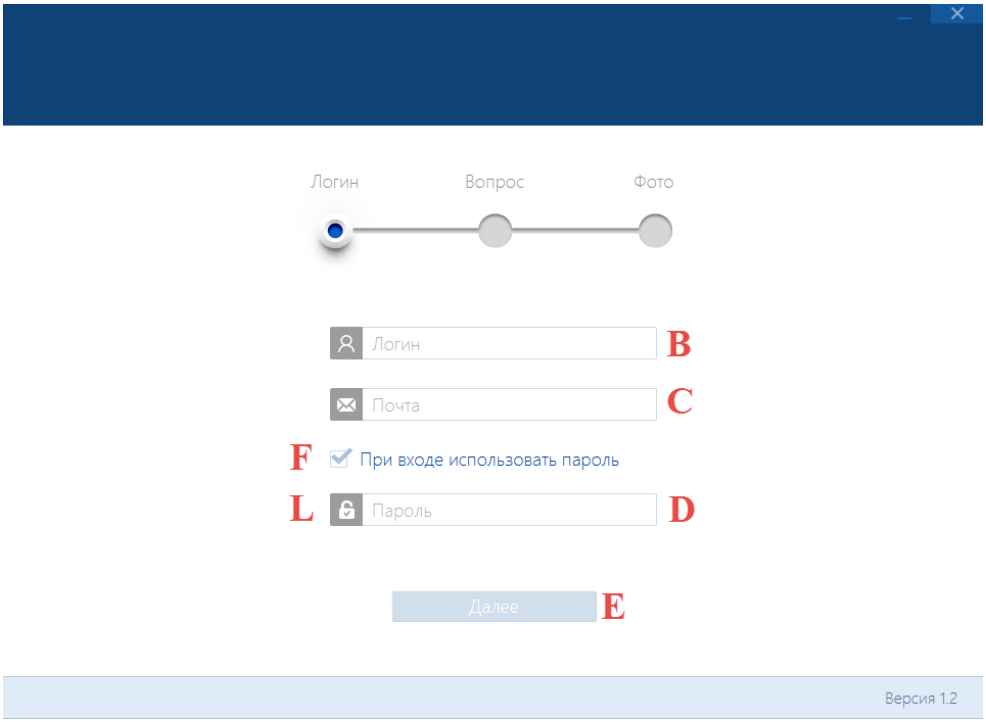

Рисунок 7. Регистрация пользователя. Часть 2.

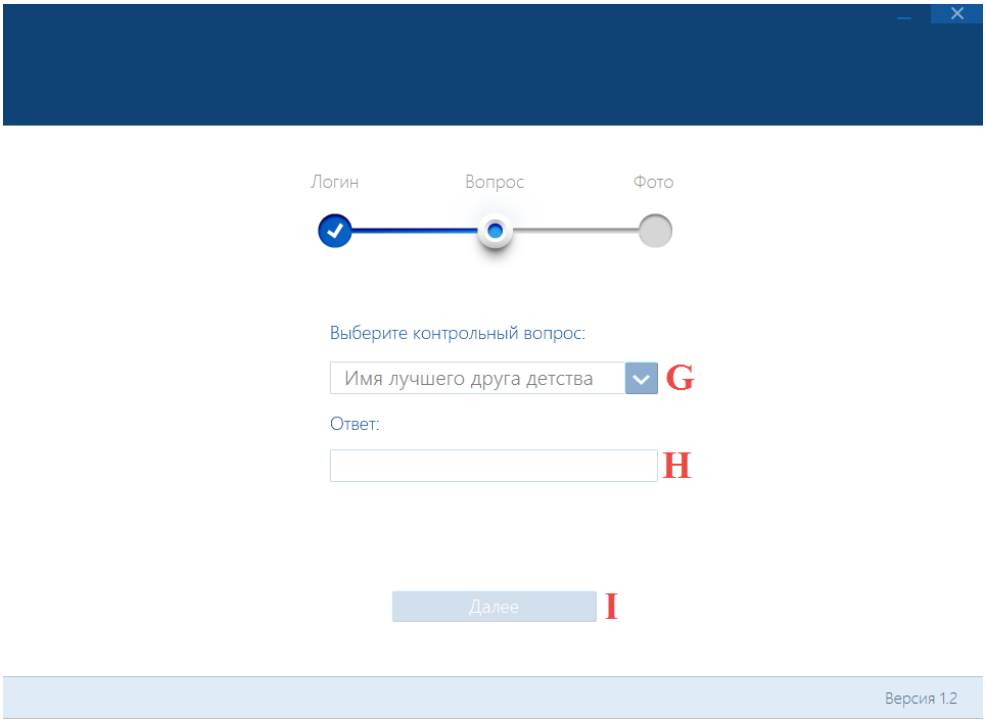

Рисунок 8. Регистрация пользователя. Часть 3.

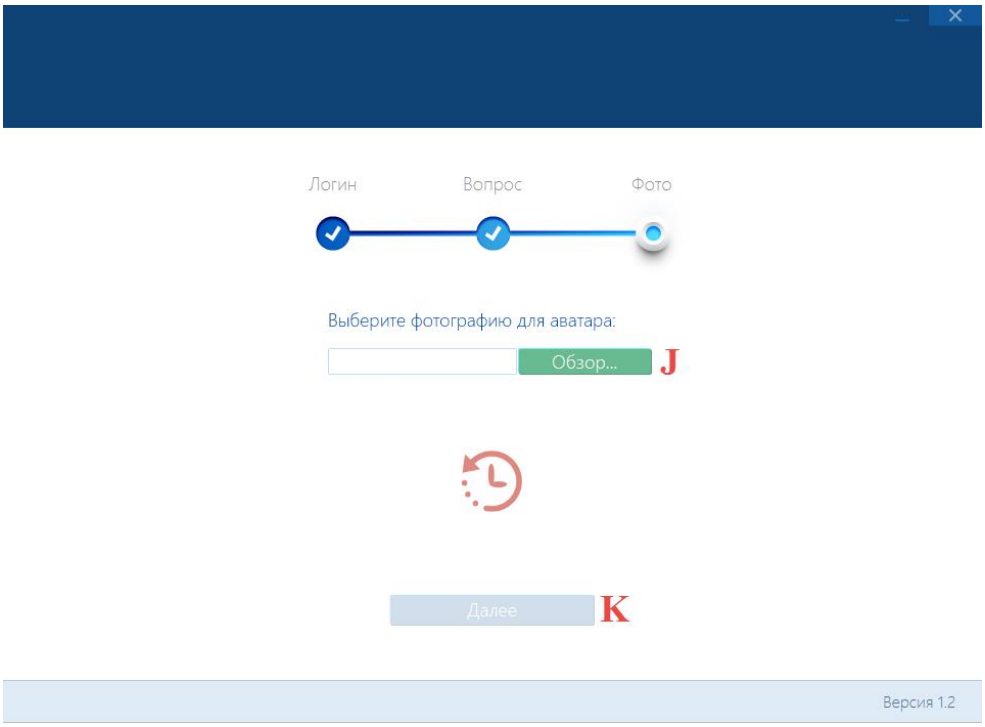

Рисунок 9. Регистрация пользователя. Часть 4.

Прецедент: Регистрация пользователя.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Желает создать свою личную учётную запись для того, чтобы иметь доступ ко всем функциям системы.

Происходящее действие: Пользователь хочет зарегистрироваться в системе.

Предварительные требования: Программа должна быть запущена.

Результат: Данные пользователя сохраняются в базе данных и для него создаётся соответствующая учётная запись. Теперь ему доступны все функции программы.

Сценарий (типичное развитие событий):

1. Пользователь желает зарегистрироваться. На главной странице он нажимает кнопку «Регистрация» (Рисунок 5. Пункт A).
2. Пользователь вводит свой логин и электронную почту (Рисунок 6. Пункт B, C). Если пользователь желает использовать пароль при входе в систему, то он отмечает пункт «При входе использовать пароль» (Рисунок 6. Пункт F). В появившемся поле (Рисунок 6. Пункт D) он вводит свой пароль. Чтобы проверить правильность введённого пароля пользователь может нажать на иконку пароля (Рисунок 6. Пункт L), текст поля к которому она принадлежит станет виден на время её удержания.

Убедившись в правильности введённых данных, пользователь нажимает на кнопку «Далее» (Рисунок 6. Пункт E).

Примечание: Валидация всей вводимой во время регистрации информации происходит в фоновом режиме, как только поле ввода выходит из фокуса. Если данные введены верно, то появляется значок ✔, если же данные не прошли валидацию, то рядом с полем появляются значок ⊘ и сообщение поясняющее причину ошибки. Пока вся необходимая для ввода информация не пройдёт удачно проверку, кнопка «Далее» будет заблокирована.

3. Пользователь выбирает контрольный вопрос (Рисунок 7. Пункт G) и вводит на него ответ (Рисунок 7. Пункт H). Пользователь нажимает на кнопку «Далее» (Рисунок 7. Пункт I).
4. Пользователь выбирает изображение для своего аватара (Рисунок 8. Пункт J) и нажимает на кнопку «Далее» (Рисунок 8. Пункт K).
5. Система сохраняет данные пользователя. **(Операция 1.1)**

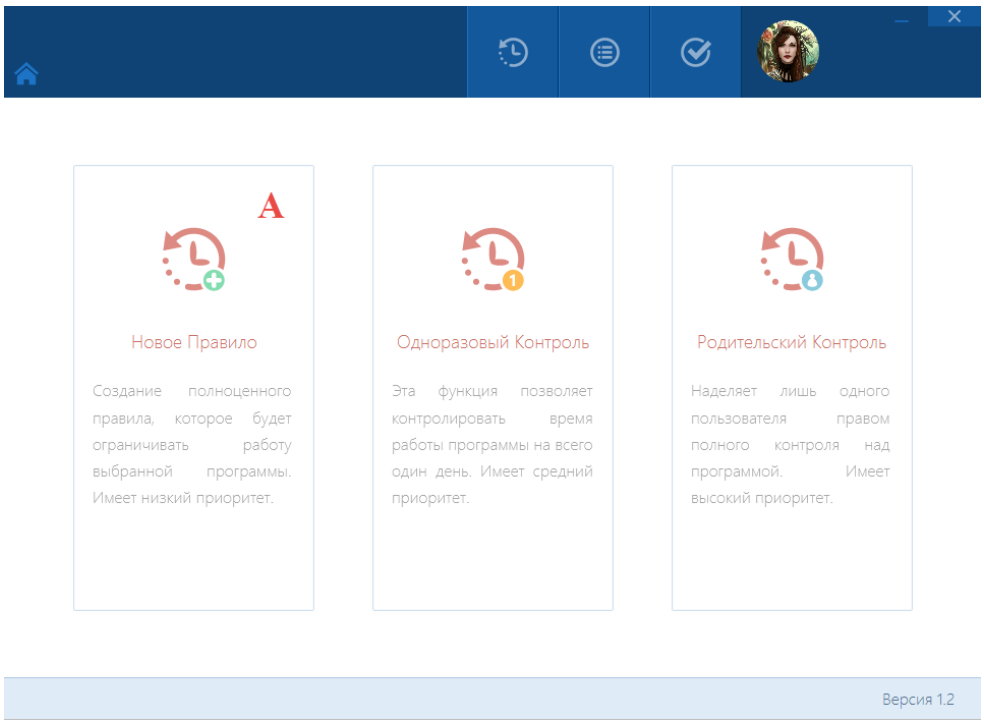

Рисунок 10. Экран выбора вида нового правила контроля.

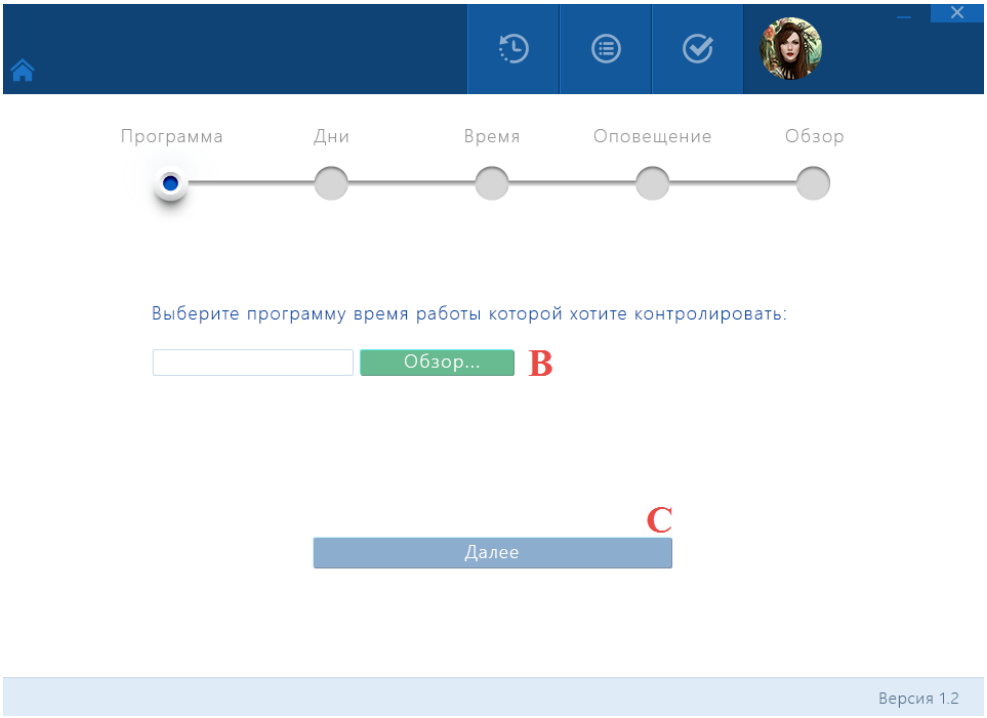

Рисунок 11. Создание нового правила контроля. Часть 1.

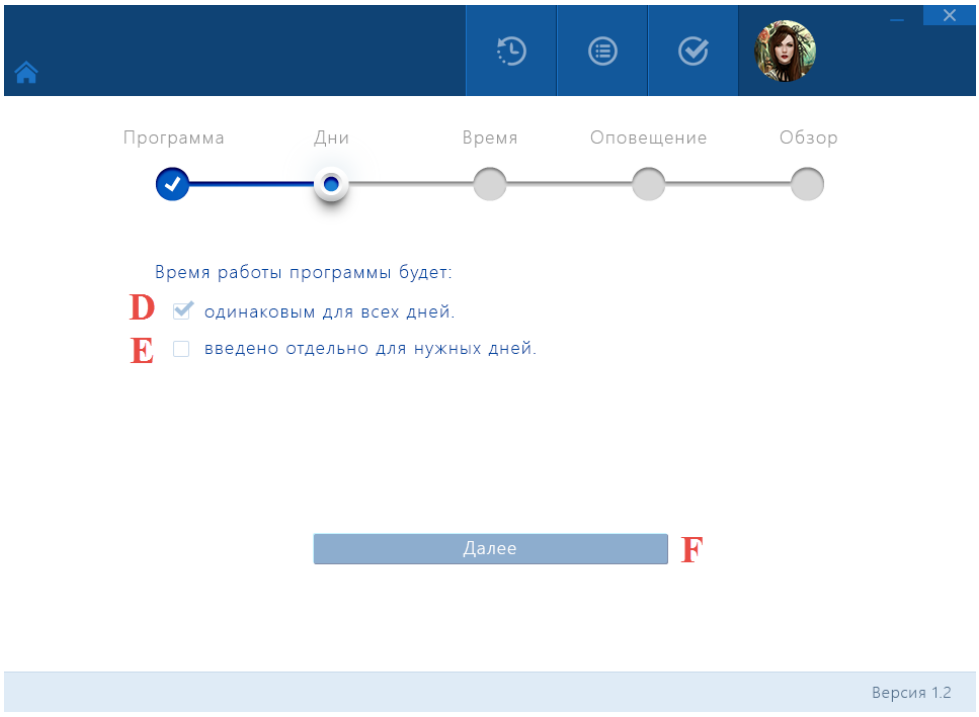

Рисунок 12. Создание нового правила контроля. Часть 2.

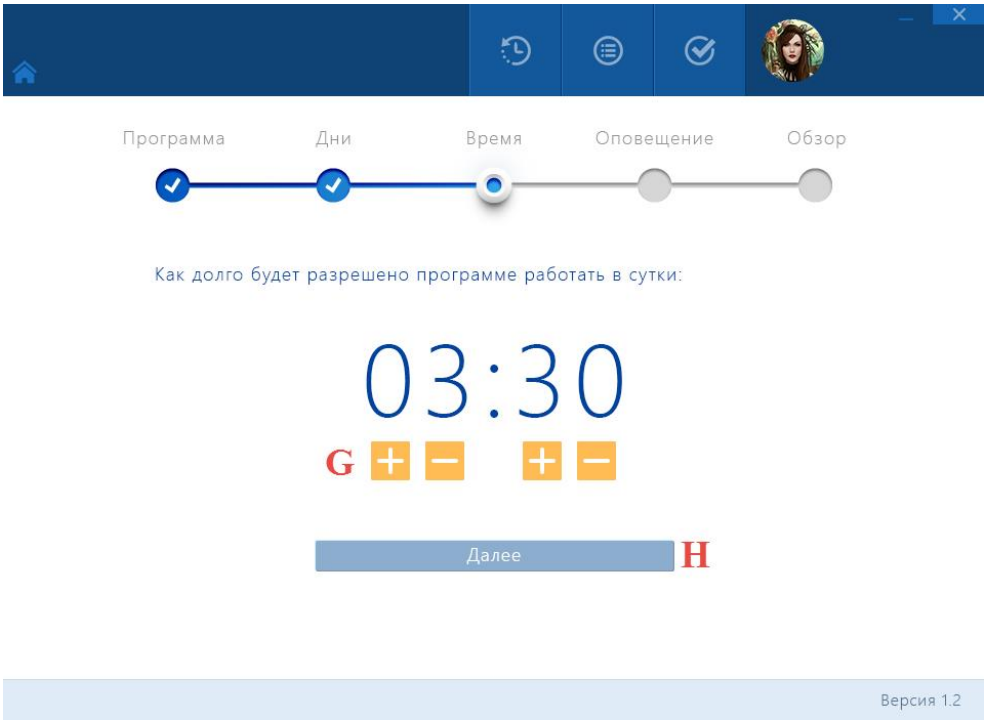

Рисунок 13. Создание нового правила контроля. Часть 3.

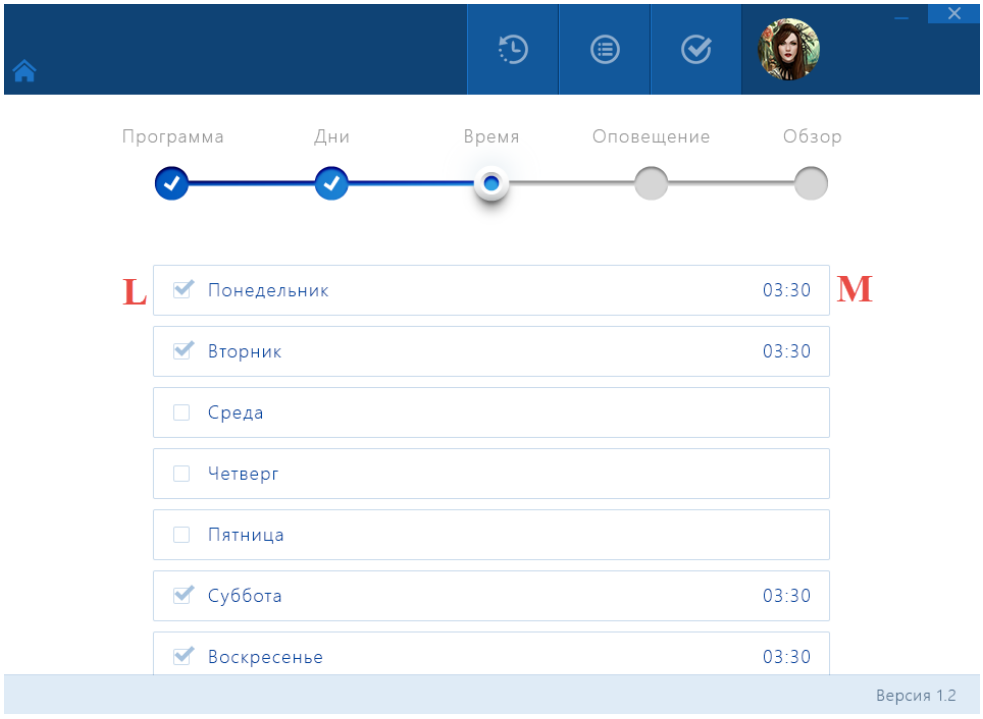

Рисунок 14. Создание нового правила контроля. Часть 3а.

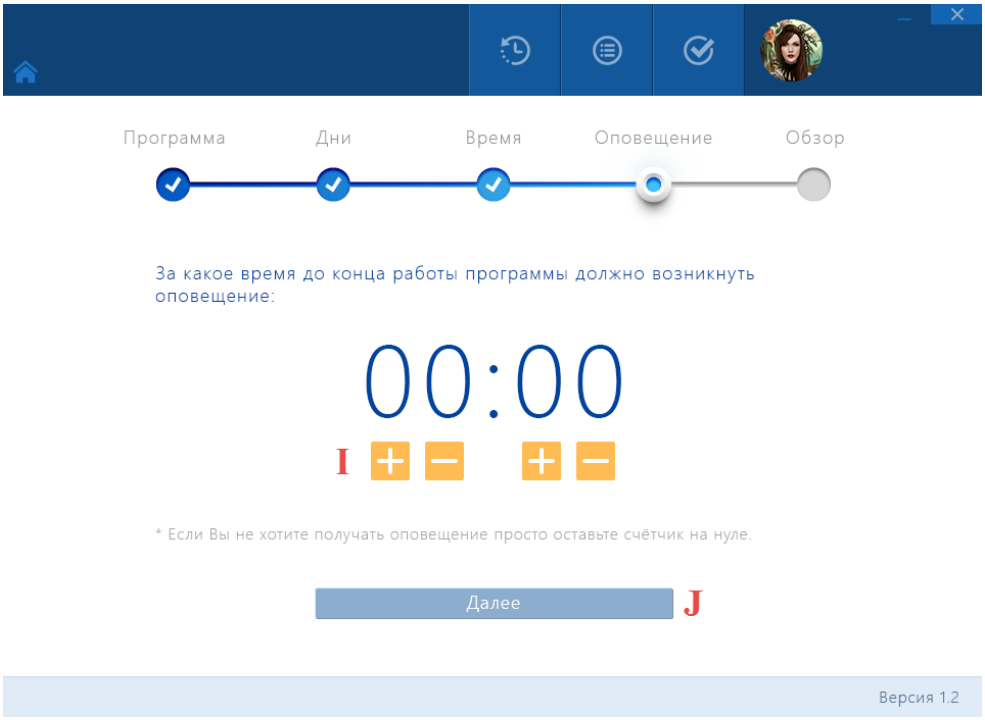

Рисунок 15. Создание нового правила контроля. Часть 4.

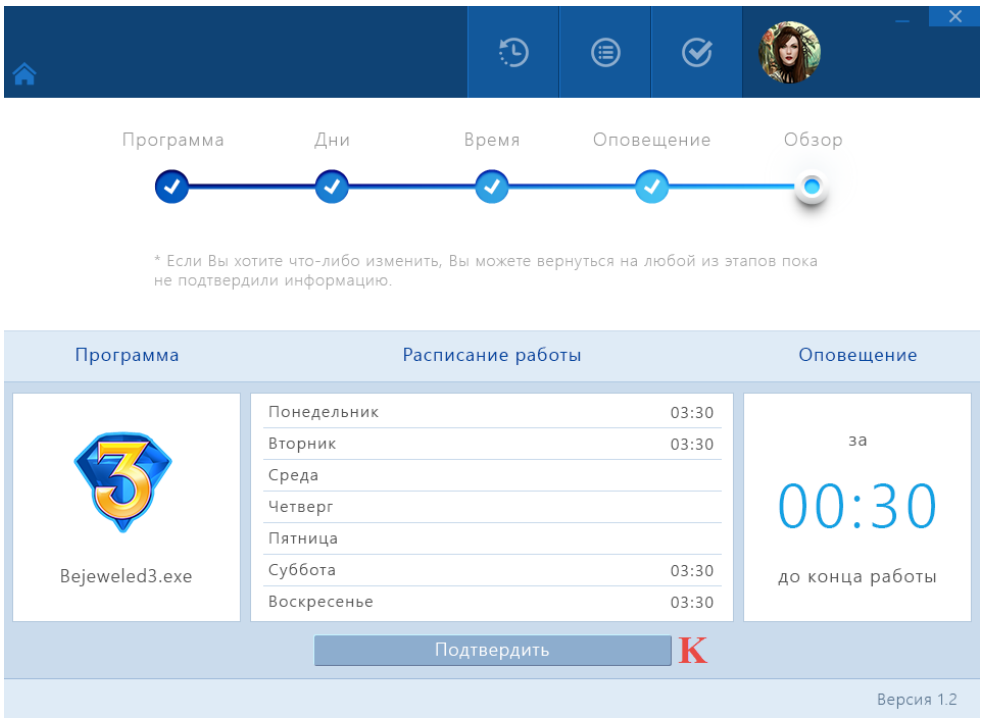

Рисунок 16. Создание нового правила контроля. Часть 5.

Прецедент: Создание правила контроля.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Желает контролировать время работы своей программы.

Происходящее действие: Пользователь хочет создать новое правило контроля.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Создано новое правило контроля времени работы программы.

Сценарий (типичное развитие событий):

1. Пользователь выбирает функцию создания нового правила контроля программы (Рисунок 9. Пункт A).
2. Пользователь вводит расположение программы (Рисунок 10. Пункт B) и нажимает на кнопку «Далее» (Рисунок 10. Пункт C).
3. Система проверяет можно ли создать данное правило контроля для этой программы.
4. Проверка успешно пройдена. Система генерирует следующий экран.
5. Пользователь выбирает как он желает ввести время работы программы: одинаково для всех дней (Рисунок 11. Пункт D) или отдельно для нужных дней (Рисунок 11. Пункт E), и нажимает на кнопку «Далее» (Рисунок 11. Пункт F).
6. Пользователь вводит как долго будет разрешено программе работать (Рисунок 12. Пункт G) и нажимает на кнопку «Далее» (Рисунок 12. Пункт H).
7. Пользователь вводит за сколько до конца работы программы должно появиться оповещение (Рисунок 14. Пункт I) и нажимает на кнопку «Далее» (Рисунок 14. Пункт J).
8. Пользователь проверяет введённые им данные и нажимает кнопку «Подтвердить» (Рисунок 15. Пункт K).
9. Система проверяет возможно ли создать данное правило контроля для этой программы. **(Операция 7.1)**
10. Программа сохраняет новое правило контроля. **(Операция 7.2)**

Дополнения (альтернативное развитие событий):

4a. Для данной программы уже создано правило с высшим приоритетом. Система выводит сообщение об ошибке и не позволяет сохранить новое правило контроля.

6a. Пользователь вводит сколько и по каким дням программа должна работать (Рисунок 13. Пункт L, пункт M).

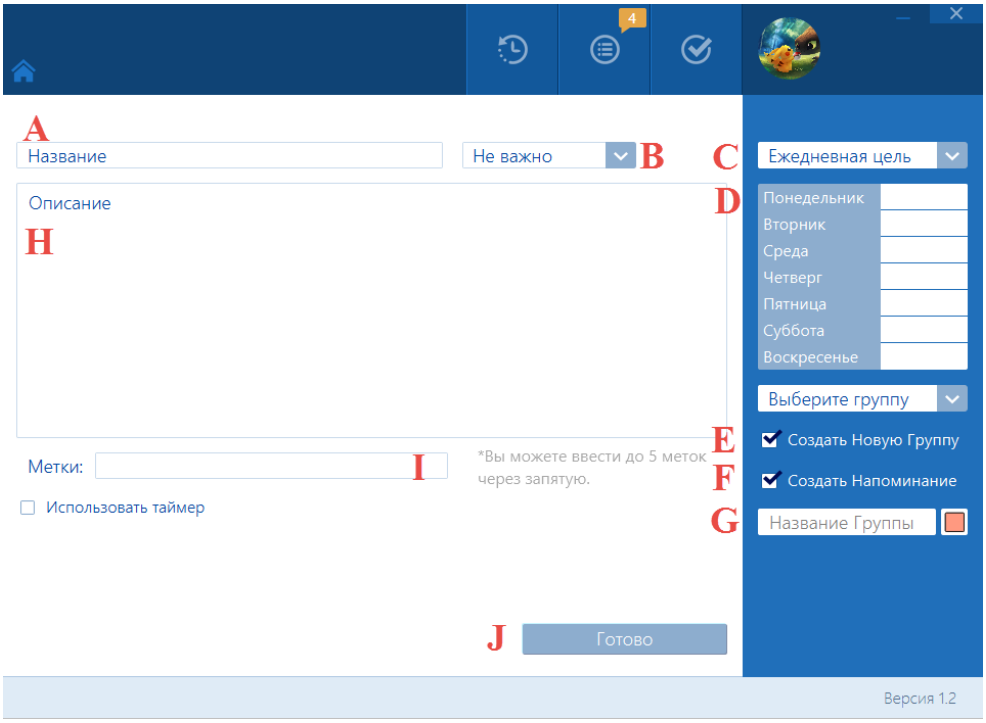

Рисунок 17. Создание новой ежедневной задачи.

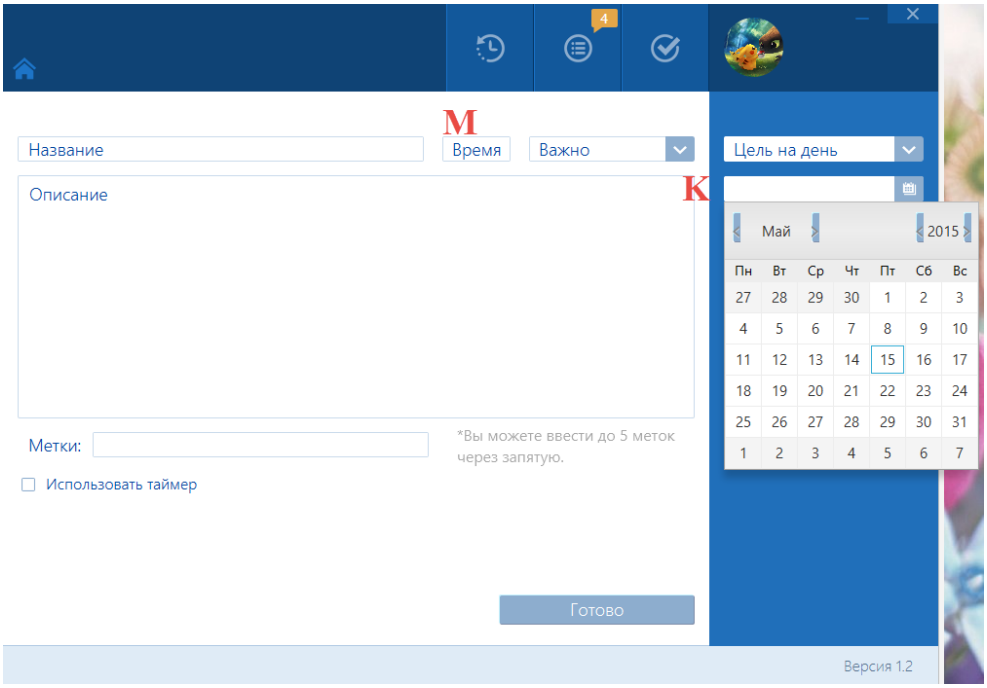

Рисунок 18. Создание новой цели на день.

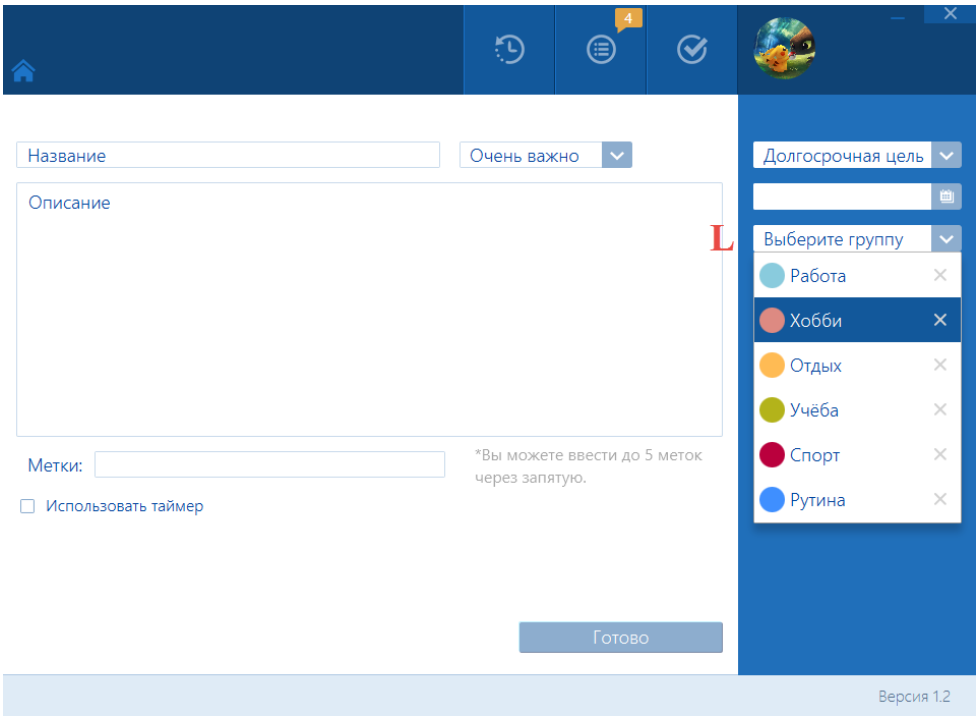

Рисунок 19. Создание новой долгосрочной цели.

Прецедент: Создание задачи.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Планирует что, во сколько и когда он должен выполнить.

Происходящее действие: Пользователь хочет добавить в список задач к выполнению новую задачу.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Создана новая задача.

Сценарий (типичное развитие событий):

1. Пользователь выбирает функцию создания новой задачи.
2. Пользователь хочет создать задачу для одного дня, он выбирает тип задачи (Рисунок 17. Пункт C), вводит название задачи (Рисунок 17. Пункт A), её описание (Рисунок 17. Пункт H), важность (Рисунок 17. Пункт B), метки (Рисунок 17. Пункт I), дату (Рисунок 18. Пункт K) и время выполнения (Рисунок 18. Пункт M).
3. Система генерирует уже сохраненные в базе данных группы. **(Операция 13.1)**
4. Пользователь выбирает группу, к которой относится задача (Рисунок 19. Пункт L).
5. Если пользователь желает, чтобы система оповестила его, когда наступит время для выполнения задачи, то он отмечает пункт «Создать Напоминание» (Рисунок 17. Пункт F).

6. Пользователь нажимает на кнопку «Готово» (Рисунок 17. Пункт J).
7. Система проводит валидацию введенных пользователем данных.
8. Валидация проходит успешно, и система сохраняет в базе данных новую задачу для одного дня. **(Операция 13.2)**

Дополнения (альтернативное развитие событий):

2а. Пользователь хочет создать ежедневную задачу, он выбирает тип задачи (Рисунок 17. Пункт C), вводит название задачи (Рисунок 17. Пункт A), её описание (Рисунок 17. Пункт H), важность (Рисунок 17. Пункт B), метки (Рисунок 17. Пункт I) и время выполнения по дням недели (Рисунок 17. Пункт D).

2б. Пользователь хочет создать долгосрочную задачу, он выбирает тип задачи (Рисунок 17. Пункт C), вводит название задачи (Рисунок 17. Пункт A), её описание (Рисунок 17. Пункт H), важность (Рисунок 17. Пункт B), метки (Рисунок 17. Пункт I) и срок выполнения (Рисунок 18. Пункт K).

4а. Пользователь выбирает функцию «Создать новую группу» (Рисунок 17. Пункт E).

1. Система генерирует поля для ввода названия группы и выбора цвета (Рисунок 17. Пункт G).
2. Пользователь вводит название группы и выбирает для неё цвет.
3. Система проверяет список уже использующихся в программе цветов. **(Операция 13.3)**
4. Система находит, что такой цвет ещё не занят ни одной группой, и он может быть использован.
    а) Система находит, что такой цвет уже занят и выводит сообщение об ошибке.
5. Система проверяет возможность использования введенного нового названия группы. **(Операция 13.4)**
6. Система находит, что такое название группы ещё не используется и оно может быть сохранено.
    а) Система находит, что такая группа уже имеется и выводит соответствующее сообщение об ошибке.

8а. Валидация проходит успешно, и система сохраняет в базе данных новую ежедневную задачу. **(Операция 13.5)**

8б. Валидация проходит успешно, и система сохраняет в базе данных новую долгосрочную задачу. **(Операция 13.6)**

8в. Валидация введенных данных не прошла успешно, и система выводит сообщение об ошибке. Информация не сохраняется в базе данных.

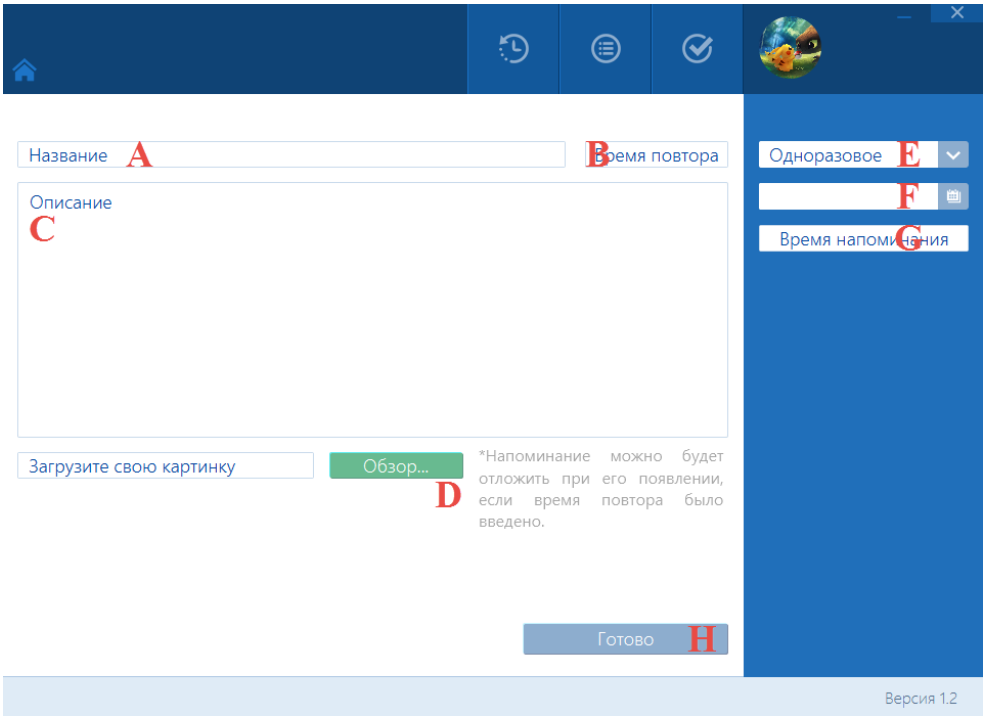

Рисунок 20. Создание нового одноразового напоминания.

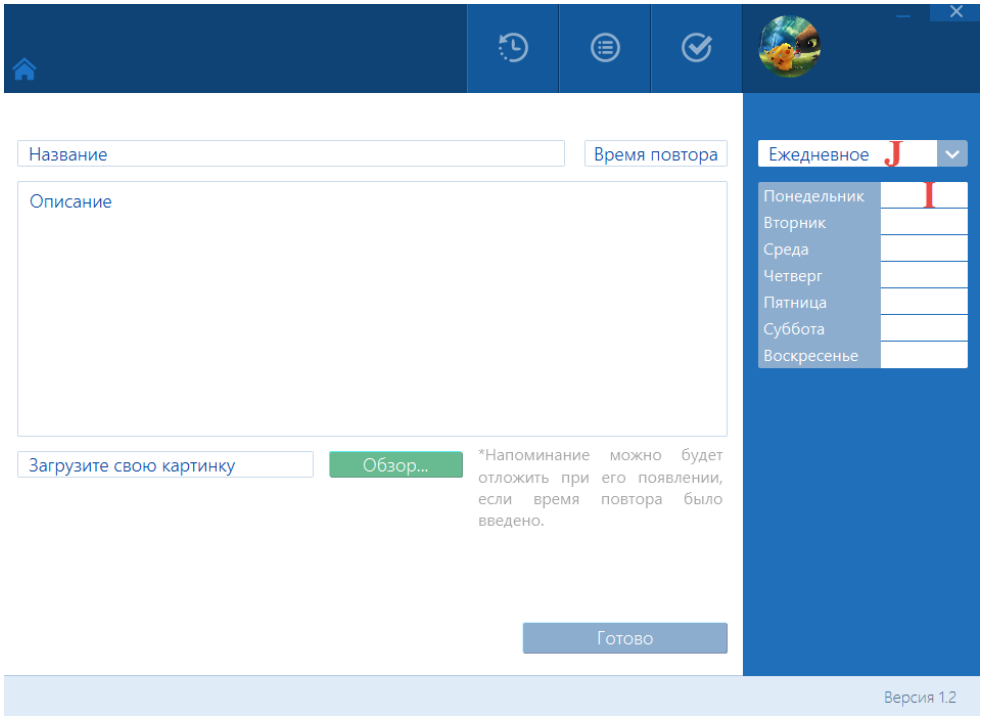

Рисунок 21. Создание нового ежедневного напоминания.

Прецедент: Создание напоминания.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Хочет не забыть о важном для него деле.

Происходящее действие: Пользователь желает создать напоминание о каком-либо событии.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Создано напоминание для события.

Сценарий (типичное развитие событий):

1. Пользователь переходит в раздел «Напоминания» (Рисунок 5. Пункт F).
2. Пользователь хочет создать одноразовое напоминание (Рисунок 20. Пункт E), он вводит название для напоминания (Рисунок 20. Пункт A), описание (Рисунок 20. Пункт C), время повторения (Рисунок 20. Пункт B), при желании, загружает картинку (Рисунок 20. Пункт D), а также день (Рисунок 20. Пункт F) и время (Рисунок 20. Пункт G), когда оно должно появиться.
3. Пользователь нажимает на кнопку «Готово» (Рисунок 20. Пункт H).
4. Система проводит валидацию введенных данных.
5. Валидация успешно пройдена. Система сохраняет в базе данных новое одноразовое напоминание. **(Операция 20.1)**

Дополнения (альтернативное развитие событий):

2а. Пользователь хочет создать ежедневное напоминание (Рисунок 21. Пункт J), он вводит название для напоминания (Рисунок 20. Пункт A), описание (Рисунок 20. Пункт C), время повторения (Рисунок 20. Пункт B), при желании, загружает картинку (Рисунок 20. Пункт D), а также время, когда оно должно появиться, расписанное по дням недели (Рисунок 21. Пункт I).

5а. Валидация успешно пройдена. Система сохраняет в базе данных новое ежедневное напоминание. **(Операция 20.2)**

5б. Валидация введенных данных не прошла успешно, и система выводит сообщение об ошибке. Информация не сохраняется в базе данных.

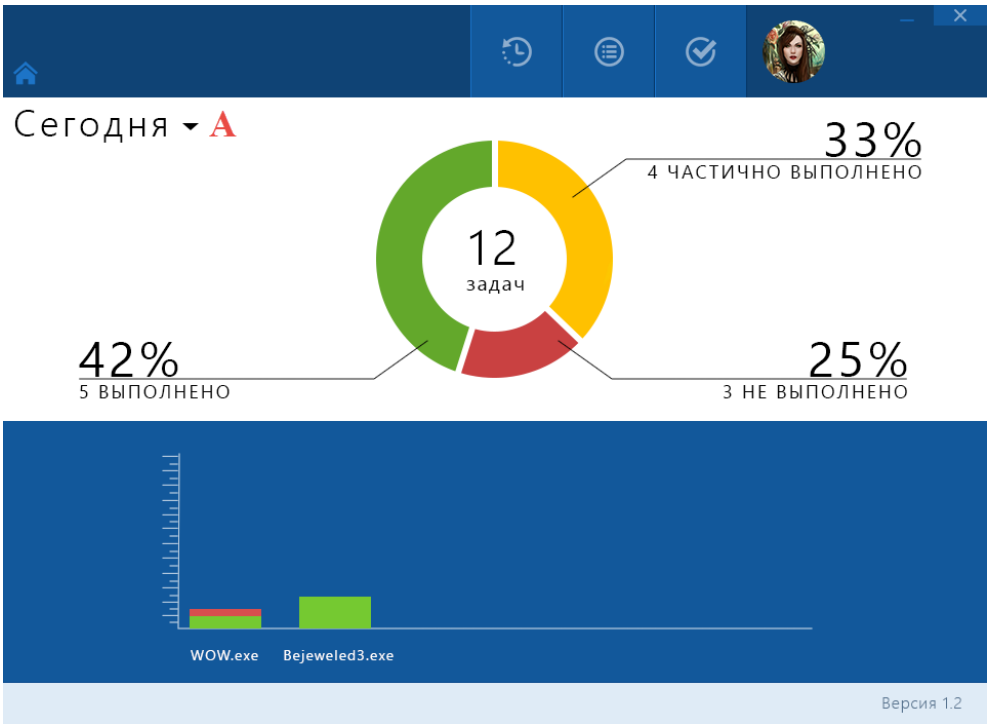

Рисунок 22. Просмотр статистики.

Прецедент: Просмотр статистики.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Хочет получить информацию о том, насколько эффективно он работал.

Происходящее действие: Пользователь хочет получить данные статистики программы о своих задачах за последний месяц.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Статистика работы пользователя составлена.

Сценарий (типичное развитие событий):

1. Пользователь переходит в раздел «Статистика» (Рисунок 5. Пункт J).
2. Система выводит начальную страницу статистики за текущий день **(Операция 23.1)**. В верхней части показаны результаты по выполнению задач пользователя, а в нижней – время работы контролируемых программ, где зелёным отмечено время работы в пределах запланированного времени, а красным время, которое было дополнительно добавлено.
3. Пользователь меняет временной промежуток на месяц (Рисунок 22. Пункт А).

4. Система генерирует отчёт статистики содержащий диаграмму и данные о том на сколько эффективно данный пользователь выполнял поставленные задачи в течение месяца. **(Операция 23.2)**

3.2. Расширенный логический дизайн базы данных.

3.2.1. Логическая схема базы данных.

3.2.1.1. Диаграмма базы данных.

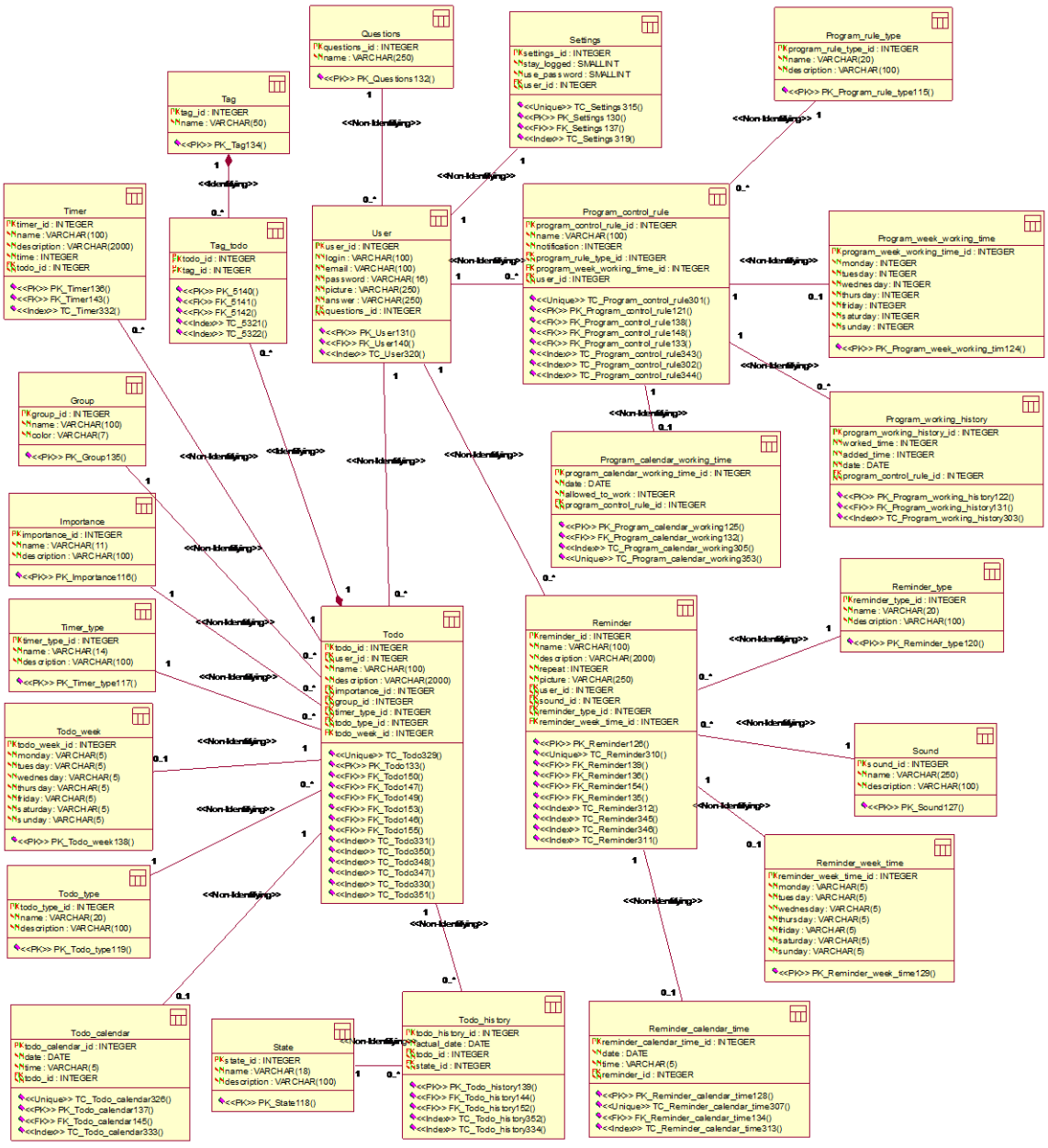

Рисунок 23. Диаграмма базы данных.

3.2.1.2. Описание таблиц.

Имя таблицы	*Отношение к регистру*	*На основе какого класса-сущности, атрибута или связи была создана?*
Group	Регистр задач	Класс-сущность: Group
Importance	Регистр классификатора	Класс-сущность: Importance
Program_calendar_working_time	Регистр контроля программ	Класс-сущность: Program_calendar_working_time
Program_rule_type	Регистр классификатора	Класс-сущность: Program_rule_type
Program_control_rule	Регистр контроля программ	Класс-сущность: Program_control_rule
Program_week_working_time	Регистр контроля программ	Класс-сущность: Program_week_working_time
Program_working_history	Регистр контроля программ	Класс-сущность: Program_working_history
Questions	Регистр пользователей	Класс-сущность: Questions
Reminder	Регистр напоминаний	Класс-сущность: Reminder
Reminder_calendar_time	Регистр напоминаний	Класс-сущность: Reminder_calendar_time
Reminder_type	Регистр классификатора	Класс-сущность: Reminder_type
Reminder_week_time	Регистр напоминаний	Класс-сущность: Reminder_week_time
Settings	Регистр пользователей	Класс-сущность: Settings
Sound	Регистр напоминаний	Класс-сущность: Sound
State	Регистр классификатора	Класс-сущность: State
Tag	Регистр задач	Класс-сущность: Tag
Tag_todo	Регистр задач	Связь: Tag – Todo
Timer	Регистр задач	Класс-сущность: Timer
Timer_type	Регистр классификатора	Класс-сущность: Timer_type
Todo	Регистр задач	Класс-сущность: Todo
Todo_calendar	Регистр задач	Класс-сущность: Todo_calendar
Todo_history	Регистр задач	Класс-сущность: Todo_history
Todo_type	Регистр классификатора	Класс-сущность: Todo_type

Имя таблицы	***Отношение к регистру***	***На основе какого класса-сущности, атрибута или связи была создана?***
Todo_week	Регистр задач	Класс-сущность: Todo_type
User	Регистр пользователей	Класс-сущность: User

Таблица 2. Описание таблиц.

3.2.1.3. Детальное описание таблиц.

См. приложение 5.5.

3.2.2. Значения классификаторов.

Название таблицы классификатора	*Свойства кодов*	*Коды*	*Соответствующие кодам названия*
Reminder_type	Монотонно возрастающие целые числа. Не генерируются автоматически системой, а вводятся разработчиком в соответствии с названием.	1 2	Одноразовое Ежедневное
Importance	Монотонно возрастающие целые числа. Не генерируются автоматически системой, а вводятся разработчиком в соответствии с названием.	1 2 3	Очень важно Важно Не важно
Program_rule_type	Монотонно возрастающие целые числа. Не генерируются автоматически системой, а вводятся разработчиком в соответствии с названием.	1 2 3	Полноценное Однодневное Родительский контроль
State	Монотонно возрастающие целые числа. Не генерируются автоматически системой, а вводятся разработчиком в соответствии с названием.	1 2 3	Не выполнено Частично выполнено Выполнено

Название таблицы классификатора	*Свойства кодов*	*Коды*	*Соответствующие кодам названия*
Timer_type	Монотонно возрастающие целые числа. Не генерируются автоматически системой, а вводятся разработчиком в соответствии с названием.	1 2 3 4	Простой таймер Сеть таймеров Pomodoro none
Todo_type	Монотонно возрастающие целые числа. Не генерируются автоматически системой, а вводятся разработчиком в соответствии с названием.	1 2 3	Цель на день Ежедневная цель Долгосрочная цель

Таблица 3. Значения классификаторов.

4. Реализация прототипа.

4.1. Реализация кода.

4.1.1. Подключение к базе данных.

Для подключения к базе данных прототипом используется библиотека Hibernate. Она упрощает маппинг получаемой информации на объекты и, кроме того, позволяет сократить количество обращений к базе данных.

Чтобы синхронизировать пользовательский интерфейс и базу данных, в классах-моделях используются *Properties*.

4.1.2. Отслеживание запущенных приложений.

Для отслеживания работы запущенных пользователем приложений был написаны классы *ProgramController* и *CheckProcess*.

Класс *ProgramController* считывает всю необходимую информацию о контролируемой программе и делает запрос классу *CheckProcess*, который сканирует системный *TaskList* на наличие контролируемой программы. Как только начало работы подконтрольного приложения будет зафиксирована, класс *ProgramController* запустит обратный отсчёт разрешенного для этой программы времени работы. Счётчик будет останавливаться и возобновляться в зависимости от выключения и включения контролируемой программы. Как только обратный отсчет достигнет нуля, *ProgramController* остановит работу контролируемого приложения.

Кроме того, класс *ProgramController* отвечает за вывод на экран предупреждения о скором отключении программы, за определенное время до окончания обратного отсчёта.

Если пользователь обладает правами для внесения изменения в правило контроля программы, то он может добавить дополнительное время, которое будет зафиксировано в истории работы программы.

4.1.3. Валидация данных.

Валидация всей вводимой пользователем информации происходит при помощи Java кода. Информация получает доступ к базе данных только в случае успешной валидации. Если же обнаруживается ошибка, то система генерирует сообщение для пользователя, в котором будет указано, где и что необходимо исправить.

4.2. Реализация пользовательского интерфейса.

4.2.1. Переключение между панелями.

Для реализации прототипа была создана 31 панель графического пользовательского интерфейса. Одна из них – основная панель *BorderPane*, куда встраиваются все остальные панели. Для эффективного переключения между ними были созданы 2 класса-контроллера: *PaneNavigator* и *MainController*. Класс *PaneNavigator* содержит в себе список наименований всех используемых программой панелей. Для смены экрана этот класс передаёт названия нужных панелей классу *MainController,* который извлекает из главной панели старые части экрана и заменяет их на новые.

4.2.2. Реакция пользовательского интерфейса на изменяющиеся данные.

Для того, чтобы пользовательский интерфейс мог сразу обновлять информацию в случае внесения изменений в базу данных, все объекты хранятся в ObservableList. А чтобы связать Java код с классами-видами, в прототипе используются бинды (bind). Они помогают связать генерируемые системой значения (например, таймер) с пользовательским интерфейсом.

4.2.3. Обновление пользовательского интерфейса.

Для отслеживания смены дней, из главного класса запускается отдельный поток (Thread), следящий за сменой суток. Как только поток фиксирует наступление нового для, он запускает обновление истории задач и контролируемых программ в базе данных, после чего обновляет пользовательский интерфейс и переменную текущего дня. После этого поток продолжает свою работу по отслеживанию смены дня.

4.3. Реализация базы данных.

4.3.1. Создание таблиц.

См. приложение 5.6.1.

4.3.2. Подготовка таблиц к работе.

См. приложение 5.6.2.

4.3.3. Удаление таблиц.

См. приложение 5.6.3.

4.4. Возможности развития.

В дальнейшем планируется создать расширение программы *Time Control* для телефонов на платформе *Android*. Это позволит добавить приложению большую мобильность. У пользователя будет возможность отслеживать свой прогресс где угодно и когда угодно. Синхронизация базы данных между компьютером и телефоном будет происходить посредством использования *Bluetooth*.

Заключение.

Главной целью дипломной работы являлась разработка прототипа программы для организации личного времени. При разработке данной программы ставились следующие цели:

- Создать функцию, которая могла бы помочь пользователю контролировать время, которое он тратит на те, или иные программы, путём установления ограничений на время использования.
- Предоставить удобный интерфейс для быстрого планирования целей пользователя.
- Дать возможность пользователю создавать напоминания о важных для него событиях.
- Вести статистику о контролируемых программах и выполненных задачах для того, чтобы дать пользователю возможность проанализировать свои действия и более эффективно распределять своё время.

Прототип был реализован на языке программирования Java, в качестве библиотеки для реализации графического интерфейса пользователя была использована JavaFX, а базой данных для хранения информации была выбрана SQLite.

Все поставленные задачи были успешно реализованы. Эта программа не имеет аналогов и может быть в будущем расширена дополнением для мобильных устройств.

Kokkuvõte.

Käesoleva diplomitöö eesmärk on isikliku aja korraldamise programmi prototüübi väljatöötamine. Käesoleva programmi läbitöötatusel olid seatud järgmised eesmärgid:

- Luua funktsiooni, mis aitaks kasutajal, kontrollida oma aega raiskamist kasutusaja limiteerimise kaudu.
- Pakkuda mugava kasutajaliidese kasutaja kiire eesmärke kavandamiseks.
- Anda võimalust kasutajale luua meeldetuletused temale tähtsatest sündmustest.
- Korraldada statistikat kontrollitud programmitest ning täidetud ülesannetest selleks, et anda kasutajale võimalust oma tegevusi analüüsida ning tõhusamalt oma aega jagada.

Prototüüp oli realiseeritud programmikeeles Java, kasutaja graafilise kasutajaliidese teostuseks oli kasutatud JavaFX, info säilitamiseks aga oli valitud andmebaasiks SQLite.

Kõik seatud eesmärgid olid edukalt teostutud. Sellel programmil ei ole analoogi ning võib laiendada tulevikus lisana mobiiltelefonidele.

Summary.

The main objective of this thesis was development of prototype for personal time management program. While developing this program goals were to:

- Create the function, which could aid user in controlling time spent on different programs usage by creating restriction rules for their time usage.
- Provide user-friendly interface for quick goals planning.
- Enable user to create reminders for important events.
- Keep statistics on controlled programs and done tasks, in order to let user analyze his actions and more efficiently distribute his time.

The prototype was implemented by using Java language, for graphical user interface library was chosen JavaFX and for local database was used SQLite.

All objectives of this thesis were successfully implemented. This program is unique and in the future can have extension for mobile devices.

Список использованной литературы.

1. Erki Eessaar. (2014). Andmebaasid II.
2. Jiawei Han, Micheline Kamber, Jian Pei. (2012). Data Mining: concepts and techniques, 3rd ed.
3. Srinivas Guruzu, Gary Mak. (2010). Hibernate Recipes: A Problem-Solution Approach.
4. С. А. Трофимов. (2002). Case-технологии, 2-ое издание.
5. Carl Dea, Mark Heckler, Gerrit Grunwald, Jose Pereda Ph.D, Sean M Phillips. (2014). JavaFX 8: Introduction by Example, 2nd ed.
6. Grant Allen, Mike Owens. (2010). The Definitive Guide to SQLite, 2nd ed.
7. Josh Juneau. (2014). Java 8 Recipes.
8. Oracle. (2012). JavaFX Scene Builder User Guide Release 2.0.
9. Kishori Sharan. (2014).Beginning Java 8 APIs, Extensions, and Libraries.
10. Определение компиляции [WWW] https://ru.wikipedia.org/wiki/Компилятор (25.05.2015)
11. Определение байта [WWW] https://ru.wikipedia.org/wiki/Байт (25.05.2015)
12. Определение JVM [WWW] https://ru.wikipedia.org/wiki/Java_(программная_платформа) (25.05.2015)
13. Определение компьютерной платформы [WWW] https://ru.wikipedia.org/wiki/Компьютерная_платформа (25.05.2015)
14. Определение лямбда выражений [WWW] https://ru.wikipedia.org/wiki/Лямбда-выражения (25.05.2015)
15. Определение Pomodoro [WWW] https://ru.wikipedia.org/wiki/Pomodoro (25.05.2015)
16. Определение учётной записи [WWW] https://ru.wikipedia.org/wiki/Учётная_запись (25.05.2015)
17. Определение чекбокса [WWW] https://ru.wikipedia.org/wiki/Флажок_(интерфейс) (25.05.2015)
18. Определение прецедента [WWW] https://ru.wikipedia.org/wiki/Прецедент_(UML) (25.05.2015)
19. Определение валидации [WWW] https://ru.wikipedia.org/wiki/Валидация (25.05.2015)
20. Определение аватара [WWW] https://ru.wikipedia.org/wiki/Аватар_(картинка) (25.05.2015)

21. Определение маппинга [WWW]
http://software-testing.ru/forum/index.php?/topic/13199-mapping/ (25.05.2015)
22. Определение Android [WWW]
https://ru.wikipedia.org/wiki/Android (25.05.2015)
23. Определение синхронизации [WWW]
https://ru.wikipedia.org/wiki/Синхронизация (25.05.2015)
24. Определение Bluetooth [WWW]
https://ru.wikipedia.org/wiki/Bluetooth (25.05.2015)

Приложения.

5.1. Детальный анализ прецедентов.

Красным отмечены операции, вносящие изменения в базу данных (создание, обновление, удаление).

Синим отмечены операции чтения информации из базы данных.

Прецедент: Регистрация пользователя.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Желает создать свою личную учётную запись для того, чтобы иметь доступ ко всем функциям системы.

Происходящее действие: Пользователь хочет зарегистрироваться в системе.

Предварительные требования: Программа должна быть запущена.

Результат: Данные пользователя сохраняются в базе данных и для него создаётся соответствующая учётная запись. Теперь ему доступны все функции программы.

Сценарий (типичное развитие событий):

1. Пользователь желает зарегистрироваться. На главной странице он нажимает кнопку «Регистрация».
2. Пользователь вводит свой логин и электронную почту. Он оставляет не отмеченным пункт о необходимости использования пароля.
3. Система проводит валидацию почты и проверяет возможность использования введенного логина.
4. Система разрешает использовать данный логин.
5. Пользователь выбирает контрольный вопрос и вводит на него ответ.
6. Пользователь выбирает изображение для своего аватара.
7. Система сохраняет данные пользователя. **(Операция 1.1)**

Дополнения (альтернативное развитие событий):

2a. Пользователь вводит свой логин и электронную почту. Он оставляет не отмеченным пункт о необходимости использования пароля.

4а. Система обнаруживает, что данный логин уже используется другим пользователем. На экран выводится сообщение об ошибке с просьбой изменить логин.

Прецедент: Идентификация пользователя.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Желает войти в систему, чтобы иметь доступ ко всем её функциям.

Происходящее действие: Пользователь хочет войти в систему.

Предварительные требования: Программа должна быть запущена, и пользователь должен быть зарегистрирован в системе.

Результат: Пользователь идентифицирован и ему предоставлен доступ ко всем возможностям системы.

Сценарий (типичное развитие событий):

1. Пользователь выбирает из предложенного ему списка свой логин. **(Операция 2.1)**
2. Система находит настройки для выбранного логина.
3. Если для данного пользователя не установлена необходимость ввода пароля для входа в систему, то программа идентифицирует пользователя и предоставляет ему право пользоваться всеми функциями программы.

Дополнения (альтернативное развитие событий):

3а. Если для данного пользователя установлена необходимость ввода пароля для входа в систему, то программа выводит поле для ввода пароля.

1. Пользователь вводит пароль.
2. Система проверяет правильность введенного пароля.
3. Если всё верно, то программа идентифицирует пользователя и предоставляет ему право пользоваться всеми функциями программы.
    а) Если введенный пароль не совпадает с паролем, сохраненным в системе, то система выводит сообщение об ошибке и отказывается идентифицировать данного пользователя.

Прецедент: Изменение личных данных пользователя.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Поменял свою почту и теперь хочет изменить свои данные.

Происходящее действие: Пользователь хочет изменить сохранённую в системе почту.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Электронная почта изменена.

Сценарий (типичное развитие событий):

1. Пользователь переходит раздел «Настройки».
2. Пользователь переходит в раздел «Личные данные пользователя».
3. Пользователь изменяет название своей почты и нажимает «Сохранить»
4. Система проводит валидацию новых данных.
5. Валидация прошла успешно, и система сохраняет новую информацию в базе данных. **(Операция 3.1)**

Дополнения (альтернативное развитие событий):

5а. Новые данные не прошли валидацию, система выводит сообщение об ошибке. Изменения в информацию пользователя не вносятся.

Прецедент: Восстановление пароля.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Хочет восстановить свой пароль от аккаунта, который он забыл.

Происходящее действие: Пользователь желает восстановить пароль.

Предварительные требования: Пользователь должен быть зарегистрирован в системе.

Результат: Пароль пользователя восстановлен.

Сценарий (типичное развитие событий):

1. Пользователь выбирает пункт «Забыли пароль?».
2. Пользователь хочет восстановить свой пароль ответив на контрольный вопрос.
3. Пользователь отвечает на контрольный вопрос.
4. Система сверяет ответ пользователя с ответом ранее сохраненном в базе данных. **(Операция 4.1)**
5. Ответ пользователя правильный, система выводит на экран пароль пользователя.

Дополнения (альтернативное развитие событий):

2а. Пользователь хочет, чтобы его пароль был прислан ему на почту.

   1. Система отправляет письмо с паролем пользователя на зарегистрированную в системе электронную почту.

5а. Ответ пользователя не правильный, система выводит на экран сообщение об ошибке и отказывает в восстановлении пароля

Прецедент: Изменение настроек пользователя.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Разрешил другому человеку пользоваться его компьютером и теперь он хочет, чтобы кроме него никто не мог получить доступ к его данным.

Происходящее действие: Пользователь хочет начать использовать пароль для входя в систему и автоматически выходить из системы при её отключении.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Состояние задачи изменено.

Сценарий (типичное развитие событий):

1. Пользователь переходит раздел «Настройки».
2. Пользователь убирает выделение с пункта «Оставаться в системе» и отмечает пункт «Использовать пароль». **(Операция 5.1)**
3. Система проверяет, есть ли у пользователя пароль.
4. У пользователя есть пароль и система сохраняет новые настройки.

Дополнения (альтернативное развитие событий):

4а. У пользователя нет пароля, система выводит на экран поле для создания пароля.

1. Пользователь вводит пароль и нажимает на кнопку «Готово».
2. Система сохраняет пароль для пользователя и его новые настройки. **(Операция 5.2)**

Прецедент: Удаление пользователя.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Больше не собирается пользоваться этим компьютером и хочет удалить свой аккаунт.

Происходящее действие: Пользователь собирается удалить свой аккаунт.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Пользователь удалён.

Сценарий (типичное развитие событий):

1. Пользователь переходит в свои настройки.
2. В меню пользователь выбирает пункт «Удаление пользователя».
3. Пользователь нажимает на кнопку «Удалить пользователя».
4. Система предупреждает о необратимости действия и просит подтверждения.
5. Пользователь подтверждает своё действие.
6. Система удаляет данного пользователя и все связанные с ним объекты из базы данных. **(Операция 6.1)**

Прецедент: Создание правила контроля.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Желает контролировать время работы своей программы.

Происходящее действие: Пользователь хочет создать новое правило контроля.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Создано новое правило контроля времени работы программы.

Сценарий (типичное развитие событий):

1. Пользователь выбирает функцию создания нового правила контроля программы.
2. Пользователь вводит расположение программы, сколько и по каким дням она должна работать, за сколько времени до конца работы приложения необходимо выводить сообщение о скором окончании работы.
3. Система проверяет возможно ли создать данное правило контроля для этой программы. **(Операция 7.1)**
4. Программа сохраняет новое правило контроля. **(Операция 7.2)**

Дополнения (альтернативное развитие событий):

4а. Для данной программы уже создано правило с высшим приоритетом. Система выводит сообщение об ошибке и не позволяет сохранить новое правило контроля.

Прецедент: Просмотр списка контролируемых программ.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Желает легко и быстро получить информацию о том какие программы, на которых распространяется контроль, сейчас запущены, сколько они уже работали и было ли использовано дополнительное время.

Происходящее действие: Пользователь хочет получить список всех контролируемых программ.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Показан список всех программ, на которых распространяются правила контроля.

Сценарий (типичное развитие событий):

1. Пользователь открывает раздел, отвечающий за созданные правила контроля.
2. Система генерирует списки всех программ, на которых распространяются правила контроля. **(Операция 8.1)**
3. Если пользователь хочет увидеть лишь те программы, правила которых он создал сам, то он выбирает пункт «Только мои».
4. Система генерирует список контролируемых программ, относящихся к данному пользователю. **(Операция 8.2)**

Прецедент: Продление времени работы программы.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Не успел сделать всё что запланировал в контролируемой программе за отведённое ей время и хочет продолжить работать в ней.

Происходящее действие: Пользователь хочет увеличить разрешённое время работы программы на текущие сутки.

Предварительные требования: Пользователь должен быть авторизован в системе и у него должны быть права на внесение изменений в правило контроля программы. Для приложения должно быть создано правило контроля.

Результат: Время, которое программе разрешено работать в данные сутки, увеличено.

Сценарий (типичное развитие событий):

1. Пользователь выбирает из списка контролируемых приложений программу, время работы которой хочет увеличить.
2. Пользователь задаёт дополнительное время работы программы.
3. Пользователь нажимает на кнопку «Сохранить».

4. Система сохраняет новое время работы программы и отображает время, которое было добавлено. **(Операция 9.1)** **(Операция 8.1)**

Прецедент: Изменение времени показа оповещения.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Чувствует, что время для появления оповещения не совсем для него удобно и хочет поменять его на более подходящее.

Происходящее действие: Пользователь хочет изменить время появления оповещения.

Предварительные требования: Пользователь должен быть авторизован в системе и у него должны быть права на внесение изменений в правило контроля программы. Для приложения должно быть создано правило контроля.

Результат: Время появления оповещения о скором прекращении работы программы изменено.

Сценарий (типичное развитие событий):

1. Пользователь выбирает из списка контролируемых приложений программу, время появления оповещения которой хочет изменить.
2. Пользователь изменяет время появления оповещения.
3. Пользователь нажимает на кнопку «Сохранить».
4. Система сохраняет новое время появления оповещения. **(Операция 10.1)**

Прецедент: Изменение расписания работы программы.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Поменял свой образ жизни и теперь желает по-другому использовать программу.

Происходящее действие: Пользователь хочет изменить дни, по которым программа будет контролироваться, а также сколько ей будет разрешено работать.

Предварительные требования: Пользователь должен быть авторизован в системе и у него должны быть права на внесение изменений в правило контроля программы. Для приложения должно быть создано правило контроля.

Результат: Изменено время, отведённое на работу контролируемой программы.

Сценарий (типичное развитие событий):

1. Пользователь выбирает из списка контролируемых приложений программу, время работы которой хочет изменить.
2. Пользователь изменяет время работы программы по дням недели.
3. Пользователь нажимает на кнопку «Сохранить».
4. Система сохраняет новое расписание разрешённого времени работы программы. **(Операция 11.1)**

Прецедент: Удаление правила контроля программы.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Удалил контролируемую программу и ему больше не нужно для неё правило контроля.

Происходящее действие: Пользователь желает удалить правило контроля своей программы.

Предварительные требования: Пользователь должен быть авторизован в системе и у него должны быть права на внесение изменений в правило контроля программы. Для приложения должно быть создано правило контроля.

Результат: Правило контроля программы успешно удалено.

Сценарий (типичное развитие событий):

1. Пользователь выбирает из списка контролируемых приложений программу, правило контроля которой желает удалить.
2. Пользователь нажимает на кнопку «Удалить».
3. Система удаляет правила контроля для соответствующей программы. **(Операция 12.1)**

Прецедент: Создание задачи.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Планирует что, во сколько и когда он должен выполнить.

Происходящее действие: Пользователь хочет добавить в список задач к выполнению новую задачу.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Создана новая задача.

Сценарий (типичное развитие событий):

1. Пользователь выбирает функцию создания новой задачи.
2. Пользователь хочет создать задачу для одного дня, он вводит название задачи, её описание, метки, дату и время выполнения.
3. Система генерирует уже сохраненные в базе данных группы. **(Операция 13.1)**
4. Пользователь выбирает группу, к которой относится задача.
5. Пользователь нажимает на кнопку «Готово».
6. Система проводит валидацию введенных пользователем данных.
7. Валидация проходит успешно, и система сохраняет в базе данных новую задачу для одного дня. **(Операция 13.2)**

Дополнения (альтернативное развитие событий):

2а. Пользователь хочет создать ежедневную задачу, он вводит название задачи, её описание, метки и время выполнения по дням недели.

2б. Пользователь хочет создать долгосрочную задачу, он вводит название задачи, её описание, метки и срок выполнения.

4а. Пользователь выбирает функцию «Создать новую группу».

1. Система генерирует поля для ввода названия группы и выбора цвета.
2. Пользователь вводит название группы и выбирает для неё цвет.
3. Система проверяет список уже использующихся в программе цветов. **(Операция 13.3)**
4. Система находит, что такой цвет ещё не занят ни одной группой, и он может быть использован.
    a) Система находит, что такой цвет уже занят и выводит сообщение об ошибке.
5. Система проверяет возможность использования введенного нового названия группы. **(Операция 13.4)**
6. Система находит, что такое название группы ещё не используется и оно может быть сохранено.
    a) Система находит, что такая группа уже имеется и выводит соответствующее сообщение об ошибке.

7а. Валидация проходит успешно, и система сохраняет в базе данных новую ежедневную задачу. **(Операция 13.5)**

7б. Валидация проходит успешно, и система сохраняет в базе данных новую долгосрочную задачу. **(Операция 13.6)**

Прецедент: Создание таймера.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Хочет максимально эффективно использовать своё время, отведённое на выполнение поставленной задачи.

Происходящее действие: Пользователь хочет создать новый таймер для задачи.

Предварительные требования: Пользователь должен быть авторизован в системе и находится в процессе создания новой задачи.

Результат: Для задачи создан и сохранен новый таймер.

Сценарий (типичное развитие событий):

1. Пользователь выбирает пункт «Создать новый таймер».
2. Система генерирует список возможных таймеров: простой таймер, сеть таймеров, таймер Pomodoro.
3. Пользователь выбирает один из таймеров:
   - 3.1. Простой таймер:
     - 3.1.1. Пользователь вводит название таймера и необходимое время работы.
     - 3.1.2. Система сохраняет таймер для данной задачи. **(Операция 14.1)**
   - 3.2. Сеть таймеров:
     - 3.2.1. Пользователь создаёт цепочку из необходимого количества таймеров, для каждого вводит название и время работы.
     - 3.2.2. Система сохраняет сеть таймеров для данной задачи. **(Операция 14.1)**
   - 3.3. Таймер Pomodoro:
     - 3.3.1. Пользователь выбирает необходимое количество «помидоров» (таймеры) для своей работы.
     - 3.3.2. Пользователь выбирает нужную величину «помидоров» (15, 20, 25, 30, 35, 40, 45, 50, 55 или 60 минут).
     - 3.3.3. Пользователь выбирает величину перерывов между «помидорами» (5, 10, 15, 20 или 25 минут).
     - 3.3.4. Пользователь вводит название и, при желании, описание для каждого «помидора».
     - 3.3.5. Система сохраняет «Pomodoro» таймер для данной задачи. **(Операция 14.1)**

Прецедент: Просмотр списка задач.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Желает посмотреть какие задачи у него запланированы.

Происходящее действие: Пользователь хочет получить список всех его задач.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Составлен список всех запланированных задач пользователя.

Сценарий (типичное развитие событий):

1. Пользователь переходит в свой «Ежедневник».
2. Система генерирует список всех созданных разовых задач на текущий день, относящихся к данному пользователю. **(Операция 15.1)**
3. Если пользователь хочет получить данные о его рутинных задачах на сегодня, то он выбирает раздел «Ежедневные задачи».
4. Система генерирует список всех созданных постоянных задач для текущего дня, относящихся к данному пользователю. **(Операция 15.2)**
5. Если пользователь хочет получить данные о его долгосрочных запланированных задачах, то он выбирает раздел «Долгосрочные цели».
6. Система генерирует список всех долгосрочных целей, относящихся к данному пользователю. **(Операция 15.3)**

Прецедент: Запуск таймера.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Готов приступить к выполнению поставленной задачи и хочет видеть насколько эффективно он работает.

Происходящее действие: Пользователь хочет запустить таймер для его задания.

Предварительные требования: Пользователь должен быть авторизован в системе. Задача должна быть создана.

Результат: Таймер запущен.

Сценарий (типичное развитие событий):

1. Пользователь переходит в свой «Ежедневник».
2. Система генерирует список созданных задач, относящихся к данному пользователю. **(Операция 15.1)**

3. Пользователь находит в списке задачу, которой он собирается заниматься.
4. Пользователь нажимает на иконку таймера.
5. Система выводит на экран страницу таймера, относящегося к данной задаче.
6. Пользователь нажимает на кнопку «Запустить таймер».
7. Система запускает текущий таймер.

Прецедент: Изменение состояния задач.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Хочет быстро получать информацию о том в каком состоянии (сделано, частично сделано, не сделано) сейчас находятся его задачи.

Происходящее действие: Пользователь хочет изменить состояние его задачи.

Предварительные требования: Пользователь должен быть авторизован в системе. Задача должна быть создана.

Результат: Состояние задачи изменено.

Сценарий (типичное развитие событий):

1. Пользователь переходит в свой «Ежедневник» через иконку в верхнем меню экрана.
2. Система генерирует список созданных задач, относящихся к данному пользователю. **(Операция 15.1)**
3. Пользователь находит задачу, состояние которой хочет изменить.
4. Пользователь нажимает на иконку настроения, символизирующую состояние задания, чтобы изменить состояние задачи на нужное (смена: не выполнено -> частично выполнено -> выполнено).
5. Система обновляет состояние задачи в базе данных. **(Операция 17.1)**

Дополнения (альтернативное развитие событий):

4а. Пользователь выбирает задачу.

1. Система генерирует полную информацию о задании.
2. Пользователь нажимает на кнопку «Изменить».
3. Пользователь меняет состояние задачи.
4. Пользователь нажимает на кнопку «Сохранить».

Прецедент: Перенесение сроков выполнения задачи.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Не успевает выполнить поставленную себе задачу и решает перенести её на другой день.

Происходящее действие: Пользователь хочет перенести задачу на более удобный для него день.

Предварительные требования: Пользователь должен быть авторизован в системе. Задача должна быть создана.

Результат: Задача перенесена на другой день.

Сценарий (типичное развитие событий):

1. Пользователь переходит в свой «Ежедневник».
2. Система генерирует список всех созданных задач, относящихся к данному пользователю. **(Операция 15.1)**
3. Пользователь выбирает задачу, которую хочет перенести.
4. Пользователь нажимает на кнопку «Изменить».
5. Пользователь выбирает дату, на которую хочет перенести задачу.
6. Пользователь нажимает на кнопку «Сохранить».
7. Система сохраняет новую дату для данной задачи. **(Операция 18.1)**

Дополнения (альтернативное развитие событий):

7а. Если новое выбранное число раньше, чем текущий день, то изменения не сохраняются, и система выводит сообщение об ошибке.

Прецедент: Удаление задачи.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Решает изменить список запланированных дел.

Происходящее действие: Пользователь желает удалить задачи, которые он более не намерен выполнять.

Предварительные требования: Пользователь должен быть авторизован в системе. Задача должна быть создана.

Результат: Задача удалена.

Сценарий (типичное развитие событий):

1. Пользователь открывает раздел, отвечающий за его задачи.

2. Система генерирует список задач на сегодняшний день.
3. Пользователь выбирает задачу, которую хочет удалить.
4. Пользователь нажимает на кнопку «Удалить».
5. Система удаляет задачу. **(Операция 19.1)**

Прецедент: Создание напоминания.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Хочет не забыть о важном для него деле.

Происходящее действие: Пользователь желает создать напоминание о каком-либо событии.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Создано напоминание для события.

Сценарий (типичное развитие событий):

1. Пользователь переходит в раздел «Напоминания».
2. Пользователь хочет создать одноразовое напоминание, он вводит название для напоминания, описание, время повторения, при желании, загружает картинку, а также день и время, когда оно должно появиться.
3. Пользователь нажимает на кнопку «Сохранить».
4. Система проводит валидацию введенных данных.
5. Валидация успешно пройдена. Система сохраняет в базе данных новое одноразовое напоминание. **(Операция 20.1)**

Дополнения (альтернативное развитие событий):

2а. Пользователь хочет создать ежедневное напоминание, он вводит название для напоминания, описание, время повторения, при желании, загружает картинку, а также время, когда оно должно появиться, расписанное по дням недели.

5а. Валидация успешно пройдена. Система сохраняет в базе данных новое ежедневное напоминание. **(Операция 20.2)**

5б. Валидация введенных данных не прошла успешно, и система выводит сообщение об ошибке. Информация не сохраняется в базе данных.

Прецедент: Просмотр списка напоминаний.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Хочет получить информацию о ближайших событиях, про которые он не должен забывать.

Происходящее действие: Пользователь хочет получить список своих напоминаний.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Составлен список всех запланированных задач пользователя.

Сценарий (типичное развитие событий):

1. Пользователь переходит в раздел отвечающий за созданные напоминания.
2. Система генерирует список всех созданных напоминаний, относящихся к данному пользователю, на текущий день. **(Операция 21.1)**
3. Если пользователь хочет получить данные о всех его напоминаниях, то он выбирает раздел «Все».
4. Система генерирует список всех созданных данным пользователем напоминаний. **(Операция 21.2)**
5. Если пользователь хочет получить данные о напоминаниях на какой-то определённый день, то он выбирает раздел «Определённый день».
6. Система выводит поле для выбора числа.
7. Пользователь выбирает число.
8. Система генерирует список напоминаний, созданных пользователем, для выбранного числа. **(Операция 21.3)**

Прецедент: Удаление напоминания.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Больше не нуждается в напоминании о данном событии.

Происходящее действие: Пользователь хочет удалить напоминание о данном событии.

Предварительные требования: Пользователь должен быть авторизован в системе. Напоминание о событии должно быть создано.

Результат: Напоминание успешно удалено.

Сценарий (типичное развитие событий):

1. Пользователь переходит в раздел отвечающий за созданные напоминания.
2. Система генерирует список всех созданных данным пользователем напоминаний. **(Операция 21.2)**

3. Пользователь выбирает из списка напоминание, которое желает удалить.
4. Пользователь нажимает на кнопку «Удалить».
5. Система удаляет данное напоминание. **(Операция 22.1)**

Прецедент: Просмотр статистики.

Главное действующее лицо: Пользователь.

Стороны и их интересы:

- Пользователь: Хочет получить информацию о том, насколько эффективно он работал.

Происходящее действие: Пользователь хочет получить данные статистики программы о своих задачах за последний месяц.

Предварительные требования: Пользователь должен быть авторизован в системе.

Результат: Статистика работы пользователя составлена.

Сценарий (типичное развитие событий):

1. Пользователь переходит в раздел «Статистика».
2. Система выводит начальную страницу статистики за текущий день. **(Операция 23.1)**
3. Пользователь меняет временной промежуток на месяц.
4. Система генерирует отчёт статистики содержащий диаграмму и данные о том на сколько эффективно данный пользователь выполнял поставленные задачи в течение месяца. **(Операция 23.2)**

5.2. Операции с базами данных.

Операция 1.1 Создание нового пользователя (логин, почта, пароль, название аватара, ответ, questions_id)

Условия:

- Список вопросов внесён в базу данных.

Результат:

- Создан новый пользователь (User).
- User.login=логин.
- User.email=почта.
- User.password=пароль (если пароль не был введен, то присваивается значение null).
- User.picture= название аватара.
- User.answer=ответ.

- User.questions_id=questions_id (id вопроса).
- Создаются настройки (Settings) для пользователя.
- Settings.stay_logged=1 (по умолчанию).
- Settings.use_password=1 (если был введён пароль) или 0 (если пароль не был введён).
- Settings.user_id=user_id (id созданного пользователя).
- Пользователь (User) связан с Вопросами (Questions), которые используются для восстановления пароля.

Использующие прецеденты: Регистрация пользователя.

Операция 3.1 Изменение электронной почты пользователя (user_id, почта)

Условия:

- Пользователь зарегистрирован и авторизован в системе.

Результат:

- User.email=почта.

Использующие прецеденты: Изменение личных данных пользователя.

Операция 5.1 Изменение настроек пользователя (user_id, оставаться в системе, использовать пароль)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.

Последующие условия:

- Settings.stay_logged=оставаться в системе (возможные значения: 1 или 0)
- Settings.use_password= использовать пароль (возможные значения: 1 или 0)
- Настройки (Settings) связаны с Пользователем (User), для которого сохраняются настройки.

Использующие прецеденты: Изменение настроек пользователя.

Операция 5.2 Создание пароля для входа в систему (user_id, пароль, оставаться в системе, использовать пароль)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.

Последующие условия:

- User.password=пароль.
- Settings.stay_logged=оставаться в системе (возможные значения: 1 или 0)
- Settings.use_password= использовать пароль (значение = 1)

- Настройки (Settings) связаны с Пользователем (User), для которого сохраняются настройки.

Использующие прецеденты: Изменение настроек пользователя.

Операция 6.1 Удаление пользователя (user_id)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.

Последующие условия:

- Пользователь удалён.
- Удалены Настройки (Settings), связанные с Пользователем (User).
- Удалены все, связанные с Пользователем (User), Правила контроля (Program_control_rule) и относящиеся к ним записи из Недельных часов работы (Program_week_working_time), Календаря работы (Program_calendar_working_time) и Истории контроля (Program_working_history).
- Удалены все, связанные с Пользователем (User), Напоминания (Reminder) и относящиеся к ним записи из Недельных часов напоминаний (Reminder_week_time) и Календаря напоминаний (Reminder_calendar_time).
- Удалены все, связанные с Пользователем (User), Задачи (Todo) и относящиеся к ним записи из Таймеров (Timer), Недельных задач (Todo_week), Календарных задач (Todo_calendar) и Истории задач (Todo_history).

Использующие прецеденты: Удаление пользователя.

Операция 7.2 Создание нового правила контроля (название программы, время работы программы для понедельника, время работы программы для вторника, время работы программы для среды, время работы программы для четверга, время работы программы для пятницы, время работы программы для субботы, время работы программы для воскресенья, время появления оповещения, user_id, program_rule_type_id)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.

Последующие условия:

- Создано Недельное расписание работы программы (Program_week_working_time)
- Program_week_working_time.monday=время работы программы для понедельника.
- Program_week_working_time.tuesday=время работы программы для вторника.
- Program_week_working_time.wednesday=время работы программы для среды.

- Program_week_working_time.thursday=время работы программы для четверга.
- Program_week_working_time.friday=время работы программы для пятницы.
- Program_week_working_time.saturday=время работы программы для субботы.
- Program_week_working_time.sunday=время работы программы для воскресенья.
- Создано новое правило контроля программы(Program_control_rule).
- Program_control_rule.name=название программы.
- Program_control_rule.notification=время появления оповещения.
- Program_control_rule.program_rule_type_id=program_rule_type_id (id типа контроля программы).
- Program_control_rule.program_week_working_time_id=program_week_working_time _id (id Недельного расписания работы программы).
- Program_control_rule.user_id=user_id (id пользователя).
- Правило контроля программы (Program_control_rule) связано с Типом контроля программы (Program_rule_type), который определяет к какому уровню приоритета относится созданное правило.
- Правило контроля программы (Program_control_rule) связано с Пользователем (User), которому принадлежит создаваемое правило контроля.
- Правило контроля программы (Program_control_rule) связано с Недельным расписанием работы программы (Program_week_working_time), где содержится разрешённое время работы программы, распределённое по дням недели.

Использующие прецеденты: Создание правила контроля.

<u>Операция 9.1</u> Создание дополнительного времени работы программы (program_control_rule_id, дополнительное время)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.
- Для программы создано правило контроля.
- В данный день программа контролируется.

Последующие условия:

- Program_working_history.added_time=дополнительное время.
- История работы программы (Program_working_history) связана с Правилом контроля программы (Program_control_rule), для которого она создаётся.

Использующие прецеденты: Продление времени работы программы.

<u>Операция 10.1</u> Изменение времени появления оповещения (program_control_rule_id, время для оповещения)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.
- Для программы создано правило контроля.

Последующие условия:

- Program_control_rule.notofication=время для оповещения

Использующие прецеденты: Изменение времени показа оповещения

<u>Операция 11.1</u> Изменение расписания контроля программы (program_control_rule_id, время работы программы для понедельника, время работы программы для вторника, время работы программы для среды, время работы программы для четверга, время работы программы для пятницы, время работы программы для субботы, время работы программы для воскресенья)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.
- Для программы создано правило контроля.

Последующие условия:

- Program_week_working_time.monday=время работы программы для понедельника.
- Program_week_working_time.tuesday=время работы программы для вторника.
- Program_week_working_time.wednesday=время работы программы для среды.
- Program_week_working_time.thursday=время работы программы для четверга.
- Program_week_working_time.friday=время работы программы для пятницы.
- Program_week_working_time.saturday=время работы программы для субботы.
- Program_week_working_time.sunday=время работы программы для воскресенья.
- Недельное расписание работы программы (Program_week_working_time) связано с Правилом контроля программы (Program_control_rule), для которого создаётся расписание работы.

Использующие прецеденты: Изменение расписания работы программы.

<u>Операция 12.1</u> Удаление правила контроля программы (program_control_rule_id)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.
- Для программы создано правило контроля.

Последующие условия:

- Удалены соответствующие записи из таблицы Истории работы программы (Program_working_history).
- Удалены соответствующие записи из таблицы Недельного расписания работы программы (Program_week_working_time).
- Правило контроля удалено из таблицы Правило контроля программы (Program_control_rule).
- Правило контроля программы (Program_control_rule) связано с Историей работы программы (Program_working_history), куда заносится информация о том, как проходит контроль над программой.
- Правило контроля программы (Program_control_rule) связано с Недельным расписанием работы программы (Program_week_working_time), где содержится разрешённое время работы программы, распределённое по дням недели.

Использующие прецеденты: Удаление правила контроля программы.

Операция 13.2 Создание новой задачи на день (user_id, название задачи, описание задачи, время выполнения задачи, дата выполнения задачи, importance_id, group_id, timer_type_id, todo_type_id)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.

Последующие условия:

- Создана новая Задача (Todo).
- Todo.user_id=user_id (id пользователя, которому принадлежит задача).
- Todo.name=название задачи.
- Todo.description=описание задачи.
- Todo.importance_id=importance_id (id уровня важности).
- Todo.group_id=group_id (id группы, к которой принадлежит задача).
- Todo.timer_type_id=timer_type_id (id вида используемого таймера).
- Todo.todo_type_id=todo_type_id (id вида задачи).
- Создан Календарь задачи (Todo_calendar).
- Todo_calendar.date=дата выполнения задачи.
- Todo_calendar.time=время выполнения задачи.
- Todo_calendar.todo_id=todo_id (id созданной задачи).
- Задачи (Todo) связаны с Уровнем важности задачи (Importance), который показывает на сколько пользователю необходимо выполнить поставленную задачу.

- Задачи (Todo) связаны с Группами (Group), которые используются пользователем для распределения своих задач.
- Задачи (Todo) связаны с Видом таймеров (Timer_type), который определяет какой таймер пользователь хочет использовать для данной работы.
- Задачи (Todo) связаны с Видом задачи (Todo_type), определяющим каким образом пользователь желает выполнить поставленную им задачу.
- Календарь задачи (Todo_calendar) связан с Задачей (Todo), для которой он был создан.

Использующие прецеденты: Создание задачи.

<u>Операция 13.5</u> Создание новой ежедневной задачи (user_id, название задачи, описание задачи, время выполнения задачи для понедельника, время выполнения задачи для вторника, время выполнения задачи для среды, время выполнения задачи для четверга, время выполнения задачи для пятницы, время выполнения задачи для субботы, время выполнения задачи для воскресенья, importance_id, group_id, timer_type_id, todo_type_id)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.

Последующие условия:

- Создано Недельное расписание задачи (Todo_week).
- Todo_week.monday=время выполнения задачи для понедельника**.**
- Todo_week.tuesday=время выполнения задачи для вторника**.**
- Todo_week.wednesday=время выполнения задачи для среды**.**
- Todo_week.thursday=время выполнения задачи для четверга
- Todo_week.friday=время выполнения задачи для пятницы**.**
- Todo_week.saturday=время выполнения задачи для субботы**.**
- Todo_week.sunday=время выполнения задачи для воскресенья**.**
- Создана новая Задача (Todo).
- Todo.user_id=user_id (id пользователя, которому принадлежит задача).
- Todo.name=название задачи.
- Todo.description=описание задачи.
- Todo.importance_id=importance_id (id уровня важности).
- Todo.group_id=group_id (id группы, к которой принадлежит задача).
- Todo.timer_type_id=timer_type_id (id вида используемого таймера).

- Todo.todo_type_id=todo_type_id (id вида задачи).
- Todo.todo_week_id=todo_week_id (id созданного недельного расписания задачи).
- Задачи (Todo) связаны с Уровнем важности задачи (Importance), который показывает на сколько пользователю необходимо выполнить поставленную задачу.
- Задачи (Todo) связаны с Группами (Group), которые используются пользователем для распределения своих задач.
- Задачи (Todo) связаны с Видом таймеров (Timer_type), который определяет какой таймер пользователь хочет использовать для данной работы.
- Задачи (Todo) связаны с Видом задачи (Todo_type), определяющим каким образом пользователь желает выполнить поставленную им задачу.
- Задачи (Todo) связаны с Недельным расписанием задачи (Todo_week), в котором хранится расписание выполнения задачи в течении недели.

Использующие прецеденты: Создание задачи.

<u>Операция 13.6</u> Создание новой долгосрочной задачи (user_id, название задачи, описание задачи, дата выполнения задачи, importance_id, group_id, timer_type_id, todo_type_id)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.

Последующие условия:

- Создана новая Задача (Todo).
- Todo.user_id=user_id (id пользователя, которому принадлежит задача).
- Todo.name=название задачи.
- Todo.description=описание задачи.
- Todo.importance_id=importance_id (id уровня важности).
- Todo.group_id=group_id (id группы, к которой принадлежит задача).
- Todo.timer_type_id=timer_type_id (id вида используемого таймера).
- Todo.todo_type_id=todo_type_id (id вида задачи).
- Создан Календарь задачи (Todo_calendar).
- Todo_calendar.date=дата выполнения задачи.
- Todo_calendar.todo_id=todo_id (id созданной задачи).
- Задачи (Todo) связаны с Уровнем важности задачи (Importance), который показывает на сколько пользователю необходимо выполнить поставленную задачу.

- Задачи (Todo) связаны с Группами (Group), которые используются пользователем для распределения своих задач.
- Задачи (Todo) связаны с Видом таймеров (Timer_type), который определяет какой таймер пользователь хочет использовать для данной работы.
- Задачи (Todo) связаны с Видом задачи (Todo_type), определяющим каким образом пользователь желает выполнить поставленную им задачу.
- Календарь задачи (Todo_calendar) связан с Задачей (Todo), для которой он был создан.

Использующие прецеденты: Создание задачи.

<u>Операция 14.1</u> Сохранение нового таймера (todo_id, название, описание, время)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.
- Задача создана.

Последующие условия:

- Создан новый Таймер (Timer).
- Timer.name=название.
- Timer.description=описание.
- Timer.time=время.
- Timer.todo_id=todo_id (id задачи).
- Таймер (Timer) связан с Задачей (Todo), для которой он был создан.

Использующие прецеденты: Создание таймера.

<u>Операция 17.1</u> Изменение состояния задачи (todo_history_id, state_id)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.
- Задача создана.
- Состояния задач внесены в базу данных.

Последующие условия:

- Todo_history.state_id=state_id.
- История задач (Todo_history) связана с Состоянием задач (State), которое определяет уровень завершённости задачи.
- История задач (Todo_history) связана с Задачей (Todo), данные которой сохраняются.

Использующие прецеденты: Изменение состояния задач.

<u>Операция 18.1</u> Изменение даты выполнения задачи (todo_id, новая дата выполнения задачи)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.
- Задача создана.

Последующие условия:

- Todo_calendar.date=новая дата выполнения задачи.
- Календарь задачи (Todo_calendar) связан с Задачей (Todo), для которой он был создан.

Использующие прецеденты: Перенесение сроков выполнения задачи.

<u>Операция 19.1</u> Удаление задачи (todo_id)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.
- Задача создана.

Последующие условия:

- Задача (Todo) удалена.
- Недельное расписание задачи (Todo_week) связанное с Задачей (Todo) удалено.
- Календарь задачи (Todo_calendar) связанный с Задачей (Todo) удалён.
- Таймер (Timer) связанный с Задачей (Todo) удалён.

Использующие прецеденты: Удаление задачи.

<u>Операция 20.1</u> Создание нового однодневного напоминания (user_id, название напоминания, описание, время повторения, название файла картинки, sound_id, reminder_type_id, дата появления напоминания, время появления напоминания)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.
- Названия звуков зарегистрированы в базе данных.
- Виды напоминаний зарегистрированы в базе данных.

Последующие условия:

- Напоминание (Reminder) создано.
- Reminder.name=название напоминания.
- Reminder.description=описание.
- Reminder.repeat=время повторения.
- Reminder.picture=название файла картинки.

- Reminder.user_id=user_id (id пользователя).
- Reminder.sound_id=sound_id (id звука).
- Reminder.reminder_type_id=reminder_type_id (id вида напоминания).
- Напоминание (Reminder) связано с Пользователем (User), который создаёт напоминание.
- Напоминание (Reminder) связано со Звуком (Sound), который используется системой при появлении напоминания.
- Напоминание (Reminder) связано с Типом напоминания (Reminder_type), который определяет вид напоминания и то, как оно должно быть использовано.
- Календарь напоминания (Reminder_calendar_time) создан.
- Reminder_calendar_time.date=дата появления напоминания.
- Reminder_calendar_time.time=время появления напоминания.
- Reminder_calendar_time.reminder_id=reminder_id (id).
- Календарь напоминания (Reminder_calendar_time) связан с Напоминанием (Reminder) для которого он был создан.

Использующие прецеденты: Создание напоминания.

<u>Операция 20.2</u> Создание ежедневного напоминания (user_id, название напоминания, описание, время повторения, название файла картинки, sound_id, reminder_type_id, время появления напоминания для понедельника, время появления напоминания для вторника, время появления напоминания для среды, время появления напоминания для четверга, время появления напоминания для пятницы, время появления напоминания для субботы, время появления напоминания для воскресенья)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.
- Названия звуков зарегистрированы в базе данных.
- Виды напоминаний зарегистрированы в базе данных.

Последующие условия:

- Недельное расписание напоминания (Reminder_week_time) создано.
- Reminder_week_time.monday=время появления напоминания для понедельника**.**
- Reminder_week_time.tuesday=время появления напоминания для вторника**.**
- Reminder_week_time.wednesday=время появления напоминания для среды**.**
- Reminder_week_time.thursday=время появления напоминания для четверга
- Reminder_week_time.friday=время появления напоминания для пятницы**.**

- Reminder_week_time.saturday=время появления напоминания для субботы**.**
- Reminder_week_time.sunday=время появления напоминания для воскресенья**.**
- Напоминание (Reminder) создано.
- Reminder.name=название напоминания.
- Reminder.description=описание.
- Reminder.repeat=время повторения.
- Reminder.picture=название файла картинки.
- Reminder.user_id=user_id (id пользователя).
- Reminder.sound_id=sound_id (id звука).
- Reminder.reminder_type_id=reminder_type_id (id вида напоминания).
- Напоминание (Reminder) связано с Пользователем (User), который создаёт напоминание.
- Напоминание (Reminder) связано со Звуком (Sound), который используется системой при появлении напоминания.
- Напоминание (Reminder) связано с Типом напоминания (Reminder_type), который определяет вид напоминания и то, как оно должно быть использовано.
- Недельное расписание напоминания (Reminder_week_time) связано с Напоминанием (Reminder), для которого было создано расписание появления напоминания на неделю.

Использующие прецеденты: Создание напоминания.

Операция 22.1 Удаление напоминания (reminder_id)

Предварительные условия:

- Пользователь зарегистрирован и авторизован в системе.
- Напоминание создано.

Последующие условия:

- Напоминание (Reminder) удалено.
- Календарь напоминания (Reminder_calendar_time) связанный с Напоминанием (Reminder) удалён.
- Недельное расписание напоминаний (Reminder_week_time) связанное с Напоминанием (Reminder) удалено.

Использующие прецеденты: Удаление напоминания.

5.3. CRUD

Данная таблица CRUD показывает отношение между классами-сущностями и прецедентами.

Прецедент **Класс-сущность**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**Итог**
Group													C R		R									CR
Importance													R		R									R
Program_calendar _working_time						R D	C R	R	R			R D												CR D
Program_control_rule						R D	C	R	R	R U	R	R D												CR UD
Program_rule_type							R	R																R
Program_week _working_time						R D	C R	R	R		R U	R D												CR UD
Program_working _history						R D			R U														R	RU D
Questions	R			R																				R
Reminder						R D														C	R	R D		CR D
Reminder_calendar _time						R D														C R	R	R D		CR D
Reminder_type																				R	R			R
Reminder_week_time						R D														C R	R	R D		CR D
Settings	C	R			U	R D																		CR UD
Sound																				R	R			R
State															R		R							R
Tag													C R		R									CR

Прецедент **Класс-сущность**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**Итог**
Timer						R D								C	R	R			R D					CR D
Timer_type														R	R	R			R					R
Todo						R D							C	R	R		R	R	R D					CR D
Todo_calendar						R D							C R		R			R U	R D					CR UD
Todo_history						R D									R		R U	R D					R	RU D
Todo_type													R		R			R	R					R
Todo_week						R D							C R		R				R D					CR D
User	C R	R	R U	R	R U	R D	R	R							R		R			R	R		R	CR UD
Classifier							R	R					R	R	R	R	R	R	R	R	R			R

Таблица 4. CRUD.

Список прецедентов:

1. Регистрация пользователя.
2. Идентификация пользователя.
3. Изменение личных данных пользователя.
4. Восстановление пароля.
5. Изменение настроек пользователя.
6. Удаление пользователя.
7. Создание правила контроля.
8. Просмотр списка контролируемых программ.
9. Продление времени работы программы.
10. Изменение времени показа оповещения.
11. Изменение расписания работы программы.
12. Удаление правила контроля.
13. Создание задачи.
14. Создание таймера.
15. Просмотр списка задач.
16. Запуск таймера.
17. Изменение состояния задач.
18. Перенесение сроков выполнения задачи.
19. Удаление задачи.
20. Создание напоминания.
21. Просмотр списка напоминаний.
22. Удаление напоминания.
23. Просмотр статистики.

5.4. Определения атрибутов.

Название класса-сущности	*Название атрибута*	*Определение атрибута*	*Возможное значение*
User	login	Имя пользователя, которое будет использовано системой во время работы с программой. Логин не может быть короче двух символов.	Алёна Гвоздева
User	email	Адрес электронной почты пользователя, куда, в случае необходимости, будут отправлены данные для восстановления пароля. Адрес должен иметь следующую структуру: текст@текст.текст	alg@gmail.com
User	password	Идентифицирующий пользователя текст, известный только пользователю. Пароль не должен быть короче 6 символов и может состоять из латинских букв и/или цифр.	GKNH7vdse
User	picture	Картинка, которая будет использоваться системой для обозначения скрытого меню пользователя.	

Название класса-сущности	*Название атрибута*	*Определение атрибута*	*Возможное значение*
User	answer	Ответ, который даёт пользователь на вопрос самоконтроля. Сочетание вопроса/ответа должно быть известно только пользователю. Ответ не может быть пустым или же состоять из пробелов.	Рыжакова
Questions	name	Вопрос, который будет использован системой для проверки пользователя, в случае если пользователь забыл свой пароль.	Девичья фамилия Вашей матери
Settings	stay_logged	Значение, отображающее выбор пользователя касательно его постоянной авторизации в системе при запуске программы.	1
Settings	use_password	Значение, отображающее желание пользователя использовать пароль при входе в систему.	0
Program_control_rule	name	Название программы, для которой было создано правило контроля.	Bejeweled3.exe
Program_control_rule	notification	Время появления оповещения до конца работы программы, для которой было создано правило контроля.	05:00

Название класса-сущности	*Название атрибута*	*Определение атрибута*	*Возможное значение*
Classifier	name	Значение классификатора, отвечающее за название. Не может быть пустым или состоять из пробелов.	Полнеценное
Classifier	description	Значение классификатора, отвечающее за описание. Не может состоять из одних пробелов.	Создание полноценного правила, которое будет ограничивать работу выбранной программы. Имеет низкий приоритет.
Program_week_week_working_time	monday	Разрешённое время работы программы для Понедельника.	03:30
Program_week_week_working_time	tuesday	Разрешённое время работы программы для Вторника.	03:30
Program_week_week_working_time	wednesday	Разрешённое время работы программы для Среды.	01:00
Program_week_week_working_time	thursday	Разрешённое время работы программы для Четверга.	03:30
Program_week_week_working_time	friday	Разрешённое время работы программы для Пятницы.	03:30
Program_week_week_working_time	saturday	Разрешённое время работы программы для Субботы.	02:00

Название класса-сущности	*Название атрибута*	*Определение атрибута*	*Возможное значение*
Program_week_week_working_time	sunday	Разрешённое время работы программы для Воскресенья.	02:00
Program_calendar_working_time	date	День, для которого будет создано правило контроля.	15.06.2015
Program_calendar_working_time	allowed_to_work	Время, которое будет разрешено программе работать в выбранный день.	01:30
Program_working_history	worked_time	Время, которая программа проработала в данный день.	04:15
Program_working_history	added_time	Время, на которое было увеличено разрешенное время работы в данный день.	01:30
Program_working_history	date	День, в который осуществлялся контроль работы программы.	04.08.2015
Reminder	name	Название напоминания. Название не может быть пустым или же состоять из пробелов.	Отправить отчёт
Reminder	description	Описание напоминания.	Необходимо предоставить варианты логотипов, а также макеты сайта.
Reminder	repeat	Время, через которое будет снова показано отложенное напоминание.	00:10
Reminder	picture	Картинка, которая будет использована системой при отображении напоминания.	

Название класса-сущности	*Название атрибута*	*Определение атрибута*	*Возможное значение*
Sound	name	Название звука, который будет использован системой при отображении напоминания.	Звук - Бурление Воды.mp3
Sound	description	Название, которое будет показано пользователю при выборе звука для напоминания.	Звук бурлящей воды.
Reminder_calendar_time	date	Число, когда будет необходимо отобразить напоминание.	04.08.2015
Reminder_calendar_time	time	Время, в которое будет необходимо отобразить напоминание.	15:00
Reminder_week_time	monday	Время для появления напоминания в Понедельник.	13:00
Reminder_week_time	tuesday	Время для появления напоминания во Вторник.	13:00
Reminder_week_time	wednesday	Время для появления напоминания в Среду.	13:00
Reminder_week_time	thursday	Время для появления напоминания в Четверг.	13:00
Reminder_week_time	friday	Время для появления напоминания в Пятницу.	13:00
Reminder_week_time	saturday	Время для появления напоминания в Субботу.	10:00
Reminder_week_time	sunday	Время для появления напоминания в Воскресенье.	10:00

Название класса-сущности	*Название атрибута*	*Определение атрибута*	*Возможное значение*
Todo	name	Название задачи. Название задачи не может быть пустым или же состоять из пробелов.	Посетить зоопарк
Todo	description	Описание задачи.	Не забыть взять фотоаппарат
Todo_history	actual_date	Дата, на которую была запланирована задача.	15.07.2015
Tag	name	Метки используются пользователем для обозначения ключевых слов, относящихся к данной задаче. Они используются для облегчения последующего поиска задач. Для одной задачи возможно использовать до 5 меток. Название метки не может быть пустым или же состоять из пробелов.	3D
Group	name	Различные группы присваиваются задачам для возможности распределения задач по разным сферам. Название группы не может быть пустым или же состоять из пробелов.	Дом

Название класса-сущности	*Название атрибута*	*Определение атрибута*	*Возможное значение*
Group	color	Цвет, который будет использован в программе для обозначения группы. Код цвета должен быть уникальным.	#808ea9
Todo_week	monday	Время выполнения задачи в Понедельник.	09:00
Todo_week	tuesday	Время выполнения задачи во Вторник.	09:00
Todo_week	wednesday	Время выполнения задачи в Среду	09:00
Todo_week	thursday	Время выполнения задачи в Четверг	09:00
Todo_week	friday	Время выполнения задачи в Пятницу.	09:00
Todo_week	saturday	Время выполнения задачи в Субботу.	09:00
Todo_week	sunday	Время выполнения задачи в Воскресенье.	09:00
Todo_calendar	date	Дата, на которую запланирована задача.	05.07.2015
Todo_calendar	time	Время начала выполнения задачи.	18:15
Timer	name	Название таймера. Название таймера не может быть пустым или же состоять из пробелов.	Подготовка
Timer	description	Описание действия, для которого был создан таймер.	Использовать праймер
Timer	time	Длительность работы таймера.	00:01

Таблица 5. Определения атрибутов.

5.5. Детальное описание таблиц.

Group

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
group_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
name	Текст	100	Не должно состоять из пробелов.		Да.
color	Текст	7	Код цвета, состоящий из # и 6 цифр.		Да.

Таблица 6. Детальное описание таблицы Group.

Primary Key (group_id)

Alternate Key (name, color)

Importance

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
importance_id	Целое число		См. значения классификаторов.		Да.

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
name	Текст	11	См. значения классификаторов.		Да.
description	Текст	100			Да.

Таблица 7. Детальное описание таблицы Importance.

Primary Key (importance_id)

Alternate Key (name)

Program_calendar_working_time

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
program_calendar _working_time_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
date	Дата				Да.
allowed_to_work	Целое число				Да.

Таблица 8. Детальное описание таблицы Program_calendar_working_time.

Primary Key (program_calendar_working_time_id)

Program_control_rule

Название столбца	**Название типа**	**Длина**	**Возможные значения**	**Значение по умолчанию**	**Обязательно**
program_control _rule_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
name	Текст	100			Да.
notification	Целое число				Нет.
program_rule _type_id	Целое число				Да.
program_week _working_time_id	Целое число				Нет.
program_calender _working_time_id	Целое число				Нет.
user_id	Целое число				Да.

Таблица 9. Детальное описание таблицы Program_control_rule.

Primary Key (importance_id)

Foreign Key (program_rule_type_id) **REFERENCES** Program_rule_type (program_rule_type_id) **ON DELETE RESTRICT**

Foreign Key (program_week_working_time_id) **REFERENCES** Program_week_working_time (program_week_working_time_id) **ON DELETE RESTRICT**

Foreign Key (program_calendar_working_time_id) **REFERENCES** Program_calendar_working_time (program_calendar_working_time_id) **ON DELETE RESTRICT**

Foreign Key (user_id) **REFERENCES** User (user_id) **ON DELETE CASCADE**

Program_rule_type

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
program_rule_type_id	Целое число		См. значения классификаторов.		Да.
name	Текст	20	См. значения классификаторов.		Да.
description	Текст	100			Да.

Таблица 10. Детальное описание таблицы Program_rule_type.

Primary Key (program_rule_type_id)

Alternate Key (name)

Program_week_working_time

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
program_week_working_time_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
monday	Целое число				Нет.
tuesday	Целое число				Нет.
wednesday	Целое число				Нет.
thursday	Целое число				Нет.
friday	Целое число				Нет.
saturday	Целое число				Нет.
sunday	Целое число				Нет.

Таблица 11. Детальное описание таблицы Program_week_working_time.

Primary Key (program_week_working_time_id)

Program_working_history

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
Program_working_history_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
worked_time	Целое число				Да.
added_time	Целое число				Нет.
date	Дата				Да.
program_control_rule_id	Целое число				Да.

Таблица 12. Детальное описание таблицы Program_working_history.

Primary Key (program_working_history_id)

Foreign Key (program_control_rule_id) **REFERENCES** Program_control_rule (program_control_rule_id) **ON DELETE CASCADE**

Questions

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
question_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
name	Текст	250			Да.

Таблица 13. Детальное описание таблицы Questions.

Primary Key (question_id)

Alternate Key (name)

Reminder

Название столбца	**Название типа**	**Длина**	**Возможные значения**	**Значение по умолчанию**	**Обязательно**
reminder_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
name	Текст	100			Да.
description	Текст	2000			Нет.
repeat	Целое число				Нет.
picture	Текст	250			Нет.
user_id	Целое число				Да.
sound_id	Целое число				Да.
reminder_type _id	Целое число				Да.

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
reminder_week_time_id	Целое число				Нет.
reminder_calendar_time_id	Целое число				Нет.

Таблица 14. Детальное описание таблицы Reminder.

Primary Key (reminder_id)

Foreign Key (user_id) **REFERENCES** User (user_id) **ON DELETE CASCADE**

Foreign Key (sound_id) **REFERENCES** Sound (sound_id) **ON DELETE RESTRICT**

Foreign Key (reminder_type_id) **REFERENCES** Reminder_type (reminder_type_id) **ON DELETE RESTRICT**

Foreign Key (reminder_week_time_id) **REFERENCES** Reminder_week_time_id (reminder_week_time_id) **ON DELETE SET NULL**

Foreign Key (reminder_calendar_time_id) **REFERENCES** Reminder_calendar_time (reminder_calendar_time_id) **ON DELETE SET NULL**

Reminder_calendar_time

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
reminder_calendar_time_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
date	Дата				Да.
time	Текст	5			Да.

Таблица 15. Детальное описание таблицы Reminder_calendar_time.

Primary Key (reminder_calendar_time_id)

Reminder_type

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
reminder_type_id	Целое число		См. значения классификаторов.		Да.
name	Текст	20	См. значения классификаторов.		Да.

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
description	Текст	100	Не должно состоять из пробелов.		Да.

Таблица 16. Детальное описание таблицы Reminder_type.

Primary Key (reminder_type_id)

Alternate Key (name)

Reminder_week_time

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
Reminder_week_time_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
monday	Текст	5			Нет.
tuesday	Текст	5			Нет.
wednesday	Текст	5			Нет.
thursday	Текст	5			Нет.
friday	Текст	5			Нет.

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
saturday	Текст	5			Нет.
sunday	Текст	5			Нет.

Таблица 17. Детальное описание таблицы Reminder_week_time.

Primary Key (reminder_week_time_id)

Settings

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
settings_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
stay_logged	Булевое значение				Да.
use_password	Булевое значение				Да.
user_id	Целое число				Да.

Таблица 18. Детальное описание таблицы Settings.

Primary Key (settings_id)

Foreign Key (user_id) **REFERENCES** User (user_id) **ON DELETE CASCADE**

Sound

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
sound_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
name	Текст	250			Да.
description	Текст	100			Да.

Таблица 19. Детальное описание таблицы Sound.

Primary Key (sound_id)

Alternate Key (name)

State

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
state_id	Целое число		См. значения классификаторов.		Да.
name	Текст	18	См. значения классификаторов.		Да.
description	Текст	100	Не должно состоять из пробелов.		Да.

Таблица 20. Детальное описание таблицы State.

Primary Key (state_id)

Alternate Key (name)

Tag

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
tag_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
name	Текст	50	Не должно состоять из пробелов.		Да.

Таблица 21. Детальное описание таблицы Tag.

Primary Key (tag_id)

Alternate Key (name)

Tag_todo

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
todo_id	Целое число				Да.
tag_id	Целое число				Да.

Таблица 22. Детальное описание таблицы Tag_todo.

Primary Key (todo_id,tag_id)

Foreign Key (todo_id) **REFERENCES** Todo (todo_id) **ON DELETE CASCADE**

Foreign Key (tag_id) **REFERENCES** Tag (tag_id) **ON DELETE RESTRICT**

Timer

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
timer_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
name	Текст	100	Не должно состоять из пробелов.		Да.
description	Текст	2000			Нет.
time	Целое число				Да.
todo_id	Целое число				Да.

Таблица 23. Детальное описание таблицы Timer.

Primary Key (timer_id)

Foreign Key (todo_id) **REFERENCES** Todo (todo_id) **ON DELETE CASCADE**

Timer_type

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
timer_type_id	Целое число		См. значения классификаторов.		Да.
name	Текст	14	См. значения классификаторов.		Да.
description	Текст	100			Да.

Таблица 24. Детальное описание таблицы Timer_type.

Primary Key (timer_type_id)

Alternate Key (name)

Todo

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
todo_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
user_id	Целое число				Да.

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
name	Текст	100	Не должно состоять из пробелов.		Да.
description	Текст	2000	Не должно состоять из пробелов.		Нет.
importance_id	Целое число				Да.
group_id	Целое число				Да.
timer_type_id	Целое число				Да.
todo_type_id	Целое число				Да.
todo_week_id	Целое число				Нет.
todo_calendar_id	Целое число				Нет.

Таблица 25. Детальное описание таблицы Todo.

Primary Key (todo_id)

Foreign Key (user_id) **REFERENCES** User (user_id) **ON DELETE CASCADE**

Foreign Key (importance_id) **REFERENCES** Importance (importance_id) **ON DELETE RESTRICT**

Foreign Key (group_id) **REFERENCES** Group (group_id) **ON DELETE RESTRICT**

Foreign Key (timer_type_id) **REFERENCES** Timer_type (timer_type_id) **ON DELETE RESTRICT**

Foreign Key (todo_type_id) **REFERENCES** Todo_type (todo_type_id) **ON DELETE RESTRICT**

Foreign Key (todo_week_id) **REFERENCES** Todo_week (todo_week_id) **ON DELETE SET NULL**

Foreign Key (todo_calendar_id) **REFERENCES** Todo_calendar (todo_calendar_id) **ON DELETE SET NULL**

Todo_calendar

Название столбца	**Название типа**	**Длина**	**Возможные значения**	**Значение по умолчанию**	**Обязательно**
todo_calendar_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
date	Дата				Да.
time	Текст	5			Да.

Таблица 26. Детальное описание таблицы Todo_calendar.

Primary Key (todo_calendar_id)

Todo_history

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
todo_history_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
actual_date	Дата				Да.
todo_id	Целое число				Нет.
state_id	Целое число				Да.

Таблица 27. Детальное описание таблицы Todo_history.

Primary Key (todo_history_id)

Foreign Key (todo_id) **REFERENCES** Todo (todo_id) **ON DELETE SET NULL**

Foreign Key (state_id) **REFERENCES** State (state_id) **ON DELETE RESTRICT**

Todo_type

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
todo_type_id	Целое число		См. значения классификаторов.		Да.
name	Текст	20	См. значения классификаторов.		Да.

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
description	Текст	100			Да.

Таблица 28. Детальное описание таблицы todo_type.

Primary Key (todo_type_id)

Alternate Key (name)

Todo_week

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
todo_week_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
monday	Текст	5			Нет.
tuesday	Текст	5			Нет.
wednesday	Текст	5			Нет.
thursday	Текст	5			Нет.
friday	Текст	5			Нет.
saturday	Текст	5			Нет.

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
sunday	Текст	5			Нет.

Таблица 29. Детальное описание таблицы Todo_week.

Primary Key (todo_week_id)

User

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
user_id	Целое число		Уникальное, автоматически генерируемое системой число. Длинна шага = 1.		Да.
login	Текст	100	Логин должен быть не короче двух символов, не может состоять из пробелов.		Да.
email	Текст	100	Почта должна иметь следующую структуру: текст@текст.текст		Да.

Название столбца	Название типа	Длина	Возможные значения	Значение по умолчанию	Обязательно
password	Текст	16	Не может содержать пробелы, должен быть не короче 6 и не длинее 16 символов.		Нет.
picture	Текст	250			Да.
answer	Текст	250	Не может состоять из пробелов.		Да.
question_id	Целое число				Да.

Таблица 30. Детальное описание таблицы User.

Primary Key (user_id)

Alternate Key (login)

Foreign Key (question_id) **REFERENCES** Questions (question_id) **ON DELETE RESTRICT**

5.6. SQL код.

5.6.1. Создание таблиц.

```
CREATE TABLE "Questions"(
question_id INTEGER PRIMARY KEY NOT NULL,
name VARCHAR(250) UNIQUE NOT NULL);
```

```
CREATE TABLE "User" (
user_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
login VARCHAR(100) UNIQUE NOT NULL,
email VARCHAR(100) NOT NULL,
password VARCHAR(16),
picture VARCHAR(250) NOT NULL,
answer VARCHAR(250) NOT NULL,
question_id INTEGER NOT NULL,
FOREIGN KEY(question_id) REFERENCES Questions(question_id) ON DELETE
RESTRICT);


CREATE TABLE "Tag"(
tag_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
name VARCHAR(50) NOT NULL);


CREATE TABLE "Group"(
group_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
name VARCHAR(100) UNIQUE NOT NULL,
color VARCHAR(7) UNIQUE NOT NULL);


CREATE TABLE "Importance"(
importance_id INTEGER PRIMARY KEY NOT NULL,
name VARCHAR(11) UNIQUE NOT NULL,
description VARCHAR(100) NOT NULL);


CREATE TABLE "Timer_type"(
timer_type_id INTEGER PRIMARY KEY NOT NULL,
name VARCHAR(14) UNIQUE NOT NULL,
description VARCHAR(100) NOT NULL);
```

```
CREATE TABLE "Todo_week"(
todo_week_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
monday VARCHAR(5),
tuesday VARCHAR(5),
wednesday VARCHAR(5),
thursday VARCHAR(5),
friday VARCHAR(5),
saturday VARCHAR(5),
sunday VARCHAR(5));

CREATE TABLE "Todo_calendar"(
todo_calendar_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
date DATE NOT NULL,
time VARCHAR(5));

CREATE TABLE "Todo_type"(
todo_type_id INTEGER PRIMARY KEY NOT NULL,
name VARCHAR(20) NOT NULL,
description VARCHAR(100) NOT NULL);

CREATE TABLE "State"(
state_id INTEGER PRIMARY KEY NOT NULL,
name VARCHAR(18) UNIQUE NOT NULL,
description VARCHAR(100) NOT NULL);

CREATE TABLE "Todo"(
todo_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
user_id INTEGER NOT NULL,
name VARCHAR(100) NOT NULL,
description VARCHAR(2000),
importance_id INTEGER NOT NULL,
group_id INTEGER NOT NULL,
timer_type_id INTEGER NOT NULL,
```

```
todo_type_id INTEGER NOT NULL,
todo_week_id INTEGER,
todo_calendar_id INTEGER,
FOREIGN KEY(user_id) REFERENCES "User"(user_id) ON DELETE CASCADE,
FOREIGN KEY(importance_id) REFERENCES "Importance"(importance_id) ON DELETE
RESTRICT,
FOREIGN KEY(group_id) REFERENCES "Group"(group_id) ON DELETE RESTRICT,
FOREIGN KEY(timer_type_id) REFERENCES "Timer_type"(timer_type_id) ON DELETE
RESTRICT,
FOREIGN KEY(todo_type_id) REFERENCES "Todo_type"(todo_type_id) ON DELETE
RESTRICT,
FOREIGN KEY(todo_week_id) REFERENCES "Todo_week"(todo_week_id) ON DELETE
SET NULL,
FOREIGN KEY(todo_calendar_id) REFERENCES "Todo_calendar"(todo_calendar_id) ON
DELETE SET NULL);

CREATE TABLE "Todo_history"(
todo_history_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
actual_date DATE NOT NULL,
todo_id INTEGER,
state_id INTEGER NOT NULL,
FOREIGN KEY(todo_id) REFERENCES "Todo"(todo_id) ON DELETE SET NULL,
FOREIGN KEY(state_id) REFERENCES "State"(state_id) ON DELETE RESTRICT);

CREATE TABLE "Tag_todo"(
todo_id INTEGER NOT NULL,
tag_id INTEGER NOT NULL,
PRIMARY KEY(todo_id,tag_id),
FOREIGN KEY(todo_id) REFERENCES "Todo"(todo_id) ON DELETE CASCADE,
FOREIGN KEY(tag_id) REFERENCES "Tag"(tag_id) ON DELETE RESTRICT);
```

```
CREATE TABLE "Settings"(
settings_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
stay_logged BOOLEAN NOT NULL,
use_password BOOLEAN NOT NULL,
user_id INTEGER NOT NULL,
FOREIGN KEY(user_id) REFERENCES "User"(user_id) ON DELETE CASCADE);


CREATE TABLE "Timer"(
timer_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
name VARCHAR(100) NOT NULL,
description VARCHAR(2000),
time INTEGER NOT NULL,
todo_id INTEGER NOT NULL,
FOREIGN KEY(todo_id) REFERENCES "Todo"(todo_id) ON DELETE CASCADE);


CREATE TABLE "Reminder_type"(
reminder_type_id INTEGER PRIMARY KEY NOT NULL,
name VARCHAR(20) UNIQUE NOT NULL,
description VARCHAR(100) NOT NULL);


CREATE TABLE "Sound"(
sound_id INTEGER PRIMARY KEY NOT NULL,
name VARCHAR(250) UNIQUE NOT NULL,
description VARCHAR(100) NOT NULL);


CREATE TABLE "Reminder_week_time"(
reminder_week_time_id INTEGER PRIMARY KEY NOT NULL,
monday VARCHAR(5),
tuesday VARCHAR(5),
wednesday VARCHAR(5),
thursday VARCHAR(5),
friday VARCHAR(5),
saturday VARCHAR(5),
```

```
sunday VARCHAR(5));

CREATE TABLE "Reminder_calendar_time"(
reminder_calendar_time_id INTEGER PRIMARY KEY NOT NULL,
date DATE NOT NULL,
time VARCHAR(5) NOT NULL);

CREATE TABLE "Reminder"(
reminder_id INTEGER PRIMARY KEY NOT NULL,
name VARCHAR(100) NOT NULL,
description VARCHAR(2000),
repeat INTEGER,
picture VARCHAR(250),
user_id INTEGER NOT NULL,
sound_id INTEGER NOT NULL,
reminder_type_id INTEGER NOT NULL,
reminder_week_time_id INTEGER,
reminder_calendar_time_id INTEGER,
FOREIGN KEY(user_id) REFERENCES "User"(user_id) ON DELETE CASCADE,
FOREIGN KEY(sound_id) REFERENCES "Sound"(sound_id) ON DELETE RESTRICT,
FOREIGN KEY(reminder_type_id) REFERENCES "Reminder_type"(reminder_type_id) ON
DELETE RESTRICT,
FOREIGN KEY(reminder_week_time_id) REFERENCES
"Reminder_week_time_id"(reminder_week_time_id) ON DELETE SET NULL,
FOREIGN KEY(reminder_calendar_time_id) REFERENCES
"Reminder_calendar_time"(reminder_calendar_time_id) ON DELETE SET NULL);

CREATE TABLE "Program_rule_type"(
program_rule_type_id INTEGER PRIMARY KEY NOT NULL,
name VARCHAR(20) UNIQUE NOT NULL,
description VARCHAR(100) NOT NULL);
```

```
CREATE TABLE "Program_week_working_time"(
program_week_working_time_id INTEGER PRIMARY KEY NOT NULL,
monday INTEGER,
tuesday INTEGER,
wednesday VINTEGER,
thursday INTEGER,
friday INTEGER,
saturday INTEGER,
sunday INTEGER);


CREATE TABLE "Program_calendar_working_time"(
program_calendar_working_time_id INTEGER PRIMARY KEY NOT NULL,
date DATE NOT NULL,
allowed_to_work INTEGER NOT NULL);


CREATE TABLE "Program_control_rule"(
program_control_rule_id INTEGER PRIMARY KEY NOT NULL,
name VARCHAR(100) NOT NULL,
notification INTEGER,
program_rule_type_id INTEGER NOT NULL,
program_week_working_time_id INTEGER,
program_calendar_working_time_id INTEGER,
user_id INTEGER NOT NULL,
FOREIGN KEY(program_rule_type_id) REFERENCES
"Program_rule_type"(program_rule_type_id) ON DELETE RESTRICT,
FOREIGN KEY(program_week_working_time_id) REFERENCES
"Program_week_working_time"(program_week_working_time_id) ON DELETE RESTRICT,
FOREIGN KEY(program_calendar_working_time_id) REFERENCES
"Program_calendar_working_time"(program_calendar_working_time_id) ON DELETE
RESTRICT,
FOREIGN KEY(user_id) REFERENCES "User"(user_id) ON DELETE CASCADE);
```

```
CREATE TABLE "Program_working_history"(
program_working_history_id INTEGER PRIMARY KEY NOT NULL,
worked_time INTEGER NOT NULL,
added_time INTEGER,
date DATE NOT NULL,
program_control_rule_id INTEGER NOT NULL,
FOREIGN KEY(program_control_rule_id) REFERENCES
"Program_control_rule"(program_control_rule_id) ON DELETE CASCADE);
```

5.6.2. Подготовка таблиц.

```
INSERT INTO "Questions"(question_id,name)
VALUES(1,'Имя лучшего друга детства');


INSERT INTO "Questions"(question_id,name)
VALUES(2,'Девичья фамилия Вашей матери');


INSERT INTO "Questions"(question_id,name)
VALUES(3,'Имя Вашего первого питомца');


INSERT INTO "Importance"(importance_id,name,description)
VALUES(1,'Очень важно','Высокая степень важности.');


INSERT INTO "Importance"(importance_id,name,description)
VALUES(2,'Важно','Средняя степень важности.');


INSERT INTO "Importance"(importance_id,name,description)
VALUES(3,'Не важно','Низкая степень важности.');


INSERT INTO "Timer_type"(timer_type_id,name,description)
VALUES(1,'Простой Таймер','Создать один таймер для задачи.');
```

```
INSERT INTO "Timer_type"(timer_type_id,name,description)
VALUES(2,'Сеть Таймеров','Создать несколько таймеров для задачи.');

INSERT INTO "Timer_type"(timer_type_id,name,description)
VALUES(3,'Pomodoro','Создать таймер Pomodoro.');

INSERT INTO "Timer_type"(timer_type_id,name,description)
VALUES(4,'none','Не создавать таймер для задачи.');

INSERT INTO "Todo_type"(todo_type_id,name,description)
VALUES(1,'Цель на день','Создать задачу на определённый день.');

INSERT INTO "Todo_type"(todo_type_id,name,description)
VALUES(2,'Ежедневная цель','Создать повторяющуюся задачу.');

INSERT INTO "Todo_type"(todo_type_id,name,description)
VALUES(3,'Долгосрочная цель','Создать задачу, которая должна быть выполнена к
определённому сроку.');

INSERT INTO "State"(state_id,name,description)
VALUES(1,'Не выполнено','Задача не выполнена.');

INSERT INTO "State"(state_id,name,description)
VALUES(2,'Частично выполнено','Задача частично выполнена.');

INSERT INTO "State"(state_id,name,description)
VALUES(3,'Выполнено','Задача выполнена.');

INSERT INTO "Group"(name,color)
VALUES('Дом','#84ddaf');
```

```
INSERT INTO "Group"(name,color)
VALUES('Работа','#89cbdd');


INSERT INTO "Group"(name,color)
VALUES('Хобби','#dd8a82');


INSERT INTO "Group"(name,color)
VALUES('Отдых','#ffbb54');
```

5.6.3. Удаление таблиц.

```
DROP TABLE "Group";
DROP TABLE "Importance";
DROP TABLE "Program_calendar_working_time";
DROP TABLE "Program_control_rule";
DROP TABLE "Program_rule_type";
DROP TABLE "Program_week_working_time";
DROP TABLE "Program_working_history";
DROP TABLE "Questions";
DROP TABLE "Reminder";
DROP TABLE "Reminder_calendar_time";
DROP TABLE "Reminder_type";
DROP TABLE "Reminder_week_time";
DROP TABLE "Settings";
DROP TABLE "Sound";
DROP TABLE "State";
DROP TABLE "Tag";
DROP TABLE "Tag_todo";
DROP TABLE "Timer";
DROP TABLE "Timer_type";
DROP TABLE "Todo";
DROP TABLE "Todo_calendar";
DROP TABLE "Todo_history";
DROP TABLE "Todo_type";
```

```
DROP TABLE "Todo_week";
DROP TABLE "User";
```