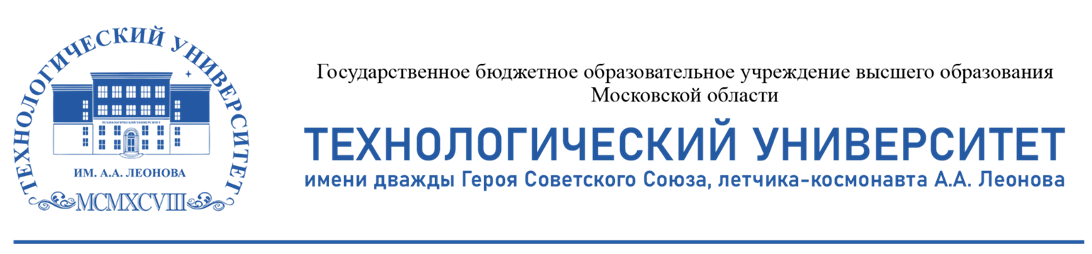
**Колледж космического машиностроения и технологии**



**ОТЧЕТ**

по производственной практике ПП.03.01

по профессиональному модулю ПМ.03 «Участие в интеграции программных модулей»

специальность 09.02.03 «Программирование в компьютерных системах»

Выполнил студен т

Ларченко Михаил Андреевич

группы: П1-18

Проверил преподаватель

Бобкова Н. Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оценка)

Королев, 2021

**Содержание**

[Введение 4](#_Toc94533825)

[1. Общие сведения о предприятии 5](#_Toc94533826)

[1.1. Сфера деятельности предприятия, история появления и развития предприятия 5](#_Toc94533827)

[1.2. Структура предприятия, функции подразделения и взаимосвязь с другими подразделениями 6](#_Toc94533828)

[1.3. Список задач подразделения в сфере информационных технологий 6](#_Toc94533829)

[2. Содержание выполняемых видов работ 7](#_Toc94533830)

[2.1. Анализ существующих в подразделении локальных и распределенных баз данных, приложений баз данных 7](#_Toc94533831)

[2.2. Задания на разработку 7](#_Toc94533832)

[2.3. Документооборот, подлежащий автоматизации 7](#_Toc94533833)

[2.4 Демонстрация IDEF-модели 8](#_Toc94533834)

[2.5. Техническое задание на разработку программного средства 9](#_Toc94533835)

[2.5.1. Наименование программы 9](#_Toc94533836)

[2.5.2. Краткая характеристика области применения 9](#_Toc94533837)

[2.5.3. Основания для разработки 9](#_Toc94533838)

[2.5.4. Функциональное назначение 9](#_Toc94533839)

[2.5.5. Эксплуатационное назначение 10](#_Toc94533840)

[2.5.7. Требования к организации входных и выходных данных 10](#_Toc94533841)

[2.5.8. Требования к временным характеристикам 10](#_Toc94533842)

[2.5.9. Требования к обеспечению надежного (устойчивого) функционирования программы 10](#_Toc94533843)

[2.5.10. Время восстановления после отказа 10](#_Toc94533844)

[2.5.11. Отказы из-за некорректных действий оператора 10](#_Toc94533845)

[2.5.12. Условия эксплуатации 10](#_Toc94533846)

[2.5.12. Требования к составу и параметрам технических средств 10](#_Toc94533847)

[2.5.13. Технико-экономические показатели 11](#_Toc94533848)

[2.5.14. Стадии и этапы разработки 11](#_Toc94533849)

[2.5.15. Порядок контроля и приемки 11](#_Toc94533850)

[2.6. Модули программного средства в соответствии с ТЗ 11](#_Toc94533851)

[2.6.1. Описание главного модуля 11](#_Toc94533852)

[2.6.2. Описание спецификаций к модулям 12](#_Toc94533853)

[2.6.3. Описание модулей 12](#_Toc94533854)

[2.6.4. Описание тестовых наборов модулей 15](#_Toc94533855)

[2.7. Пользовательский интерфейс программного средства 15](#_Toc94533856)

[2.8. Отладка и тестирование программного средства 18](#_Toc94533857)

[2.9. Комплект технологической документации 19](#_Toc94533858)

[Заключение 20](#_Toc94533859)

[Дневник производственной практики на предприятии «ККМТ» 21](#_Toc94533860)

[Дневник производственной практики на предприятии 21](#_Toc94533861)

[Приложение 1. Руководство оператора 22](#_Toc94533862)

[Приложение 2. Руководство программиста 37](#_Toc94533863)

[Приложение 3. Листинг кода 39](#_Toc94533864)

# Введение

При обучении на 4 курсе МГОТУ “Колледж космического машиностроения и технологий”, студентами группы П1-18 Ларченко Михаилом была пройдена практика по модулю ПМ.03 «Разработка программных модулей программного обеспечения для компьютерных систем».

# 1. Общие сведения о предприятии

## 1.1. Сфера деятельности предприятия, история появления и развития предприятия

Детские технопарки «Кванториум» — это федеральная сеть образовательных площадок, оснащенных высокотехнологичным оборудованием, где дети учатся по принципу проектного обучения: от теории сразу к практике.

Детский технопарк «Кванториум» — уникальная среда для ускоренного развития ребенка по актуальным научно-исследовательским и инженерно-техническим направлениям, оснащенная высокотехнологичным оборудованием. Мы обучаем детей универсальным навыкам и предметным компетенциям через решение реальных кейсов от промышленных партнеров по ключевым проблемам развития науки и техники и привитие 4K-компетенций (креативности, коммуникативности, критического мышления, умения работать в команде).

Кванториум — это школа мышления, основная цель которой — воспитать поколение специалистов, способных обеспечить технологический прорыв в нашей стране. Мы учим детей учиться: наставник не передает ребенку накопленные знания, а учит его задавать вопросы, работать с большим объемом информации и верифицировать ее.

Открытие первого в Московской области детского технопарка «Кванториум» состоялось на базе ГБОУ ВО МО «Технологический университет» в ноябре 2016 года. Под руководством высококвалифицированных наставников дети обучаются навыкам инженерных специальностей на современном высокотехнологическом оборудовании. Обучение построено по проектному принципу, на основе практических кейсов: от теории сразу к практике.

## 1.2. Структура предприятия, функции подразделения и взаимосвязь с другими подразделениями

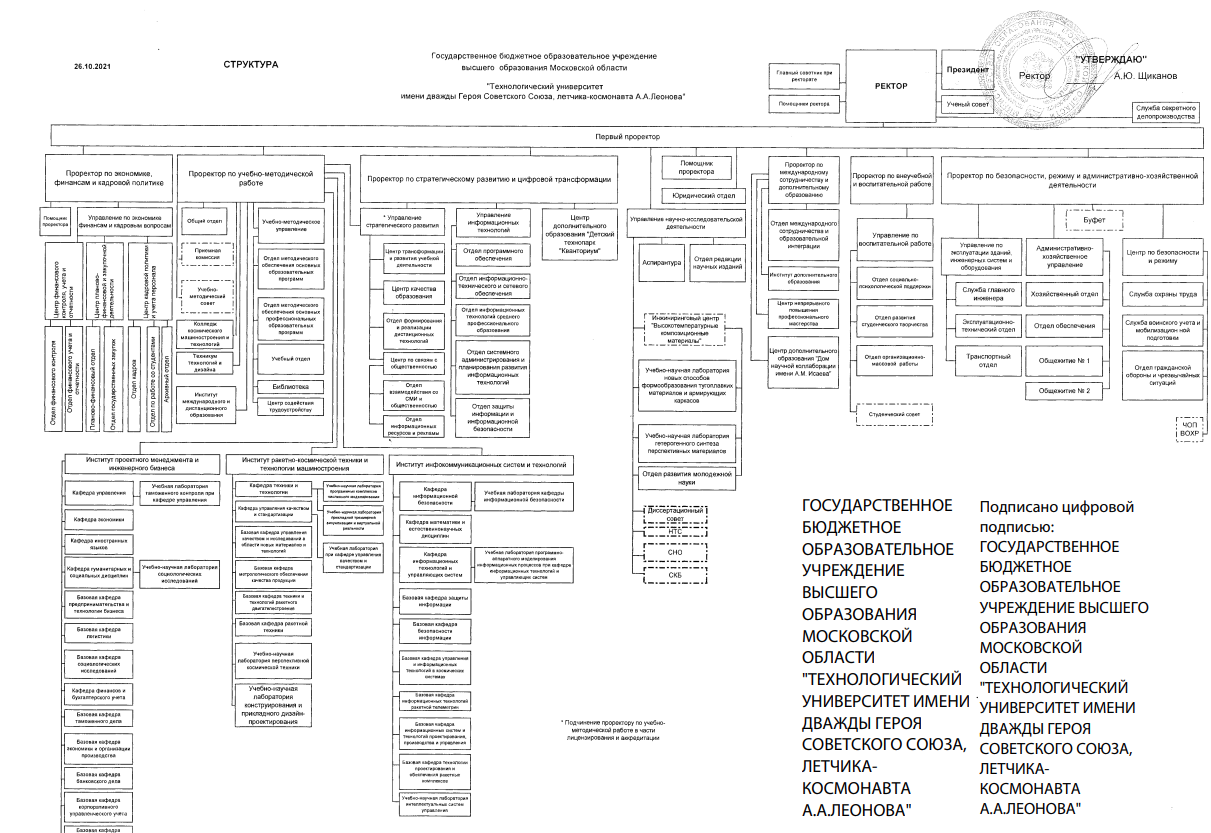


Рисунок 1.Структура предприятия

## 1.3. Список задач подразделения в сфере информационных технологий

В список задач подразделения входит создание и поддержка программ для улучшения и поддержки уровня образования ученикам в детский технопарке «Кванториум».

Разрабатывается программное обеспечение для «Кванториума», которое помогает преподавателям проверять задание на их наличие.

# Содержание выполняемых видов работ

## 2.1. Анализ существующих в подразделении локальных и распределенных баз данных, приложений баз данных

На данный момент в подразделении отсутствуют базы данных, приложения базы данных.

## 2.2. Задания на разработку

Разработать настольное приложение для проверки заданий. Программа будет состоять из интерфейса который упрощает поиск студентов и заданий. В приложении можно добавлять, удалять, и изменять задаваемые параметры в колонке студентов и заданий.

## 2.3. Документооборот, подлежащий автоматизации

Документооборот в отделе формирования и реализации дистанционных технологий представляет из себя настольное приложение для проверки заданий, а по окончанию проверки заданий, выводит отчет обо всех студентов в группе.



Рисунок 2.Документооборот

## 2.4 Демонстрация IDEF-модели

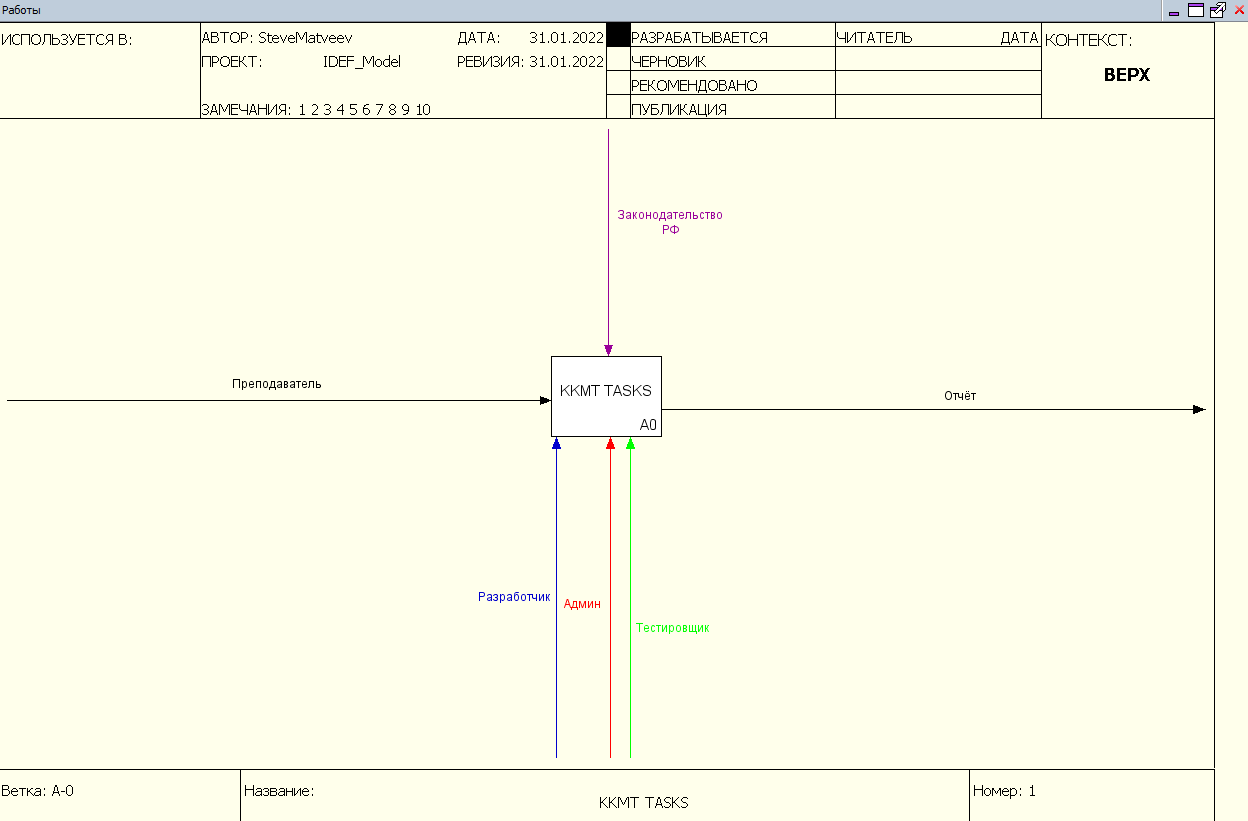


Рисунок 3.IDEF-модель 1-й уровень

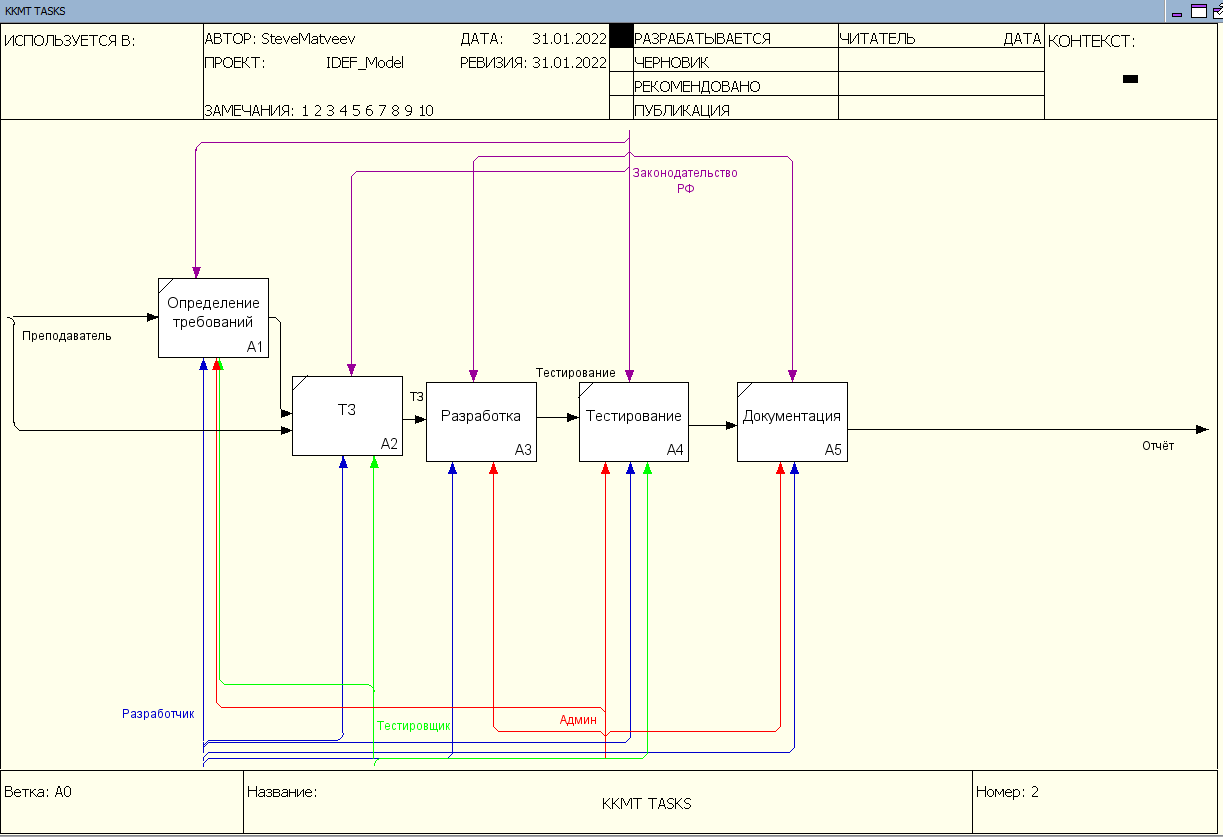


Рисунок 4.IDEF-модель 2-й уровень

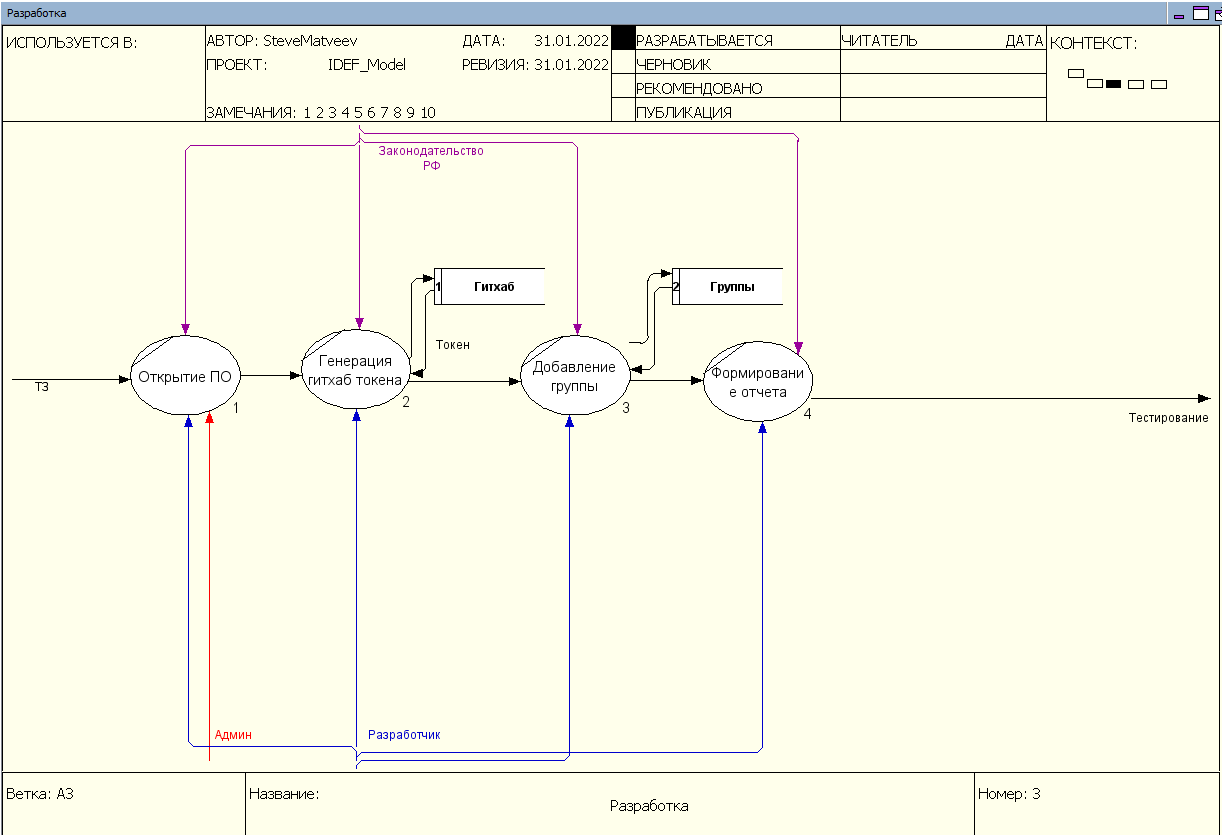


Рисунок 5. IDEF-модель 3-й уровень

## Техническое задание на разработку программного средства

### **Наименование программы**

Наименование программы – “KKMT-task 1.0.”.

### **Краткая характеристика области применения**

Необходимо разработать настольное приложение для проверки заданий. Программа будет состоять из интерфейса, который упрощает поиск студентов и заданий. В приложении можно добавлять, удалять и изменять задаваемые параметры в колонке студентов и заданий.

### **Основания для разработки**

Программа разрабатывалась, чтобы закрыть производственную практику.

### **Функциональное назначение**

Открывать базу данных учеников, получать их статус выполненных работ и сохранять отчет. И добавлять, редактированить и удалять группы, добавлять задания, редактировать и удалять задания.

### **Эксплуатационное назначение**

Облегчение проверки заданий, преподавателями.

* + 1. **Требования к составу выполняемых функции**

Программа должна содержать 1 окно. Главный экран, который содержит кнопки с верху для работы с фалом.

### **Требования к организации входных и выходных данных**

Входные данные представляют собой название и содержание текстового файла.

### **Требования к временным характеристикам**

Все операции в программе не должны блокировать графический интерфейс более чем на 1337мс.

### **Требования к обеспечению надежного (устойчивого) функционирования программы**

Программа не должна аварийно завершаться при условии наличия достаточных ресурсов на устройстве (ОЗУ, Темп. ЦП и т.д.)

### **Время восстановления после отказа**

Время восстановления после отказа, вызванным критическим состоянием устройства не должно превышать 5 минут.

### **Отказы из-за некорректных действий оператора**

При некорректных действиях оператора программа не завершиться аварийно.

### **2.5.12. Условия эксплуатации**

Программа должна запускаться как отдельная программа для ОС Windows 7 и выше, где установлен интерпретатор С#, а также компилятор Lazarus.

### **Требования к составу и параметрам технических средств**

Минимальный состав технических средств:

* ОС: Windows 7 и выше
* ОЗУ: 1 Гб
* ЦП: 1 ГГц.
* ПЗУ: 50 Мб.

### **Технико-экономические показатели**

В разные операционные системы уже как правило встроен текстовый редактор, так что данная программа не будет востребована. Программа сделана для получения опыта в данной сфере.

### **Стадии и этапы разработки**

Разработка состоит из следующих стадий:

* Разработка требований
* Проектирование
* Реализация
* Тестирование
* Внедрение

### **Порядок контроля и приемки**

После разработки всей программы происходит финальное тестирование всего функционала группой контроля качества.

## Модули программного средства в соответствии с ТЗ

### **2.6.1. Описание главного модуля**

Главный модуль состоит из главной функции, которая запускается при старте программы инициализирует класс Application и создает главную форму.

Листинг 1 Главная функция

program KkmtTesting;

{$mode objfpc}{$H+}

uses

{$IFDEF UNIX}

cthreads,

{$ENDIF}

{$IFDEF HASAMIGA}

athreads,

{$ENDIF}

Interfaces, // this includes the LCL widgetset

Forms, MainFormUnit, GroupFormUnit, taskformUnit, SettingsFormUnit;

{$R \*.res}

begin

RequireDerivedFormResource:=True;

Application.Scaled:=True;

Application.Initialize;

Application.CreateForm(TMainForm, MainForm);

Application.Run;

end.

### **2.6.2. Описание спецификаций к модулям**

Всего разработано 6 модулей:

* **MainFormUnit**
* **SettingsFormUnit**
* **TaskFormUnit**
* **GroupFormUnit**
* **AppDatabase**
* **Utils**

### **2.6.3. Описание модулей**

**MainFormUnit** – Модуль, который содержит класс главной формы программы.

Рассмотрим функцию запуска обновления выполненных веток и построим блок схему:

**Листинг 2. Функция обновления**

procedure TMainForm.UpdateStatusButtonClick(Sender: TObject);

begin

if not NowUpdating then

begin

if (SelectedGroup = -1) and (SelectedTask = -1) then exit;

Label2.Caption := StaticText2.Caption;

Label4.Caption := StaticText1.Caption;

Thread := TReportUpdateThread.Create(True, SelectedGroup, SelectedTask);

Thread.Start;

end

else

begin

Thread.StopUpdating;

end;

end;

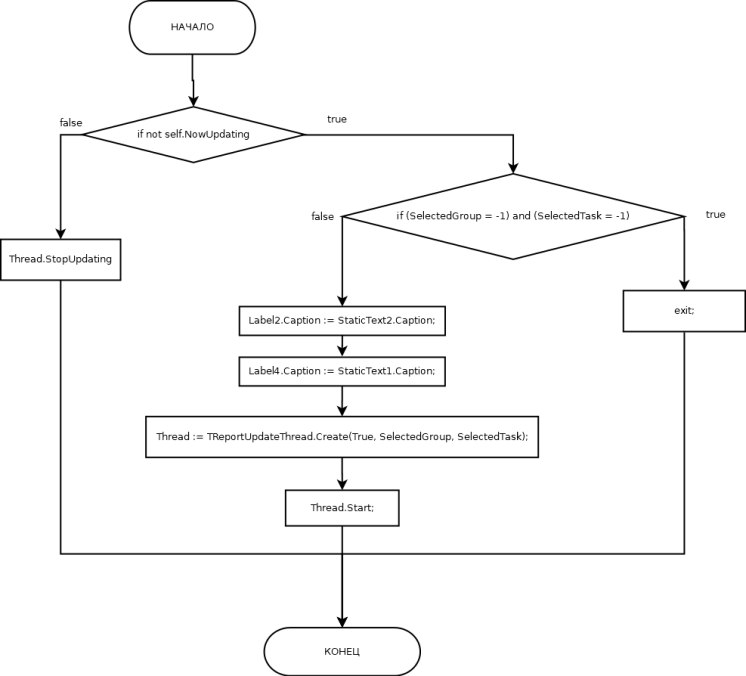


Рисунок 6.Блок схема класса MainFormUnit

**SettingsFormUnit** - Модуль с классом формы настроек программы.

Имеет два текстовых поля:

* Логин пользователя GitHub
* Персональный токен для этого логина пользователя

При нажатии кнопки "Сохранить" записывает данные в поля в файл config.json в папке программы.

**TaskFormUnit** - Модуль с классом формы для добавления, изменения заданий.

Имеет поле ввода имени задания, а также дерево с ожидаемой структурой ветки начиная от корня ветки.

Также имеет три кнопки:

* Добавить элемент дерева как ребенка выделенного элемента
* Удалить выделенный элемент
* Изменить название выделенного элемента

**GroupFormUnit** - Модуль с классом формы для редактирования, добавления групп и их репозитория для проверки.

Имеет два поля:

* Имя группы
* Уникальный идентификатор репозитория группы

При сохранении данных этой формы выполняется проверка на существование репозитория.

**AppDatabase** - Модуль содержащий классы и функции для работы с базой данных приложения.

Имеет в себе класс TAppDatabase с методами для выполнения запросов с параметрами.

* Exec - для выполнения простого SQL запроса
* ExecOut - для выполнения SQL запроса с возвращением результата

**Utils** - Хранит в себе функции различного назначения.

* CreateGitHubHTTP - для инициализации объекта TFPHttpClient для работы с GitHub API
* OpenDialog - для открытия формы в виде простого диалога

### **2.6.4. Описание тестовых наборов модулей**

В некоторых модулях производится проверка данных на корректность, чтобы сообщить пользователю о проблеме и предотвратить аварийное завершение программы.

В GroupFormUnit происходит проверка на существование репозитория при нажатии на кнопку "Button1".

## Пользовательский интерфейс программного средства

**Тест 1. Проверка имя группы**

Проверка происходит на пустоту имени группы и на пробелы. При непрохождении проверки выводится messagebox с сообщением "Имя группы не должно быть пустым или иметь пробелы".

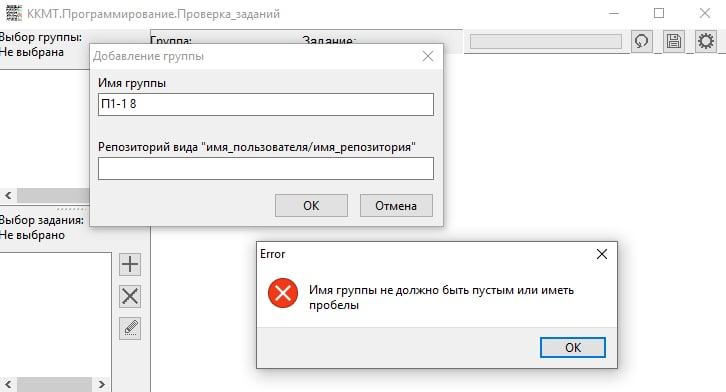


Рисунок 7.Ошибка при не правильном вводе группы

**Тест 2. Проверка заполнения вида репозитория**

Проверка на существование репозитория. Если он не существует, то будет показана ошибка "Репозиторий не найден".

Рисунок 8.Ошибка при неправильном вводе репозитория

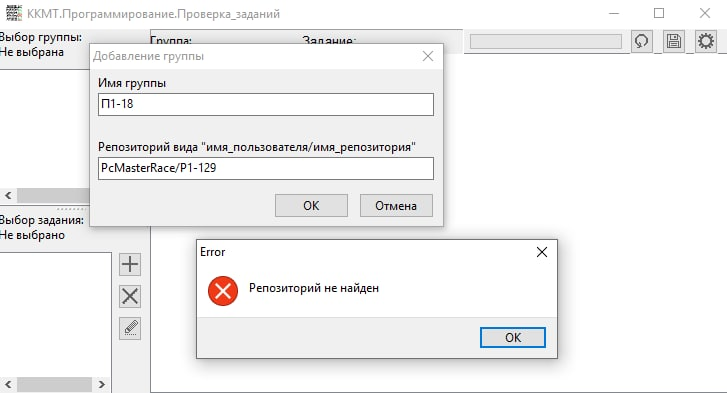
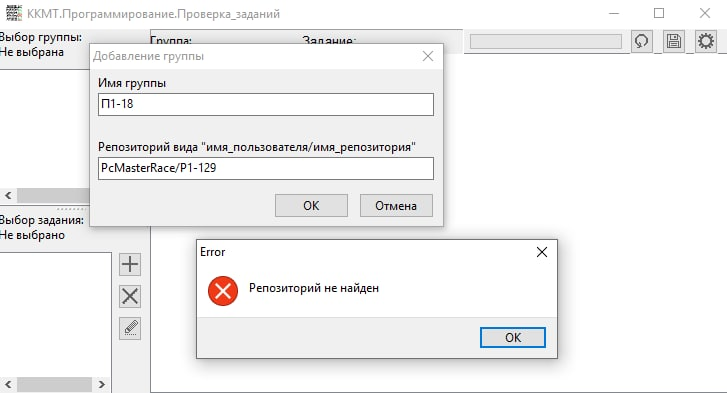


Рисунок 9.Ошибка при неправильном вводе репозитория.

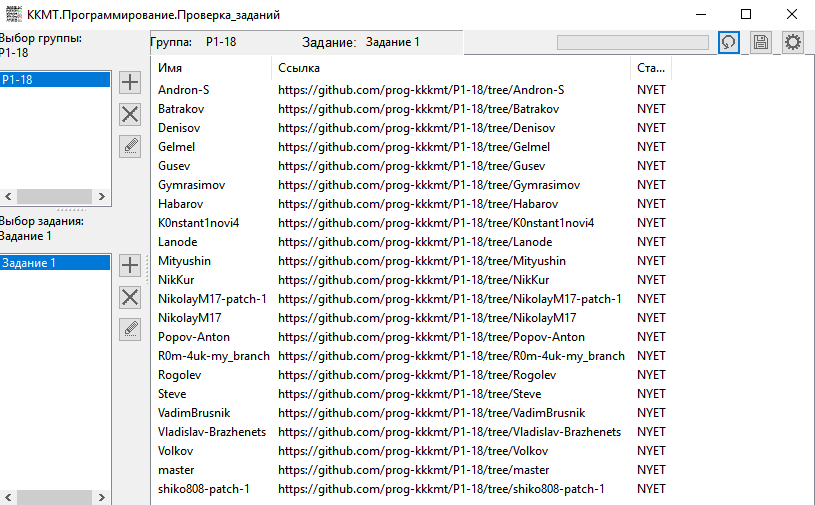


Рисунок 10.Демонстрация успешного вывода

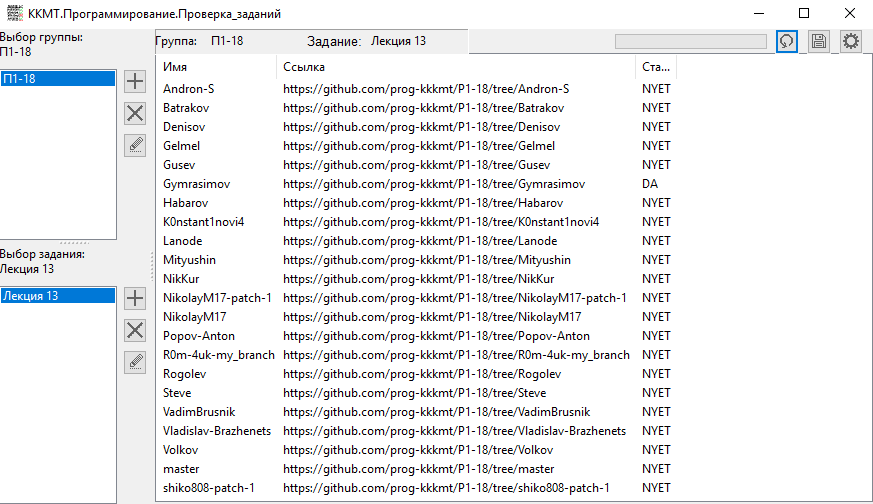


Рисунок 11.Демонстрация вывода с найденным заданием

## Отладка и тестирование программного средства

Во время разработки программы мы столкнулись с багом при котором не было проверки репозитория на существование. Поэтому при запуске обновления списка выполнения задания программа выдавала ошибку и не могла продолжить работу дальше.

После того как мы добавили проверку на наличие репозитория в диалог добавления новой группы, баг исчез

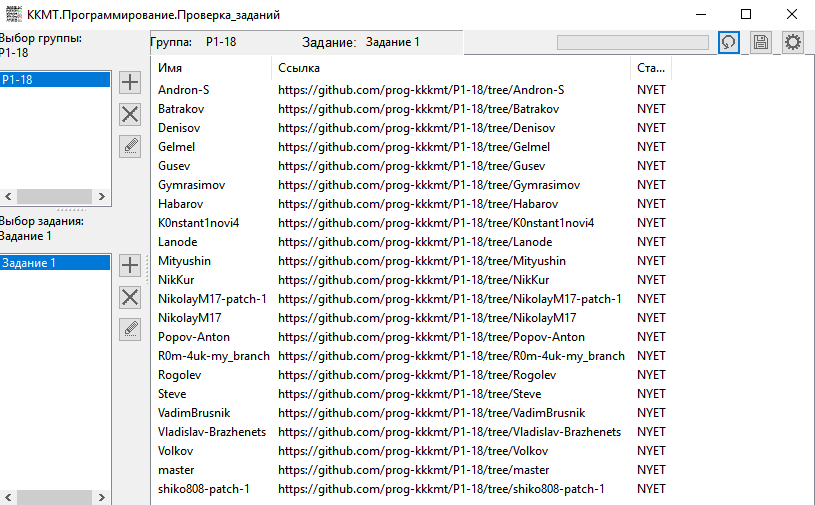


Рисунок 12.Исправление бага

## Комплект технологической документации

Разработал комплект технологической документации. Документация будет распределена в приложение 1.

## Заключение

В ходе производственной практики был получен опыт в среде коллективной работы и разработке настольного приложения. Оптимизировали программу, устранив ошибки.

# Дневник производственной практики на предприятии «ККМТ»

# Дневник производственной практики на предприятии

|  |  |  |
| --- | --- | --- |
| Дата | Содержание работы | Подпись руководителя практики от предприятия |
|  | Изучение и анализ предметной области |  |
|  | Получение задания на конкретную разработку |  |
|  | Изучение документооборота, подлежащего автоматизации |  |
|  | Построение IDEF-модели |  |
|  | Составить техническое задание на разработку программного средства. |  |
|  | Разработать модули программного средства в соответствии с ТЗ |  |
|  | Разработать пользовательский интерфейс программного средства. |  |
|  | Провести отладку и тестирование программного средства. |  |
|  | Разработать комплект технологической документации. |  |

Руководитель: \_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_

# **Приложение 1. Руководство оператора**

**Руководство по использованию программы KkmtTesting**

Для того, чтобы запустить программу, вам нужно разархивировать папку.

Чтобы запустить программу, вам нужно нажать на KkmtTesting с типом Lazarus Project Main Source.

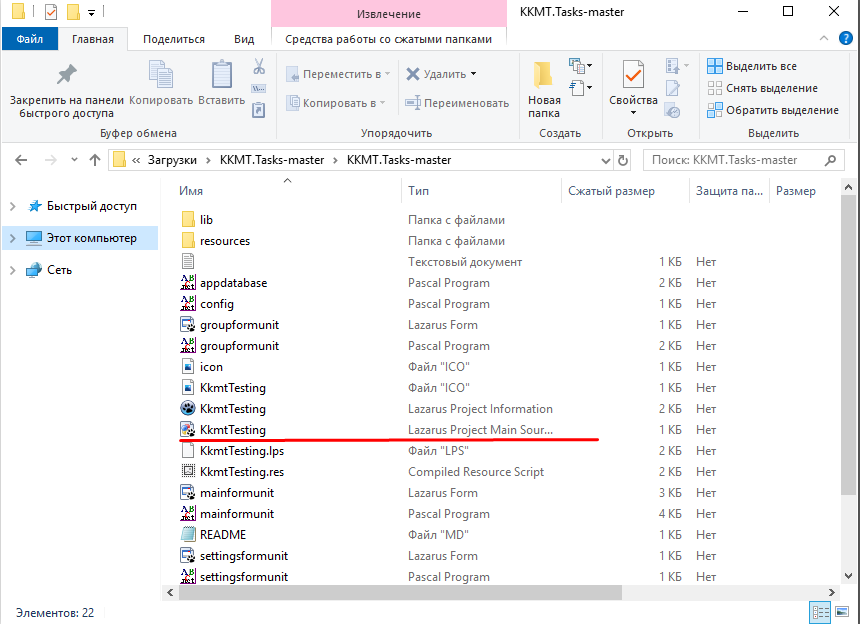


Рисунок 13.Папка с проектом

Прежде чем начать компиляцию, вам нужно выбрать режим сборки, для этого вам нужно нажать на стрелочку вниз, справа от шестеренки.

И вам нужно выбрать режим в зависимости от вашей операционной системы.

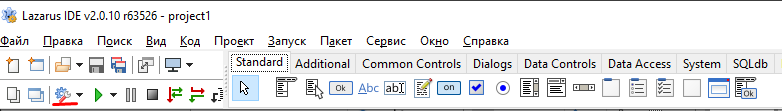


Рисунок 14.Выбор операционной системы(1)

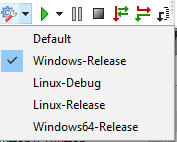


Рисунок 15.Выбор операционной системы(2)

Для того, чтобы скомпилировать программу, вам нужно войти в кладку Запуск и нажать Компилировать.

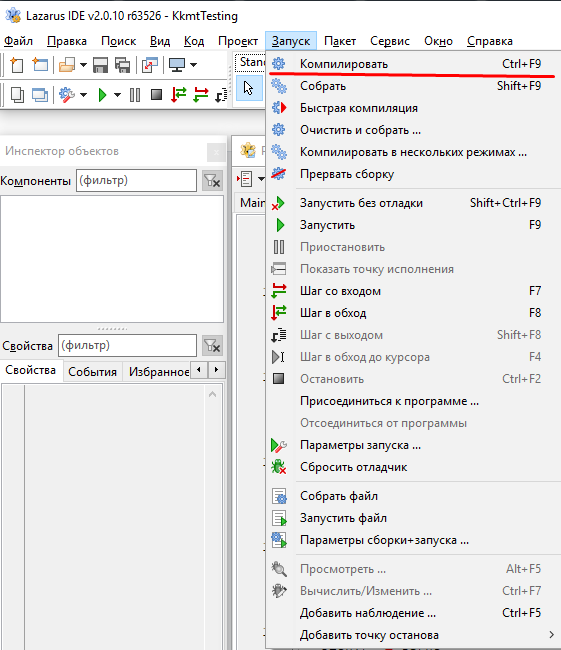


Рисунок 16.Компиляция программы

После выполнения компиляции, вам нужно зайти в KKMT.Tasks(текущая версия)\build\Зависимость от выбора сборки\KkmtTesting

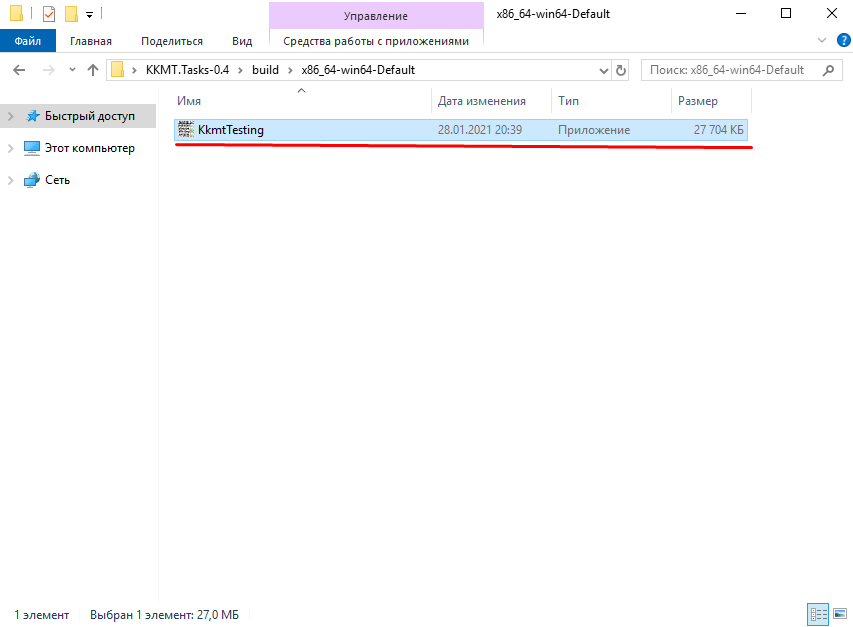


Рисунок 17.Запуск exe файла

Если у вас виндовс с разрядностью 32 бит или 64 бит, то вам нужно переместить дллку в папку с приложением(скрин приложения выше). Для этого вам нужно перейти по пути KKMT.Tasks(текущая версия)\lib\sqlite\х32(в зависимость от разрядности вашей ОС)\sqlite3.dll .Переносим эту длл в путь KKMT.Tasks(текущая версия)\build\Зависимость от выбора сборки.

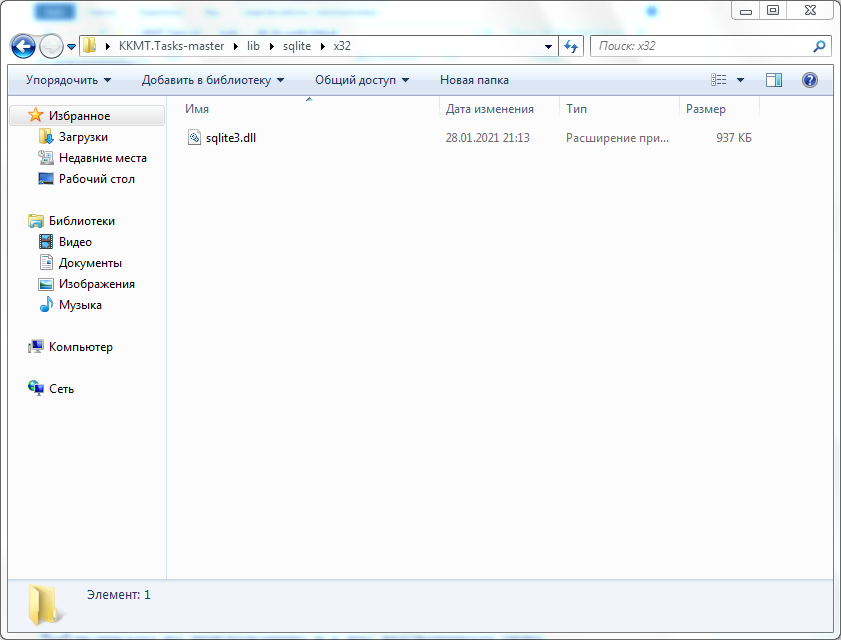


Рисунок 18.Перенос dll файла(1)

Нужно, чтобы папка выглядела так.

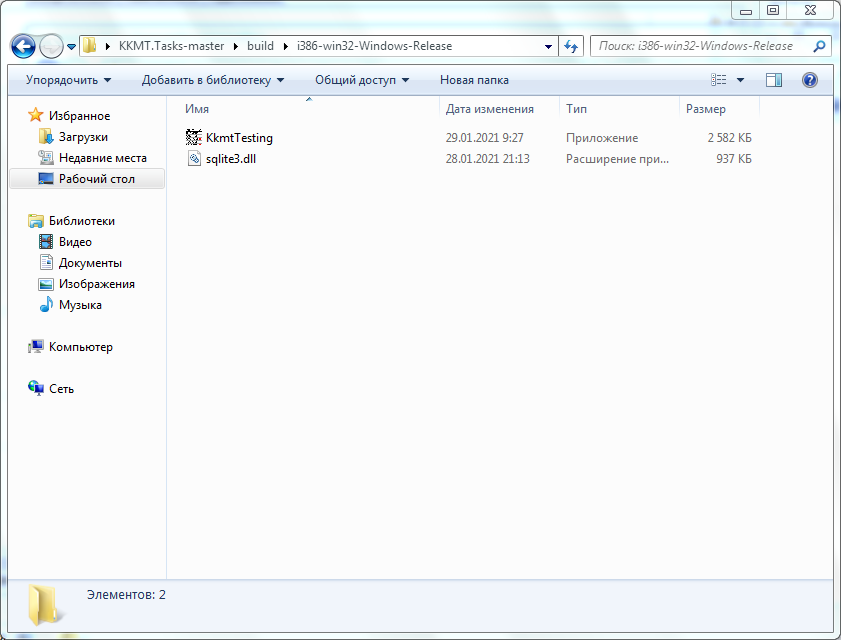


Рисунок 19.Перенос dll файла(2)

Даблкликаем по приложению и у нас высвечивает окно.

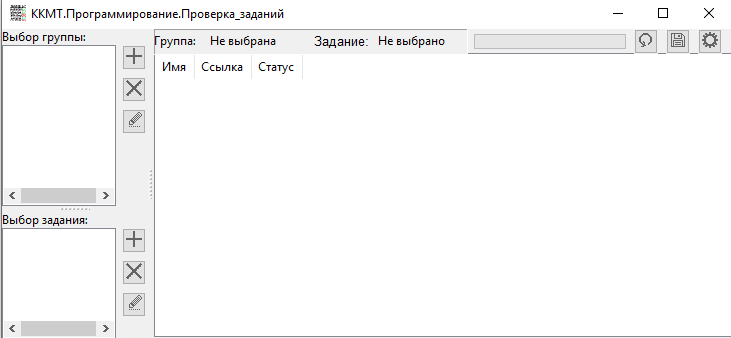


Рисунок 20.Открытие приложения

Но для начала вам нужно зайти на гитхаб для получения токена.

Для самого начала вам нужно получить токен GitHub API, для этого вам нужно авторизоваться на GitHub и после нажать на вашу аватарку в правом углу сверху и после нажимаете Settings.



Рисунок 21.Генерация токена(1)

После того, как вы перешли в свой профиль, нажимаете Developer settings.

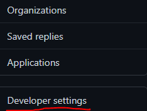


Рисунок 22.Генерация токена(2)

После перехода в Developer settings, вам нужно нажать Personal access token

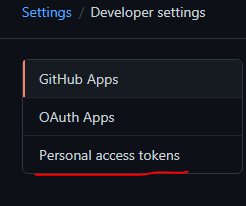


Рисунок 23.Генерация токена(3)

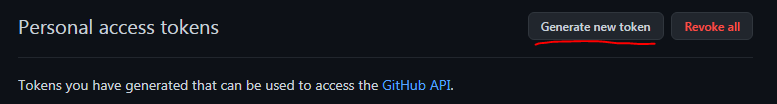


Рисунок 24.Генерация токена(4)

Чтобы сгенерировать токен, нажмите Generate new token.

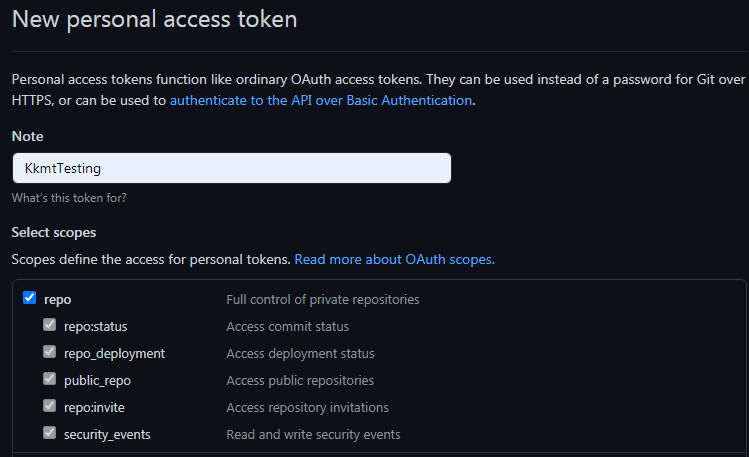


Рисунок 25.Генерация токена(5)

После нажатия на кнопку, вам нужно будет ввести название токена(можно любое) и поставить галочку в кладке repo и после нажать ниже Generate token( больше нигде не надо ставить галочки).

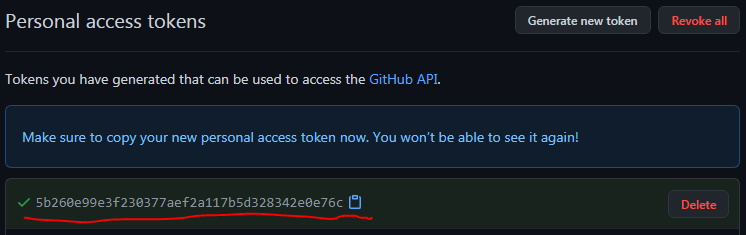


Рисунок 26.Генерация токена(6)

После у вас появляется токен и вы копируете его(можно нажать кнопку справа, чтобы скопировать одним кликом.

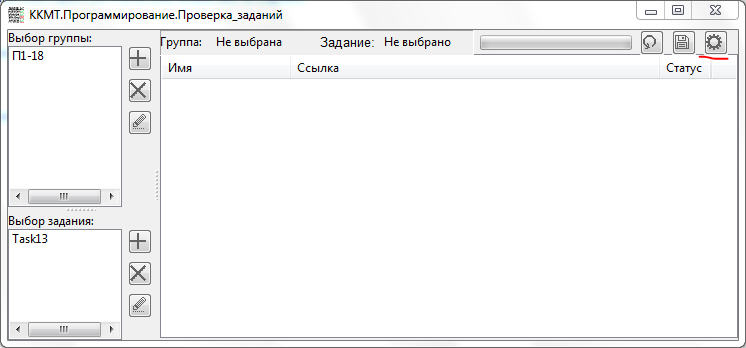


Рисунок 27.Ввод токена и имени(1)

Готово, у вас теперь есть токен и вам теперь нужно нажать шестеренку в правом верхнем углу.

Вам высветилось специальное окошко для ввода токена и имени на гитхаб.

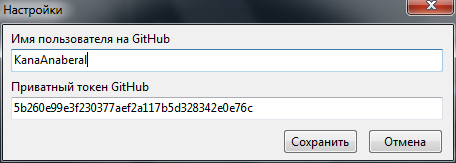


Рисунок 28. Ввод токена и имени(2)

Это все делалось для того, чтобы вы не перезагружали программу после реквестов, т.к. без токена можно получить 60 реквестов в час, а с токеном 5000 в час.

После того, как вы ввели токен и имя, вам нужно выбрать группу и задание, для этого вы нажимаете плюсик.

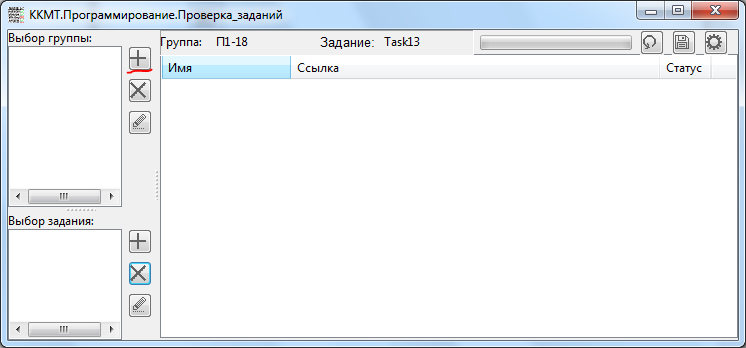


Рисунок 29.Ввод имя группы и репозитория(1)

Затем вводите имя группы(имя группы можно выбрать любое, но главное, чтобы не было 2 одинаковых групп) и репозиторию( путь к группе).

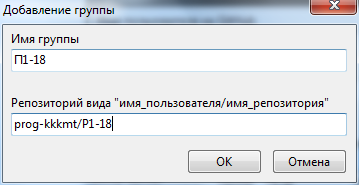


Рисунок 30.Ввод имя группы и репозитория(1)

У нас теперь есть группа и теперь нам нужно выбрать задание, чтобы добавить задание, нажмите на плюсик.

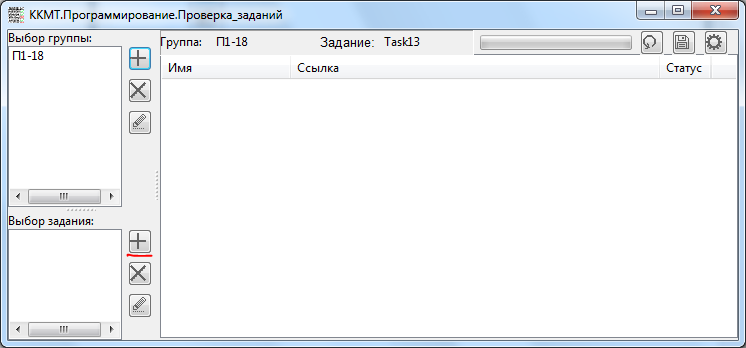


Рисунок 31.Ввод задания(1)

Вам высветилось окошко с добавлением задания, чтобы добавить задание нажмите плюсик.

\

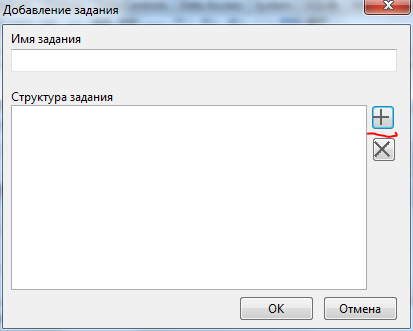


Рисунок 32.Ввод задания(2)

После вам высвечивается окно, где вам нужно ввести имя файла для проверки.

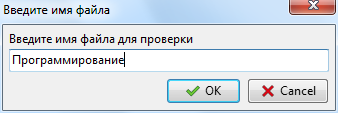


Рисунок 33.Ввод задания(3)

У вас появилось окошко, чтобы добавить дополнительное задание. Чтобы выбрать следующее ответвление к файлу (если конечно вы нуждаетесь в этом), то можно задать следующий путь. Для этого вам нужно нажать на то название, которые вы написали и нажать снова плюсик.

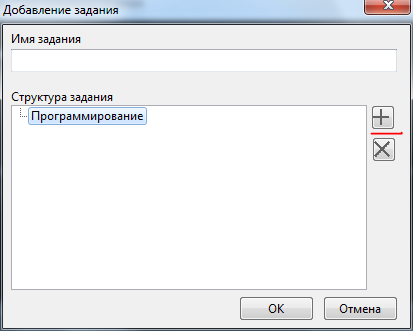


Рисунок 34.Ввод задания(4)

После вы вводите снова название файла и пишете имя задания (можно любое).

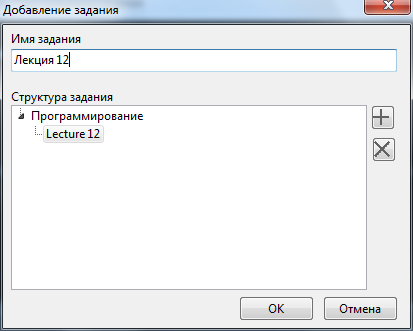


Рисунок 35.Ввод задания(5)

И так, чтобы проверите задание у группы, вам нужно даблкликнуть по группе (там где у вас написано “Выбор группы:”) и нажать задание (там где у вас написано “Выбор задание:”).

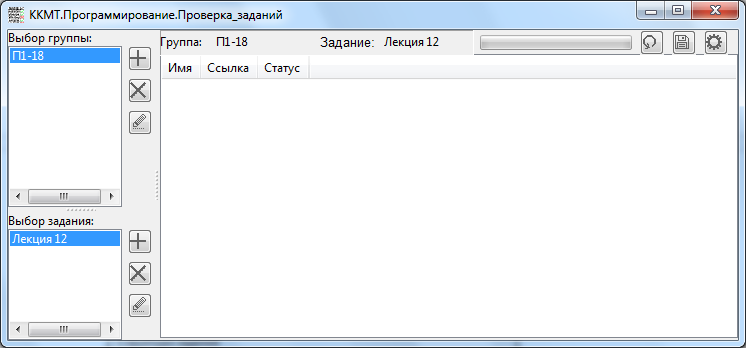


Рисунок 36.Введено задание и группа

Мы выбрали группу и задание, поэтому чтобы получить результат, вам нужно нажать на закругленную стрелочку.

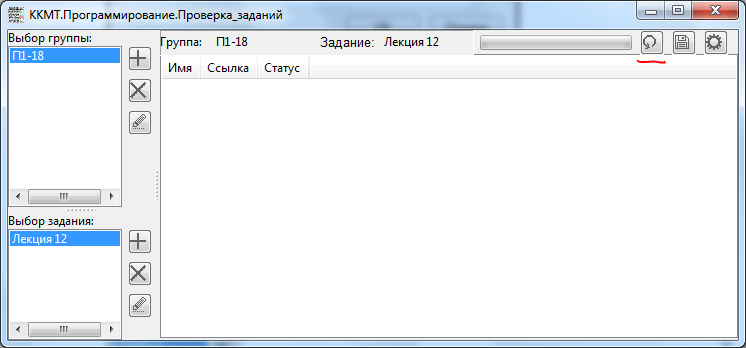


Рисунок 37.Процесс проверки на наличие задания

В итоге вы получаете результат, и мы видим, что только у Герасимова есть двенадцатая лекция.

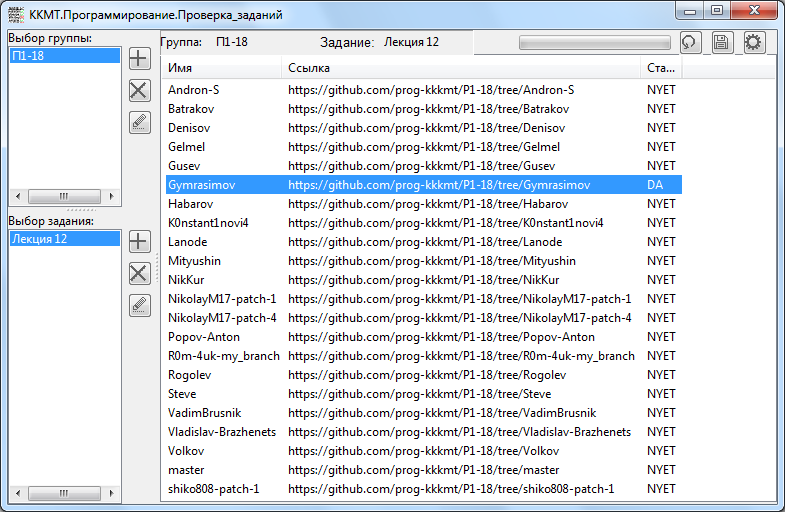


Рисунок 38.Проверка на наличие задания

Статус выдает NYET, когда у студента отсутствует задание, а когда у студента есть это задание, выдает DA.

Также можно перейти на ветку в репозиторий, кликнув на данные студента.

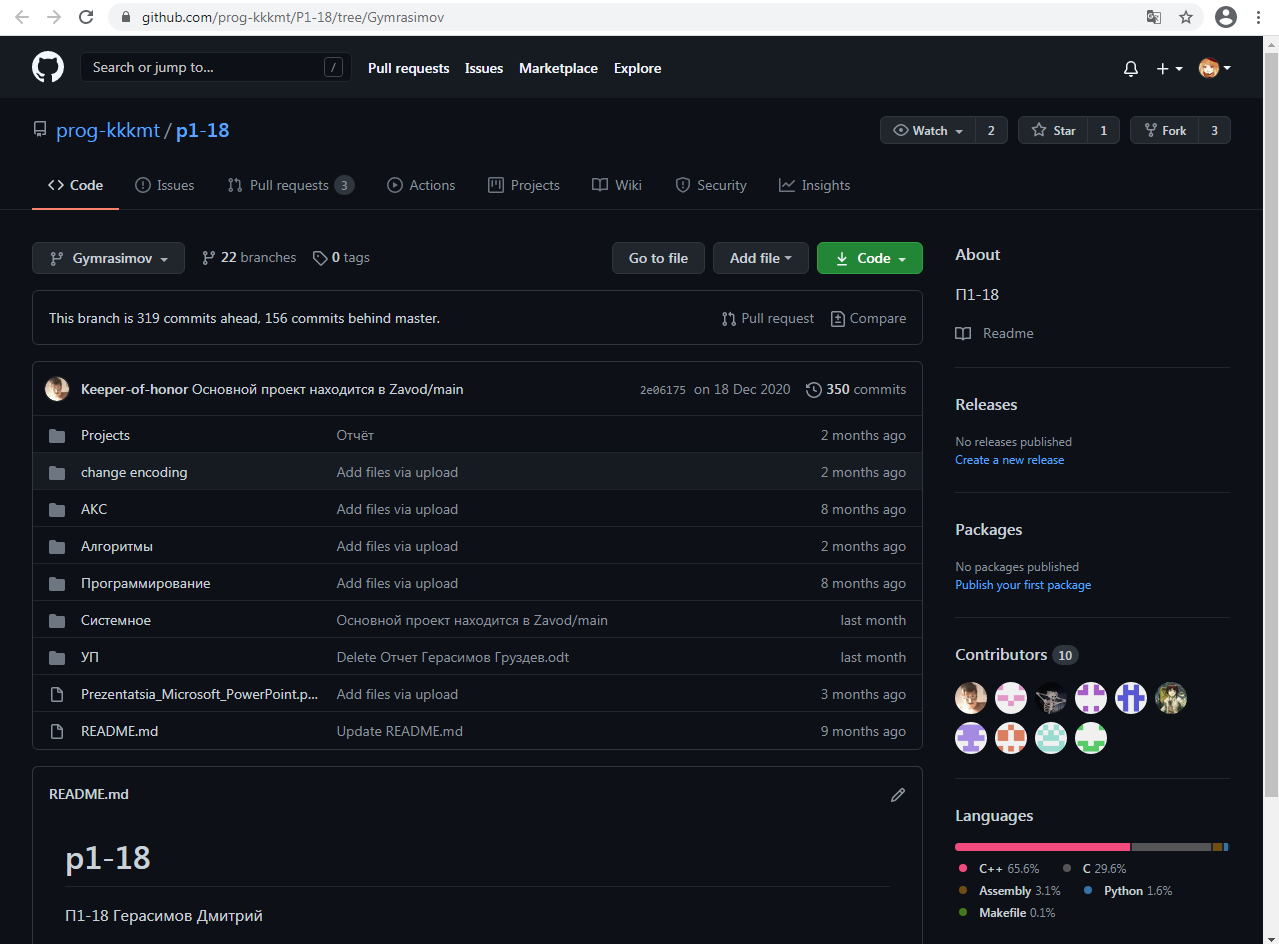


Рисунок 39.Проверка GitHub(1)

Можем также проверить, есть ли правда у студента этот путь.

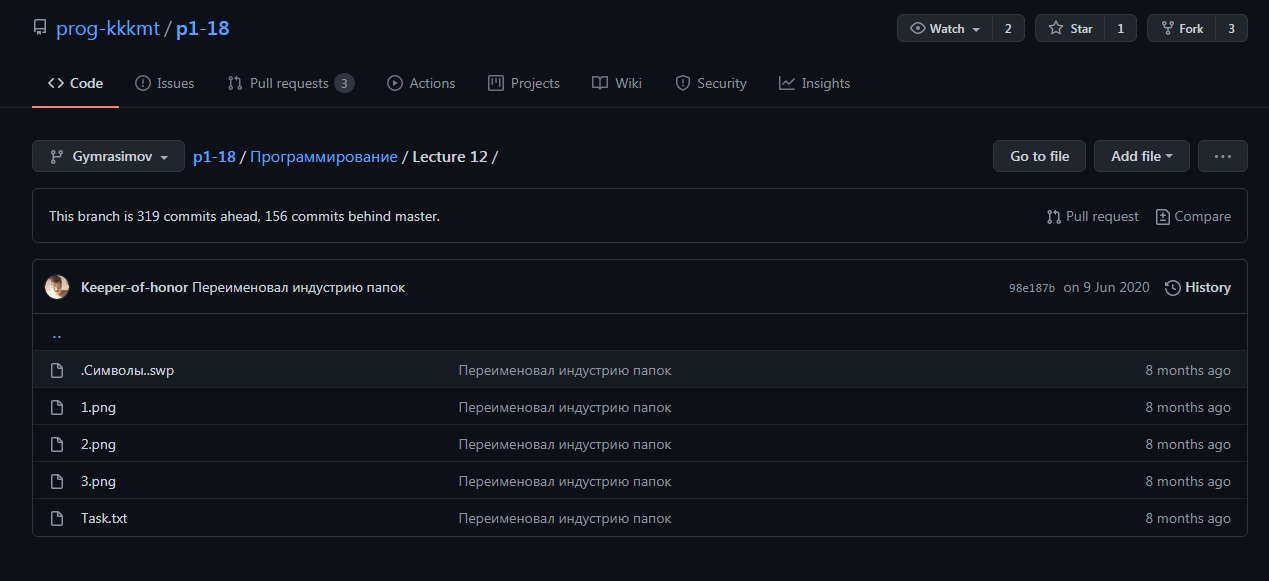
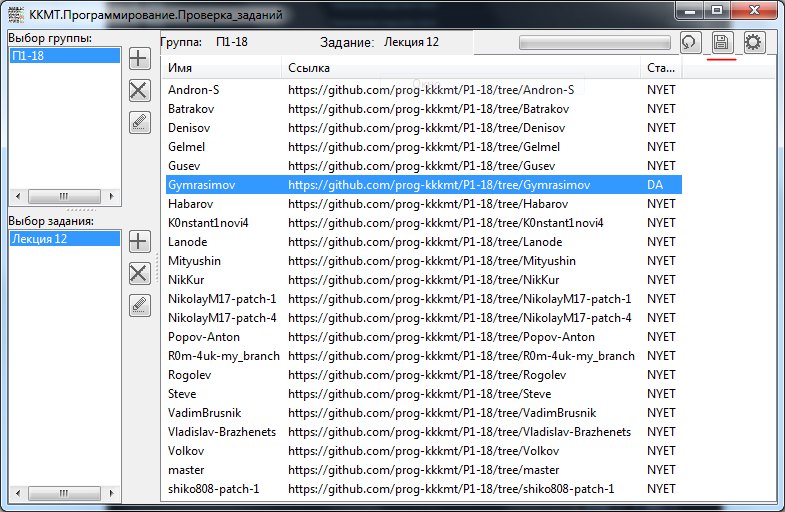


Рисунок 40.Проверка GitHub(2)

Можно сохранить результат, для этого вам нужно нажать на значок дискеты в правом верхнем углу.

Рисунок 41.Сохранение результата



Сохраненный результат можно найти по пути: KKMT.Tasks(текущая версия)\build\Зависимость от выбора сборки\reports

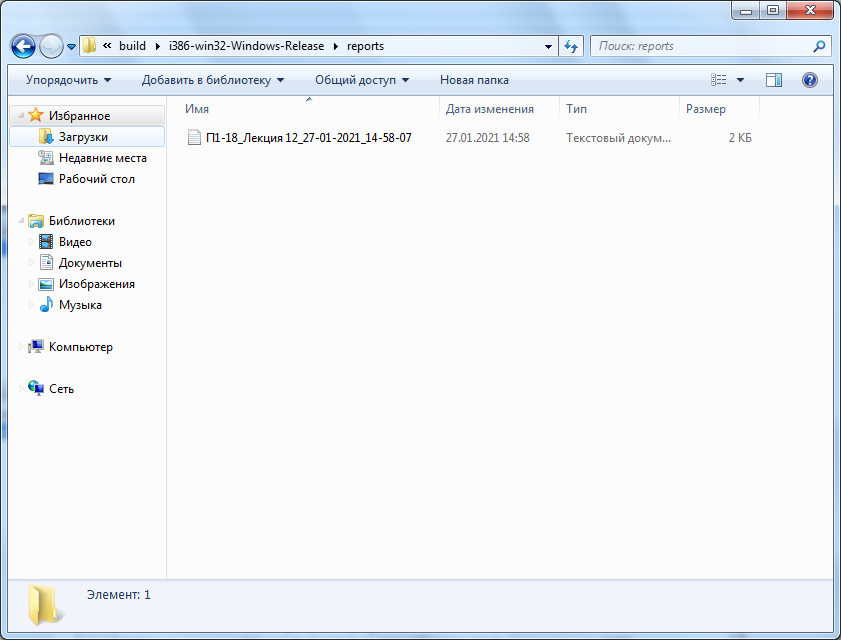


Рисунок 42.Текстовый файл с результатами

Как можем увидеть, наш результат сохранился в текстовый файл.

Это были все важные аспекты, также есть удаление групп и заданий из списка, вам для этого нужно нажать на крестик.

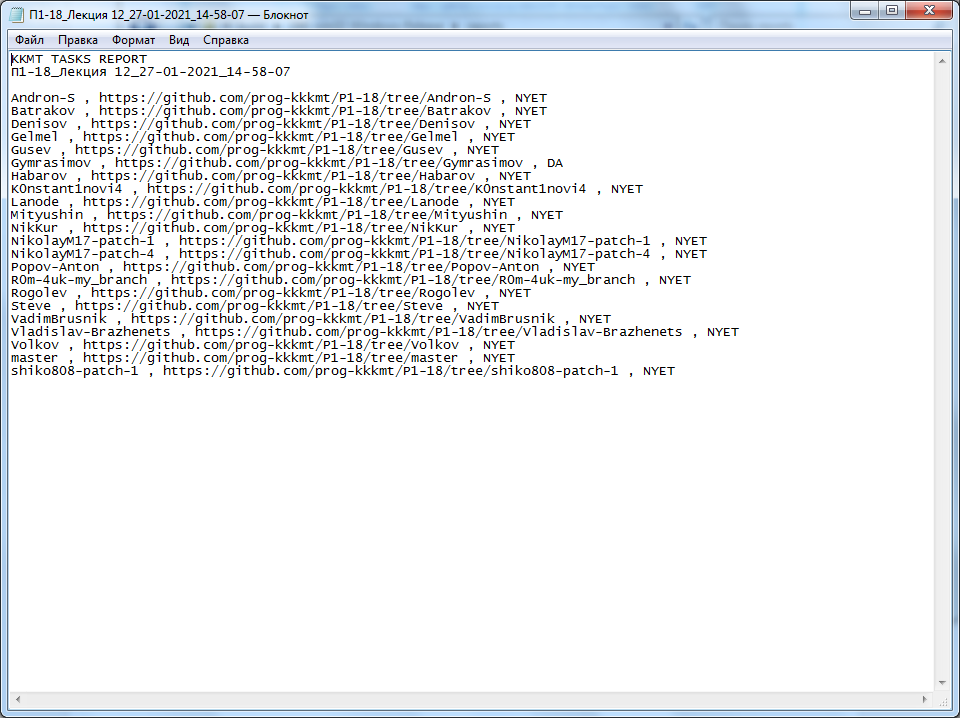


Рисунок 43.Текстовый файл с результатами

# **Приложение 2. Руководство программиста**

**Наименование программы**

Наименование программы – “KKMT.Tasks-0.1”.

**Краткая характеристика области применения**

Программа “KKMT.Tasks” предназначена для личного пользования любым пользователем. Программа позволяет создавать, открывать, сохранять и редактировать базу данных учеников, получать их статус выполненных работ и сохранять отчет.

**Основания для разработки**

Программа разрабатывалась, чтобы закрыть производственную практику.

**Функциональное назначение**

Открывать базу данных учеников, получать их статус выполненных работ и сохранять отчет. И добавлять, редактированить и удалять группы, добавлять задания, редактировать и удалять задания.

**Эксплуатационное назначение**

Облегчение проверки заданий, преподавателями.

**Требования к составу выполняемых функции**

Программа должна содержать 1 окно. Главный экран, который содержит кнопки с верху для работы с базой данных. Кнопки: создания, открытия, сохранения базы данных сохранения отчета и получения статуса.

**Требования к организации входных и выходных данных**

Входные данные представляют собой название группы и название и описание задач для выполнения.  
Выходные данные представляют из себя текстовый файл с отчетом о выполнении.

**Требования к временным характеристикам**

Все операции в программе не должны блокировать графический интерфейс более чем на 1337мс.

**Самовосстанавливаемость программы**

Данная функция предусмотрена в программе.

**Режим работы программы**

Особых временных рамок не выявлено.

**Требования к составу**

Программа требуется преподавателям, поэтому никаких определённых требований не присутствует.

# **Приложение 3. Листинг кода**

**Листинг 1. MainFormUnit.pas**

unit MainFormUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, CheckLst,

ExtCtrls, Grids, ValEdit, ComCtrls, SQLite3, jsonparser, fpjson, fphttpclient,

Types, LCLIntf, Buttons,

Utils, AppDatabase,

GroupFormUnit, TaskFormUnit, SettingsFormUnit;

type

{ TMainForm }

// Класс главной формы

TMainForm = class(TForm)

// Кнопка добавления группы

AddGroupButton: TBitBtn;

// Кнопока добавления группы

AddTaskList: TBitBtn;

// Кнопка удаления группы

DeleteGroupButton: TBitBtn;

// Кнопка удаления таска

DeleteTaskButton: TBitBtn;

// Кнопка редактирования группы

EditGroupButton: TBitBtn;

// Группа редактирования таска

EditTaskButton: TBitBtn;

IconList: TImageList;

Panel3: TPanel;

// Кнопка сохранения репорта

SaveStatusReportButton: TBitBtn;

Panel1: TPanel;

Panel2: TPanel;

ProgressBar1: TProgressBar;

// Кнопка настроек

SettingsButton: TBitBtn;

Splitter1: TSplitter;

Splitter2: TSplitter;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

// Лист групп

GroupsList: TListBox;

StaticText1: TStaticText;

StaticText2: TStaticText;

// Лист тасков

TasksList: TListBox;

// Главный лист с статусом выполнения заданий

ListView1: TListView;

TasksPanel: TPanel;

GroupsPanel: TPanel;

// Кнопка обновления репорта

UpdateStatusButton: TBitBtn;

// Нажатие на кнопку добавления группы

procedure AddGroupButtonClick(Sender: TObject);

// Нажатие на кнопку добавления таска

procedure AddTaskListClick(Sender: TObject);

// Нажатие лист групп

procedure GroupsListClick(Sender: TObject);

// Двойное нажатие на лист с статусами заданий

procedure ListView1DblClick(Sender: TObject);

// Нажатие на кнопку сохранения репорта

procedure SaveStatusReportButtonClick(Sender: TObject);

// Нажатие на кнопку настроек

procedure SettingsButtonClick(Sender: TObject);

// Нажатие на лист тасков

procedure TasksListClick(Sender: TObject);

// Нажатие на кнопку обновления статусов заданий

procedure UpdateStatusButtonClick(Sender: TObject);

// Нажатие на кнопку удаления группы

procedure DeleteGroupButtonClick(Sender: TObject);

// Нажатие на кнопку удаления таска

procedure DeleteTaskButtonClick(Sender: TObject);

// Нажатие на кнопку редактирования группы

procedure EditGroupButtonClick(Sender: TObject);

// Нажатие на кнопку редактирования таска

procedure EditTaskButtonClick(Sender: TObject);

// Создание формы

procedure FormCreate(Sender: TObject);

// Функция обновления листа

procedure UpdateList(TableName: String; UpdatingList: TListBox);

//

procedure UpdateCurrentGroup;

//

procedure UpdateCurrentTask;

// Toggle блокировку формы

procedure ChangeFormBlock(AEnabled: Boolean);

private

// Айди выбранной группы

SelectedGroup: Integer;

// Айди выбранного таска

SelectedTask: Integer;

// Айди загруженной группы

LoadedGroup: Integer;

// Айди загруженного таска

LoadedTask: Integer;

public

// Обновляется ли сейчас статус задания

NowUpdating: Boolean;

end;

var

MainForm: TMainForm;

implementation

{$R \*.lfm}

type

{ TReportUpdateThread }

TReportUpdateThread = class(TThread)

private

// Нужна ли остановка обновления

FStop: Boolean;

// Количество веток для проверки

FMax: Integer;

// Уже проверено веток

FProgress: Integer;

// Айди выбранной группы

FSelectedGroup: Integer;

// Айди выбранного таска

FSelectedTask: Integer;

// Лист итемов для добавления в вывод

FListItemToAdd: TListItem;

// Изменить прогресс бар

procedure ChangeProgressBar;

// Обновить прогресс

procedure UpdateProgress;

// Проверка была выполнена успешно

procedure SuccessfulExecute;

protected  
 // Главная функция потока

procedure Execute; override;

// Вызовится в любом случае после завершения работы

procedure AfterExecute(Sender: TObject);

public

constructor Create(CreateSuspended: Boolean; SelectedGroup, SelectedTask: Integer);

// Запуск проверки

procedure Start;

// Остановка проверки

procedure StopUpdating;

end;

var

// Поток проверки заданий

Thread: TReportUpdateThread;

{ TMainForm }

procedure TMainForm.FormCreate(Sender: TObject);

begin

UpdateList('groups', GroupsList);

UpdateList('tasks', TasksList);

SelectedGroup := -1;

SelectedTask := -1;

LoadedGroup := -1;

LoadedTask := -1;

end;

procedure TMainForm.UpdateStatusButtonClick(Sender: TObject);

begin

if not NowUpdating then

begin

if (SelectedGroup = -1) and (SelectedTask = -1) then exit;

Label2.Caption := StaticText2.Caption;

Label4.Caption := StaticText1.Caption;

Thread := TReportUpdateThread.Create(True, SelectedGroup, SelectedTask);

Thread.Start;

end

else

begin

Thread.StopUpdating;

end;

end;

procedure TMainForm.SaveStatusReportButtonClick(Sender: TObject);

var

AppDB: TAppDatabase;

ResultTable: TTable;

GroupName, TaskName: String;

ReportContents: TStringList;

FileName: String;

ListItem: TListItem;

begin

if (LoadedGroup = -1) or (LoadedTask = -1) then

exit;

AppDB := TAppDatabase.Create;

ResultTable := AppDB.ExecOut('SELECT name FROM groups WHERE id=:id', [par('id', LoadedGroup)]);

GroupName := ResultTable[0].KeyData['name'];

ResultTable := AppDB.ExecOut('SELECT name FROM tasks WHERE id=:id', [par('id', LoadedTask)]);

TaskName := ResultTable[0].KeyData['name'];

ResultTable.Free;

AppDB.Free;

ReportContents := TStringList.Create;

FileName := Format('%s\_%s\_%s',

[GroupName, TaskName, FormatDateTime('DD-MM-YYYY\_hh-nn-ss', Now)]);

ReportContents.Add('KKMT TASKS REPORT');

ReportContents.Add(Format('%s', [FileName]));

ReportContents.Add('');

for ListItem in ListView1.Items do

ReportContents.Add(Format('%s , %s , %s',

[ListItem.Caption, ListItem.SubItems[0], ListItem.SubItems[1]]));

ForceDirectories(Application.Location+'/reports');

ReportContents.SaveToFile(Application.Location+'/reports/'+FileName+'.txt');

end;

procedure TMainForm.SettingsButtonClick(Sender: TObject);

var

ModalForm: TSettingsForm;

begin

ModalForm := TSettingsForm.Create(nil);

ModalForm.ShowModal;

ModalForm.Free;

end;

procedure TMainForm.ListView1DblClick(Sender: TObject);

begin

if ListView1.ItemIndex < 0 then

exit;

OpenURL(ListView1.Items[ListView1.ItemIndex].SubItems.Strings[0]);

end;

{ === GroupsPanel === }

procedure TMainForm.GroupsListClick(Sender: TObject);

begin

if GroupsList.ItemIndex < 0 then

exit;

SelectedGroup := TIntObj(GroupsList.Items.Objects[GroupsList.ItemIndex]).I;

StaticText2.Caption := GroupsList.GetSelectedText;

end;

procedure TMainForm.AddGroupButtonClick(Sender: TObject);

var

ModalForm: TGroupForm;

begin

ModalForm := TGroupForm.Create(MainForm);

ModalForm.ShowModal;

ModalForm.Free;

//OpenDialog(TGroupForm);

UpdateList('groups', GroupsList);

end;

procedure TMainForm.DeleteGroupButtonClick(Sender: TObject);

var

AppDB: TAppDatabase;

begin

if GroupsList.ItemIndex >= 0 then

begin

AppDB := TAppDatabase.Create;

AppDB.Exec('DELETE FROM groups WHERE id=:id',

[par('id', TIntObj(GroupsList.Items.Objects[GroupsList.ItemIndex]).I)]);

UpdateCurrentGroup;

UpdateList('groups', GroupsList);

end;

end;

procedure TMainForm.EditGroupButtonClick(Sender: TObject);

var

ModalForm: TGroupForm;

begin

if GroupsList.ItemIndex >= 0 then

begin

ModalForm := TGroupForm.Create(MainForm, TIntObj(GroupsList.Items.Objects[GroupsList.ItemIndex]).I);

ModalForm.ShowModal;

if ModalForm.ModalResult = mrOK then

begin

UpdateCurrentGroup;

UpdateList('groups', GroupsList);

end;

ModalForm.Free;

end;

end;

{ === TasksPanel === }

procedure TMainForm.TasksListClick(Sender: TObject);

begin

if TasksList.ItemIndex < 0 then

exit;

SelectedTask := TIntObj(TasksList.Items.Objects[TasksList.ItemIndex]).I;

StaticText1.Caption := TasksList.GetSelectedText;

end;

procedure TMainForm.AddTaskListClick(Sender: TObject);

var

ModalForm: TTaskForm;

begin

ModalForm := TTaskForm.Create(MainForm);

ModalForm.ShowModal;

ModalForm.Free;

UpdateList('tasks', TasksList);

end;

procedure TMainForm.DeleteTaskButtonClick(Sender: TObject);

var

AppDB: TAppDatabase;

begin

if TasksList.ItemIndex >= 0 then

begin

AppDB := TAppDatabase.Create;

AppDB.Exec('DELETE FROM tasks WHERE id=:id',

[par('id', TIntObj(TasksList.Items.Objects[TasksList.ItemIndex]).I)]);

UpdateCurrentTask;

UpdateList('tasks', TasksList);

end;

end;

procedure TMainForm.EditTaskButtonClick(Sender: TObject);

var

ModalForm: TTaskForm;

begin

if TasksList.ItemIndex >= 0 then

begin

ModalForm := TTaskForm.Create(MainForm, TIntObj(TasksList.Items.Objects[TasksList.ItemIndex]).I);

ModalForm.ShowModal;

if ModalForm.ModalResult = mrOk then

begin

UpdateCurrentTask;

UpdateList('tasks', TasksList);

end;

ModalForm.Free;

end;

end;

procedure TMainForm.UpdateList(TableName: String; UpdatingList: TListBox);

var

AppDB: TAppDatabase;

Table: TTable;

Row: TRow;

begin

AppDB := TAppDatabase.Create;

Table := AppDB.ExecOut('SELECT \* FROM '+TableName+';');

UpdatingList.Clear;

for Row in Table do

begin

UpdatingList.Items.AddObject(Row.KeyData['name'], TIntObj.Create(Row.KeyData['id']));

end;

end;

procedure TMainForm.UpdateCurrentGroup;

var

AppDB: TAppDatabase;

Table: TTable;

begin

if SelectedGroup <> -1 then

begin

AppDB := TAppDatabase.Create;

Table := AppDB.ExecOut('SELECT \* FROM groups WHERE id=:id;', [par('id', SelectedGroup)]);

if Table.Count = 0 then

begin

SelectedGroup := -1;

Label2.Caption := '';

end

else

begin

Label2.Caption := Table.Items[0].KeyData['name'];

end;

end;

if LoadedGroup <> -1 then

begin

Table := AppDB.ExecOut('SELECT \* FROM groups WHERE id=:id;', [par('id', LoadedGroup)]);

if Table.Count = 0 then

begin

LoadedGroup := -1;

ListView1.Clear;

end;

end;

Table.Free;

end;

procedure TMainForm.UpdateCurrentTask;

var

AppDB: TAppDatabase;

Table: TTable;

begin

if SelectedTask <> -1 then

begin

AppDB := TAppDatabase.Create;

Table := AppDB.ExecOut('SELECT \* FROM tasks WHERE id=:id;', [par('id', SelectedTask)]);

if Table.Count = 0 then

begin

SelectedTask := -1;

Label4.Caption := '';

end

else

begin

Label4.Caption := Table.Items[0].KeyData['name'];

end;

end;

if LoadedTask <> -1 then

begin

Table := AppDB.ExecOut('SELECT \* FROM tasks WHERE id=:id;', [par('id', LoadedTask)]);

if Table.Count = 0 then

begin

LoadedTask := -1;

ListView1.Clear;

end;

end;

Table.Free;

end;

procedure TMainForm.ChangeFormBlock(AEnabled: Boolean);

begin

GroupsList.Enabled := AEnabled;

AddGroupButton.Enabled := AEnabled;

DeleteGroupButton.Enabled := AEnabled;

EditGroupButton.Enabled := AEnabled;

TasksList.Enabled := AEnabled;

AddTaskList.Enabled := AEnabled;

DeleteTaskButton.Enabled := AEnabled;

EditTaskButton.Enabled := AEnabled;

SaveStatusReportButton.Enabled := AEnabled;

SettingsButton.Enabled := AEnabled;

end;

{ TReportUpdateThread }

constructor TReportUpdateThread.Create(CreateSuspended: Boolean; SelectedGroup, SelectedTask: Integer);

begin

inherited Create(CreateSuspended);

FreeOnTerminate := True;

FStop := False;

FProgress := 0;

FSelectedGroup := SelectedGroup;

FSelectedTask := SelectedTask;

FListItemToAdd := nil;

OnTerminate := @AfterExecute;

MainForm.ProgressBar1.Style := TProgressBarStyle.pbstMarquee;

MainForm.ProgressBar1.Position := 0;

MainForm.ListView1.Clear;

end;

procedure TReportUpdateThread.Start;

begin

MainForm.NowUpdating := True;

MainForm.UpdateStatusButton.ImageIndex := 1;

MainForm.ChangeFormBlock(False);

inherited Start;

end;

procedure TReportUpdateThread.StopUpdating;

begin

FStop := True;

end;

procedure TReportUpdateThread.AfterExecute(Sender: TObject);

begin

MainForm.ProgressBar1.Style := TProgressBarStyle.pbstNormal;

MainForm.ProgressBar1.Position := 0;

MainForm.NowUpdating := False;

MainForm.UpdateStatusButton.ImageIndex := 3;

MainForm.ChangeFormBlock(True);

end;

procedure TReportUpdateThread.ChangeProgressBar;

// this method is executed by the mainthread and can therefore access all GUI elements.

begin

MainForm.ProgressBar1.Style := TProgressBarStyle.pbstNormal;

MainForm.ProgressBar1.Max := FMax;

end;

procedure TReportUpdateThread.UpdateProgress;

// this method is executed by the mainthread and can therefore access all GUI elements.

begin

MainForm.ProgressBar1.StepIt;

MainForm.ListView1.Items.AddItem(FListItemToAdd);

end;

procedure TReportUpdateThread.SuccessfulExecute;

begin

MainForm.LoadedGroup := FSelectedGroup;

MainForm.LoadedTask := FSelectedTask;

end;

procedure TReportUpdateThread.Execute;

var

HTTP: TFPHttpClient;

AppDB: TAppDatabase;

Table: TTable;

GitHubId: String;

HttpResult: String;

BranchesJSON: TJSONArray;

Enum, Enum2: TJSONEnum;

Files: TStringList;

Counter: Integer;

begin

HTTP := CreateGitHubHTTP;

AppDB := TAppDatabase.Create;

Table := AppDB.ExecOut('SELECT github\_id FROM groups WHERE id=:id;', [par('id', FSelectedGroup)]);

if Table.Count <= 0 then

exit;

GitHubId := Table[0].KeyData['github\_id'];

Table := AppDB.ExecOut('SELECT \* FROM tasks WHERE id=:id;', [par('id', FSelectedTask)]);

if Table.Count <= 0 then

exit;

Files := TStringList.Create;

for Enum in TJSONArray(GetJSON(Table[0].KeyData['files'])) do

Files.Add(Enum.Value.AsString);

try

HttpResult := HTTP.Get(Format('https://api.github.com/repos/%s/branches', [GitHubId] ));

except

MessageDlg('Ошибка получения данных о ветках репозитория. Возможно репозиторий не существует.', mtError,

[mbOK], 0);

exit;

end;

BranchesJSON := TJSONArray(GetJSON(HttpResult));

FMax := BranchesJSON.Count;

Synchronize(@ChangeProgressBar);

for Enum in BranchesJSON do

begin

if FStop then break;

FListItemToAdd := TListItem.Create(MainForm.ListView1.Items);

FListItemToAdd.Caption := TJSONObject(enum.Value)['name'].AsString;

FListItemToAdd.SubItems.Add(Format('https://github.com/%s/tree/%s',

[GitHubId, TJSONObject(enum.Value)['name'].AsString]));

HttpResult := HTTP.Get(Format('https://api.github.com/repos/%s/git/trees/%s?recursive=1',

[GitHubId, TJSONObject(enum.Value).GetPath('commit.sha').AsString]));

Counter := Files.Count;

for Enum2 in TJSONObject(GetJSON(HttpResult)).Arrays['tree'] do

begin

if Files.IndexOf(TJSONObject(Enum2.Value).Strings['path']) <> -1 then

Dec(Counter);

end;

if Counter = 0 then

// something good

FListItemToAdd.SubItems.Add('DA')

else

// somethinf bad

FListItemToAdd.SubItems.Add('NYET');

Synchronize(@UpdateProgress);

end;

BranchesJSON.Free;

AppDB.Free;

HTTP.Free;

Table.Free;

Files.Free;

Synchronize(@SuccessfulExecute);

end;

end.

**Листинг 2. GroupFormUnit.pas**

unit GroupFormUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, ExtCtrls,

fphttpclient,

AppDatabase, Utils;

type

{ TGroupForm }

TGroupForm = class(TForm)

// OK кнопка

Button1: TButton;

// Кнопка отмены

Button2: TButton;

// Поле имени группы

LabeledEdit1: TLabeledEdit;

// Поле айди репозитория группы

LabeledEdit2: TLabeledEdit;

// Нажатие на ОК кнопку

procedure Button1Click(Sender: TObject);

// Нажатие на кнопку отмены

procedure Button2Click(Sender: TObject);

private

// Айди группы для режима реадктирования

EditId: Integer;

public

constructor Create(TheOwner: TComponent);

constructor Create(TheOwner: TComponent; GroupForEditing: Integer); overload;

// Изменить дизайн формы для редактирования группы

procedure MakeForEdit(GroupId: Integer);

end;

implementation

{$R \*.lfm}

{ TGroupForm }

procedure TGroupForm.Button1Click(Sender: TObject);

var

HTTP: TFPHttpClient;

HttpResult: String;

AppDB: TAppDatabase;

begin

HTTP := CreateGitHubHTTP;

AppDB := TAppDatabase.Create;

//ShowMessage(Format('https://api.github.com/repos/%s', [LabeledEdit2.Text]));

if String(LabeledEdit1.Text).IsEmpty or String(LabeledEdit1.Text).Contains(' ') then

begin

MessageDlg('Имя группы не должно быть пустым или иметь пробелы', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

try

HttpResult := HTTP.Get(Format('https://api.github.com/repos/%s', [LabeledEdit2.Text]));

except

if HTTP.ResponseStatusCode = 404 then

begin

MessageDlg('Репозиторий не найден', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end

else

begin

MessageDlg('Ошибка получения данных о репозитории. Возможно репозиторий не существует.', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

end;

try

if EditId = -1 then

AppDB.Exec('INSERT INTO groups (name, github\_id) VALUES (:name, :rep\_id);',

[par('name', LabeledEdit1.Text), par('rep\_id', LabeledEdit2.Text)])

else

AppDB.Exec('UPDATE groups SET name=:name, github\_id=:rep\_id WHERE id=:id;',

[par('name', LabeledEdit1.Text), par('rep\_id', LabeledEdit2.Text), par('id', EditId)] );

except

on E: Exception do

begin

MessageDlg('Ошибка при записи в базу данных.'+LineEnding+E.Message, mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

end;

end;

procedure TGroupForm.Button2Click(Sender: TObject);

begin

end;

constructor TGroupForm.Create(TheOwner: TComponent);

begin

inherited Create(TheOwner);

EditId := -1;

end;

constructor TGroupForm.Create(TheOwner: TComponent; GroupForEditing: Integer);

begin

inherited Create(TheOwner);

MakeForEdit(GroupForEditing);

end;

procedure TGroupForm.MakeForEdit(GroupId: Integer);

var

AppDB: TAppDatabase;

ResultTable: TTable;

begin

AppDB := TAppDatabase.Create;

ResultTable := AppDB.ExecOut('SELECT \* FROM groups WHERE id=:id;', [par('id', GroupId)]);

Self.LabeledEdit1.Text := ResultTable[0].KeyData['name'];

Self.LabeledEdit2.Text := ResultTable[0].KeyData['github\_id'];

Self.Caption := 'Изменение группы';

Self.EditId := GroupId;

end;

end.

**Листинг 3. TaskFormUnit.pas**

unit TaskFormUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, ExtCtrls,

CheckLst, ColorBox, ShellCtrls, ComCtrls, Buttons, AppDatabase, fpjson;

type

{ TTaskForm }

TTaskForm = class(TForm)

// ОК кнопка

Button1: TButton;

// Кнопка отмены

Button2: TButton;

// Кнопка добавления дочерней ноды имени файла

Button3: TBitBtn;

// Кнопка удаления ноды имени файла

Button4: TBitBtn;

Label1: TLabel;

// Имя таска

LabeledEdit1: TLabeledEdit;

Panel1: TPanel;

// Дерево файлов

TreeView1: TTreeView;

// Нажатие на кнопку сохранения

procedure Button1Click(Sender: TObject);

// Нажатие на кнопку отмены

procedure Button2Click(Sender: TObject);

// Нажатие на кнопку добавления ноды

procedure Button3Click(Sender: TObject);

// Нажатие на кнопку удаления ноды

procedure Button4Click(Sender: TObject);

// Выделение ноды

procedure TreeView1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

private

// Айди таска для режима редактирования

EditId: Integer;

// Создание массива путей для каждого файла

function TreeToPathsGenerator(Tree: TTreeNodes): TStringList;

// Создание дерева из массива путей

function TreeFromPathsGenerator(Paths: TStringList): TTreeNodes;

public

constructor Create(TheOwner: TComponent);

constructor Create(TheOwner: TComponent; TaskForEditing: Integer); overload;

// Изменить дизайн формы для режима редактирования

procedure MakeForEdit(TaskId: Integer);

end;

implementation

{$R \*.lfm}

{ TTaskForm }

procedure TTaskForm.Button3Click(Sender: TObject);

var

Filename: String;

begin

Filename := InputBox('Введите имя файла',

'Введите имя файла для проверки', '');

if not Filename.IsEmpty then

TreeView1.Items.AddChild(TreeView1.Selected, Filename);

end;

procedure TTaskForm.Button4Click(Sender: TObject);

begin

if TreeView1.Selected <> nil then

TreeView1.Items.Delete(TreeView1.Selected);

end;

procedure TTaskForm.TreeView1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var

i: Integer;

begin

if not Assigned(TreeView1.GetNodeAt(X, Y)) then

for i := 0 to TreeView1.Items.Count - 1 do

TreeView1.Items[i].Selected := false;

end;

procedure TTaskForm.Button1Click(Sender: TObject);

var

AppDB: TAppDatabase;

FilePath: String;

json: TJSONArray;

begin

if Length(LabeledEdit1.Text) <= 0 then

begin

MessageDlg('Введите имя задания', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

if TreeView1.Items.Count <= 0 then

begin

MessageDlg('Задайте структуру задания', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

json := TJSONArray.Create;

AppDB := TAppDatabase.Create;

for FilePath in TreeToPathsGenerator(TreeView1.Items) do

json.Add(FilePath.Substring(1));

//ShowMessage(json.AsJSON);

try

if EditId = -1 then

AppDB.Exec('INSERT INTO tasks (name, files) VALUES (:name, :files);', [par('name', LabeledEdit1.Text), par('files', json.AsJSON)] )

else

AppDB.Exec('UPDATE tasks SET name=:name, files=:files WHERE id=:id;',

[par('name', LabeledEdit1.Text), par('files', json.AsJSON), par('id', EditId)] );

except

on E: Exception do

begin

MessageDlg('Ошибка при записи в базу данных.'+LineEnding+E.Message, mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

end;

end;

procedure TTaskForm.Button2Click(Sender: TObject);

begin

end;

function TTaskForm.TreeToPathsGenerator(Tree: TTreeNodes): TStringList;

var Paths: TStringList;

procedure TreeWalker(Node: TTreeNode; PathString: String);

var

i: Integer;

begin

PathString += '/'+Node.Text;

if Node.HasChildren then

for i:=0 to Node.Count-1 do

TreeWalker(Node.Items[i], PathString)

else

Paths.Add(PathString);

end;

var

Node: TTreeNode;

begin

Paths := TStringList.Create;

for Node in Tree do

begin

if Node.Parent = nil then

TreeWalker(Node, '');

end;

Result := Paths;

end;

function TTaskForm.TreeFromPathsGenerator(Paths: TStringList): TTreeNodes;

var

Path: String;

PathParsed: TStringArray;

i, j: Integer;

Node, ChildNode: TTreeNode;

begin

Result := TreeView1.Items;

for Path in Paths do

begin

PathParsed := Path.Split(['/'], TStringSplitOptions.ExcludeEmpty);

Node := Result.FindNodeWithText(PathParsed[0]);

if (Node = nil) or (Node.Parent <> nil) then

Node := Result.AddChild(nil, PathParsed[0]);

for i := 1 to Length(PathParsed)-1 do

begin

ChildNode := Node.FindNode(PathParsed[i]);

if ChildNode = nil then

ChildNode := Result.AddChild(Node, PathParsed[i]);

Node := ChildNode;

end;

end;

end;

constructor TTaskForm.Create(TheOwner: TComponent);

begin

inherited Create(TheOwner);

EditId := -1;

end;

constructor TTaskForm.Create(TheOwner: TComponent; TaskForEditing: Integer);

begin

inherited Create(TheOwner);

MakeForEdit(TaskForEditing);

end;

procedure TTaskForm.MakeForEdit(TaskId: Integer);

var

AppDB: TAppDatabase;

ResultTable: TTable;

Paths: TStringList;

Enum: TJSONEnum;

begin

AppDB := TAppDatabase.Create;

ResultTable := AppDB.ExecOut('SELECT \* FROM tasks WHERE id=:id;', [par('id', TaskId)]);

Self.LabeledEdit1.Text := ResultTable[0].KeyData['name'];

Paths := TStringList.Create;

for Enum in TJSONArray(GetJSON(ResultTable[0].KeyData['files'])) do

Paths.Add(Enum.Value.AsString);

Self.TreeFromPathsGenerator(Paths);

Self.Caption := 'Изменение задания';

Self.EditId := TaskId;

end;

end.

**Листинг 4. AppDatabase.pas**

unit AppDatabase;

interface

uses

fpjson, sqldb, db, SQLite3Conn, sysutils, Generics.Collections, fgl, Variants, SQLite3;

type

// SQL параметр вы виде пары имя-значение

TParamPair = specialize TPair<String, Variant, String>;

// Динамический массив параметров

TParamsArray = Array of TParamPair;

// Строка таблицы

TRow = specialize TFPGMap<String, Variant>;

// Таблица

TTable = specialize TList<TRow>;

// Класс базы данных приложения

TAppDatabase = class (TInterfacedObject)

// Объект соединения с базой данных

Connection: TSQLite3Connection;

// Объект транзакции

Transaction: TSQLTransaction;

// Объект запроса

Query: TSQLQuery;

// Выполнить SQL запрос с параметрами без возврата результата

procedure Exec(AQuery: String; AParams: TParamsArray);

// Выполнить SQL запрос без параметров и без возврата результата

procedure Exec(AQuery: String);

// Выполнить SQL запрос с параметрами и с возвратом результата

function ExecOut(AQuery: String; AParams: TParamsArray): TTable;

// Выполнить SQL запрос без параметров с возвратом результата

function ExecOut(AQuery: String): TTable;

constructor Create;

destructor Free;

end;

// Функция для упрощенного создания объекта TParamPair

function par(AName: String; AValue: Variant): TParamPair;

implementation

// ===== TDatabase =====

procedure TAppDatabase.Exec(AQuery: String; AParams: TParamsArray);

var

Param: TParamPair;

begin

Transaction.StartTransaction;

Query.SQL.Text := AQuery;

if Length(AParams) > 0 then

for Param in AParams do

Query.ParamByName(Param.Key).Value := Param.Value;

Query.Prepare;

try

Query.ExecSQL;

finally

Transaction.Commit;

Query.Clear;

Transaction.EndTransaction;

end;

end;

procedure TAppDatabase.Exec(AQuery: String);

begin

Exec(AQuery, nil);

end;

function TAppDatabase.ExecOut(AQuery: String; AParams: TParamsArray): TTable;

var

Param: TParamPair;

Row: TRow;

Field: TField;

begin

Transaction.StartTransaction;

Query.SQL.Text := AQuery;

if Length(AParams) > 0 then

for Param in AParams do

Query.ParamByName(Param.Key).Value := Param.Value;

Query.Prepare;

try

Query.Open;

Result := TTable.Create;

while not Query.EOF do

begin

Row := TRow.Create;

for Field in Query.Fields do

Row.Add(Field.FieldName, Field.Value);

Result.Add(Row);

Query.Next;

end;

finally

Query.Close;

Transaction.Commit;

Query.Clear;

Transaction.EndTransaction;

end;

end;

function TAppDatabase.ExecOut(AQuery: String): TTable;

begin

Result := ExecOut(AQuery, nil);

end;

constructor TAppDatabase.Create;

begin

//if not FileExists('./application.db') then

//begin

// FileCreate('./application.db');

//end;

Connection := TSQLite3Connection.Create(nil);

Transaction := TSQLTransaction.Create(nil);

Query := TSQLQuery.Create(nil);

Connection.DatabaseName := './application.db';

Connection.Transaction := Transaction;

Query.DataBase := Connection;

Connection.Open;

Exec('CREATE TABLE IF NOT EXISTS "groups" ("name" TEXT NOT NULL UNIQUE,"github\_id" TEXT NOT NULL UNIQUE,"id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE)');

Exec('CREATE TABLE IF NOT EXISTS "tasks" ("id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,"name" TEXT NOT NULL,"files" TEXT NOT NULL)');

end;

destructor TAppDatabase.Free;

begin

Transaction.EndTransaction;

Connection.Connected := False;

Connection.Free;

Transaction.Free;

Query.Free;

end;

function par(AName: String; AValue: Variant): TParamPair;

begin

Result.Key := AName;

Result.Value := AValue;

end;

end.

**Листинг 5. SettingFormUnit.pas**

unit SettingsFormUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, StdCtrls;

type

{ TSettingsForm }

TSettingsForm = class(TForm)

// Кнопка сохранения

Button1: TButton;

// Кнопка отмены

Button2: TButton;

// Поле с именем пользователя GitHub

LabeledEdit1: TLabeledEdit;

// Поле с токеном пользователя GitHub

LabeledEdit2: TLabeledEdit;

// Нажатие на кнопку сохранения

procedure Button1Click(Sender: TObject);

// Нажатие на кнопку отмены

procedure Button2Click(Sender: TObject);

// Создания формы

procedure FormCreate(Sender: TObject);

private

// Сохранение настроек в конфиг

procedure LoadConfig;

// Загрузка настроек из конфига

procedure SaveConfig;

public

end;

implementation

{$R \*.lfm}

uses

jsonConf;

{ TSettingsForm }

procedure TSettingsForm.Button2Click(Sender: TObject);

begin

Self.Close;

end;

procedure TSettingsForm.FormCreate(Sender: TObject);

begin

LoadConfig;

end;

procedure TSettingsForm.Button1Click(Sender: TObject);

begin

SaveConfig;

Self.Close;

end;

procedure TSettingsForm.LoadConfig;

var

c: TJSONConfig;

begin

c := TJSONConfig.Create(nil);

c.Filename := Application.Location+'/config.json';

LabeledEdit1.Text := c.GetValue('/github/login', '');

LabeledEdit2.Text := c.GetValue('/github/token', '');

c.Free;

end;

procedure TSettingsForm.SaveConfig;

var

c: TJSONConfig;

begin

c := TJSONConfig.Create(nil);

c.Filename := Application.Location+'/config.json';

c.SetValue('/github/login', LabeledEdit1.Text);

c.SetValue('/github/token', LabeledEdit2.Text);

c.Free;

end;

end.

**Листинг 6. Utils.pas**

unit Utils;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Dialogs, fpjson, fphttpclient, Forms;

type

// Класс инкапсуляции объекта Integer

TIntObj = class

private

FI: Integer;

public

property I: Integer Read FI;

constructor Create(IValue: Integer);

end;

// Создание и подготовка HttpClient для работы с GitHub API

function CreateGitHubHTTP: TFPHttpClient;

// Функция быстрого открытия модального окна

procedure OpenDialog(FormClass: TFormClass);

implementation

uses

opensslsockets, jsonConf;

{ TIntObj }

constructor TIntObj.Create(IValue: Integer);

begin

Inherited Create;

FI := IValue;

end;

function CreateGitHubHTTP: TFPHttpClient;

var

c: TJSONConfig;

begin

c := TJSONConfig.Create(nil);

Result := TFPHTTPClient.Create(nil);

c.Filename := Application.Location+'/config.json';

Result.AllowRedirect := True;

Result.AddHeader('User-Agent', 'Mozilla/5.0 (compatible; fpweb)');

Result.AddHeader('Accept', 'application/vnd.github.v3+json');

Result.UserName := c.GetValue('/github/login', '');

Result.Password := c.GetValue('/github/token', '');

end;

procedure OpenDialog(FormClass: TFormClass);

var

ModalForm: TForm;

begin

ModalForm := FormClass.Create(nil);

ModalForm.ShowModal;

ModalForm.Free;

end;

end.