Проект ООТРПО

Дайджест по третьему этапу

Участники проекта: гр. 5303 Допира В, Бочкарев И, Ильянов В., гр. 5304 Павлов Д

Выбранный контейнер: многодольный граф

В дайджесте представлены:

- План итерации.
- Демонстрация работы системы
- План тестирования
- Материалы поддержки пользователей
- План развёртывания

Система автоматизированной генерации расписания

План итераций

Версия 1.0

История Ревизий

Дата	**Версия**	**Описание**	**Автор**
22.05.2020	1.0	Первоначальная версия	Павлов Данила

1. Введение

1.1 Цель

Цель документа – описание плана итерации проекта. В данной итерации проводится реализация большей части функциональности продукта.

1.2 Определения и сокращения

Представлены в артефакте Глоссарий.

1.3 План

График сдачи каждой задачи проекта представлен ниже (см. табл.1).

Таблица 1 – График проекта

Фаза	***Задача***	***Окончание работы***	***Исполнители***
Построение	План итерации	22.05.2020	Павлов Данила
	Разработка системы	03.06.2020	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
	Тестирование системы	22.05.2020	Допира Валерия, Бочкарев Иван
	Материалы поддержки пользователей	22.05.2020	Ильянов Вячеслав
	План развёртывания	22.05.2020	Допира Валерия

3. Нагрузка исполнителей

- Допира Валерия

Таблица 2 – Нагрузка на исполнителя 1

Задача	***Окончание работы***	***Затраченное время***	***Исполнители***
Разработка системы	03.06.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
Тестирование системы	22.05.2020	2 дня	Допира Валерия, Бочкарев Иван
План развёртывания	22.05.2020	1 день	Допира Валерия

- Бочкарев Иван

Таблица 3 – Нагрузка на исполнителя 2

Задача	***Окончание работы***	***Затраченное время***	***Исполнители***
Разработка системы	03.06.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
Тестирование системы	22.05.2020	2 дня	Допира Валерия, Бочкарев Иван

- Павлов Данила

Таблица 4 – Нагрузка на исполнителя 3

Задача	***Окончание работы***	***Затраченное время***	***Исполнители***
План итерации	22.05.2020	1 день	Павлов Данила
Разработка системы	03.06.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав

- Ильянов Вячеслав

Таблица 5 – Нагрузка на исполнителя 4

Задача	***Окончание работы***	***Затраченное время***	***Исполнители***
Разработка системы	03.06.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
Материалы поддержки пользователей	22.05.2020	2 дня	Ильянов Вячеслав

История Ревизий

Дата	Версия	Описание	Автор
29.05.2020	1.0	Первоначальная версия	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав

Демонстрация системы

Ссылка на видеоматериалы:

https://drive.google.com/open?id=1_fy_S-CCY1uhef3Mi2IZ36LuDdoEmFs2

Описание:

На видео продемонстрирована текущая версия программы, содержащая ключевые компоненты (контейнер). Данные загружаются из файла формата JSON. Также через приложение можно создать новый файл, открыть, изменить и сохранить текущий (сохранить и сохранить как).

При нажатии на элементы таблиц и затем правую мышку открывается контекстное меню. В окне данных их можно добавлять, редактировать и удалять. В окне групп можно соотнести группу, предмет и указать количество часов. Все действия влияют на используемую модель (двудольный граф), добавляя и удаляя ребро между двумя долями. Расписание пока не выводится в приложении, будет добавлено позже. Визуализируется граф во вкладке «Визуализация графа». Доступно удаление вершин и фильтрация.

Система автоматизированной генерации расписания

Тестирование системы

Версия 1.0

История Ревизий

Дата	Версия	Описание	Автор
22.05.2020	1.0	Первоначальная версия	Допира Валерия, Бочкарев Иван

1. Введение

1.1. Цель

Целью данного плана тестирования является описание тестирования разработанного приложения.

1.2. Контекст

Целью тестирования является тщательная проверка всех функций приложения. Будут протестированы как позитивные, так и негативные сценарии, которые также могут выявить некоторые ошибки.

Ручное тестирование уже было проведено во 2 итерации.

1.3. Определения, акронимы, сокращения

См. глоссарий проекта.

2. Стратегии тестирования

В результате первого цикла тестирования, где проводятся функциональные тесты, будут внесены некоторые исправления и дополнения и внесены в план тестирования. Первый цикл даст определенное понимание стабильности системы и поможет определить необходимый набор тестов, который будет выполнен в процессе тестирования приложения. Такой метод даст возможность получить подробный отчет о продукте.

Планируется четыре этапа тестирования:

– первый этап – анализ, создание плана тестирования, частичное выполнение некоторых функциональных тестов;

– второй этап будет посвящен подробному выполнению функциональных тестов, раскрывающих и описывающих ошибки;

– третий этап – проверка исправленных ошибок и выполнение регрессионного тестирования;

– четвёртый этап заключается в тестировании пользовательского интерфейса с раскрытием и описанием ошибок.

Такая система тестирования позволяет выполнять детальное тестирование и предотвращать, исправлять ошибки на ранних этапах.

2.1. Виды тестирования

2.1.1. Функциональное тестирование

Цель: обнаружение функциональных ошибок посредством выполнения функциональных тестов.

2.1.2. Регрессионное тестирование

Цель: проверка изменений, внесенных в приложение, чтобы убедиться, что в новой версии нет ошибок в тех частях, где тестирование уже было выполнено.

2.1.3. Тестирование удобства пользования

Цель: дать оценку уровня удобства использования приложения по следующим пунктам:

– производительность, эффективность – количество времени и/или шагов необходимое пользователю для завершения основных задач приложения;

– правильность – количество ошибок, сделанное пользователем во время работы с приложением;

– активизация в памяти – как много пользователь помнит о работе приложения после приостановки работы с ним на длительный период времени;

– эмоциональная реакция – ощущения пользователя после завершения задачи.

2.1.4. Тестирование производительности

Цель: определение масштабируемости приложения под нагрузкой, при этом происходит:

– измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций;

– определение количества пользователей, одновременно работающих с приложением.

2.1.5 Тестирование стабильности (надежности)

Цель: проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки. Время выполнения операций может играть в данном виде тестирования второстепенную роль.

2.1.6 Тестирование на отказ и восстановление

Цель: проверка систем восстановления (или дублирующих основной функционал систем), которые, в случае возникновения сбоев, обеспечат сохранность и целостность данных тестируемого продукта. Объектом тестирования являются весьма вероятные эксплуатационные проблемы, такие как:

– отказ электричества на компьютере-сервере;

– отказ электричества на компьютере-клиенте;

– незавершенные циклы обработки данных (прерывание работы фильтров данных, прерывание синхронизации);

– объявление или внесение в массивы данных невозможных или ошибочных элементов;

– отказ носителей данных.

3. ТЕСТИРОВАНИЕ

Приложение было протестировано, внесены изменения.

3.1. Функциональное тестирование

3.1.1 Визуализация графа в классе `GraphWidget`

Задается сцена и ее параметры. Затем на нее помещаются 5 точек разных цветов (класс `Node`). Позиции устанавливаются с помощью функции `setPos()`. Некоторые из них соединяются ребрами с помощью функции `addItem()` и класса `Edge`. Код:

```
scene->setItemIndexMethod(QGraphicsScene::NoIndex);
```

```
scene->setSceneRect(-200, -200, 400, 400);

setScene(scene);
setCacheMode(CacheBackground);
setViewportUpdateMode(BoundingRectViewportUpdate); //reconsider
setRenderHint(QPainter::Antialiasing);
setTransformationAnchor(AnchorUnderMouse);
scale(qreal(1), qreal(1));
//setMinimumSize(400, 400);

//@brief
Node *node1 = new Node (this,15,Qt::cyan);
Node *node2 = new Node (this,15,Qt::gray);
Node *node3 = new Node (this,15,Qt::red);
Node *node4 = new Node (this,15,Qt::blue);
Node *node5 = new Node (this,15,Qt::green);


scene->addItem(node1);
scene->addItem(node2);
scene->addItem(node3);
scene->addItem(node4);
scene->addItem(node5);

scene->addItem(new Edge(node1, node2));
scene->addItem(new Edge(node2, node3));
scene->addItem(new Edge(node2, node4));
scene->addItem(new Edge(node2, node5));
scene->addItem(new Edge(node5, node4));

node1->setPos(0,0);
node2->setPos(0,60);
node3->setPos(20,10);
node4->setPos(40,25);
node5->setPos(10,-30);

nodes.push_back(node1);
nodes.push_back(node2);
nodes.push_back(node3);
nodes.push_back(node4);
nodes.push_back(node5);

readGraph(QPointF(-30,-20));
```

Результат представлен на рисунке 1. Ожидаемый результат соответствует полученному.

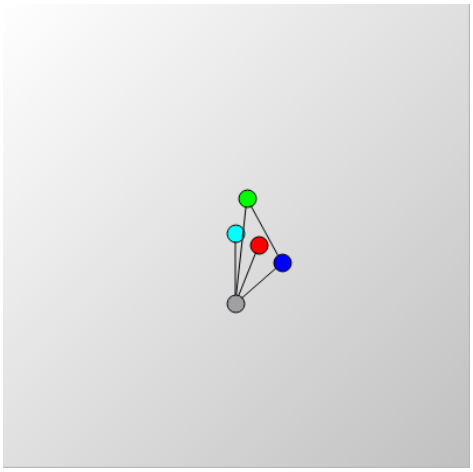


Рисунок 1 — Визуализация графа

3.1.2 Проверка на наследование `Node` от `abstractNodeRepository` в классе `RepositoryTemplate`

```
template <class T>
void RepositoryTemplate<T>::checkNode() {
   abstractNodeRepository* test = dynamic_cast<abstractNodeRepository*> (new T);
   if (!test) {
        throw std::runtime_error("The type of template does not satisfy the
required. The type must be a descendant of the abstractNodeRepository class.");
   }
}
```

Ожидаемый результат соответствует полученному. Ошибок не было.

3.1.3 Проверка на изменения (удаление, добавление, редактирование) репозиториев в главной вкладке в классе `DialogLinkGroupSubjectWindow`

Для группы:

```
void
DialogLinkGroupSubjectWindow::editDataRepoGroup(RepositoryTemplate<GroupStudents
> repoGroupStudents){
    //Проверка на изменения(удаление, добавление, редактирование) репозиториев в
главной вкладке
    if (repoRecGroupStudent.getAmount()==0){
        for (int i =0; i<repoGroupStudents.getAmount(); i++){
            repoRecGroupStudent.add(repoGroupStudents.getByIndex(i));
        }

    }else
          if (repoGroupStudents.getAmount()>repoRecGroupStudent.getAmount()){
                                    int   raz   =   repoGroupStudents.getAmount()-
repoRecGroupStudent.getAmount();
            int addE = repoRecGroupStudent.getAmount();
            for (int i =0; i<raz; i++){
                repoRecGroupStudent.add(repoGroupStudents.getByIndex(i));
                ++addE;
            }
    }else
```

```
          if (repoGroupStudents.getAmount()<repoRecGroupStudent.getAmount()){
                                  //int   raz   =   repoRecGroupStudent.getAmount()-
repoGroupStudents.getAmount();
            //int delE = repoRecGroupStudent.getAmount()-1;
            for (int i =0; i<dinGr.size(); i++){
                repoRecGroupStudent.remove(repoRecGroupStudent.getById(repoRecGr
oupStudent.getByIndex(dinGr[i]).id).id);
            //--delE;
            }
    }
    else
         if (repoGroupStudents.getAmount()==repoRecGroupStudent.getAmount()){
            for (int i =0; i<repoGroupStudents.getAmount(); i++){
                                                                                if
(repoGroupStudents.getById(repoGroupStudents.getByIndex(i).id).name!
=repoRecGroupStudent.getById(repoRecGroupStudent.getByIndex(i).id).name){
                     repoRecGroupStudent.update(repoRecGroupStudent.getById(repoR
ecGroupStudent.getByIndex(i).id).id,repoGroupStudents.getById(repoGroupStudents.
getByIndex(i).id).name);
                }
            }
          }
}
```

Ожидаемый результат соответствует полученному. Ошибок нет.

Для предмета:

```
void
DialogLinkGroupSubjectWindow::editDataRepoSubject(RepositoryTemplate<Subject>
repoSubjects){
    //Проверка на изменения(удаление, добавление, редактирование) репозиториев в
главной вкладке
    if (repoRecSubject.getAmount()==0){
        for (int i =0; i<repoSubjects.getAmount(); i++){
            repoRecSubject.add(repoSubjects.getByIndex(i));
        }

    }else
          if (repoSubjects.getAmount()>repoRecSubject.getAmount()){
            int raz = repoSubjects.getAmount()-repoRecSubject.getAmount();
            int addE = repoRecSubject.getAmount();
            for (int i =0; i<raz; i++){
                repoRecSubject.add(repoSubjects.getByIndex(addE));
                ++addE;
            }
    }else
          if (repoSubjects.getAmount()<repoRecSubject.getAmount()){
            //int raz = repoRecSubject.getAmount()-repoSubjects.getAmount();
            //int delE = repoRecSubject.getAmount()-1;
            for (int i =0; i<dinSb.size(); i++){
                repoRecSubject.remove(repoRecSubject.getById(repoRecSubject.getB
yIndex(dinSb[i]).id).id);
            //--delE;
            }
    }
    else
         if (repoSubjects.getAmount()==repoRecSubject.getAmount()){
            for (int i =0; i<repoSubjects.getAmount(); i++){
                  if (repoSubjects.getById(repoSubjects.getByIndex(i).id).name!
=repoRecSubject.getById(repoRecSubject.getByIndex(i).id).name){
```

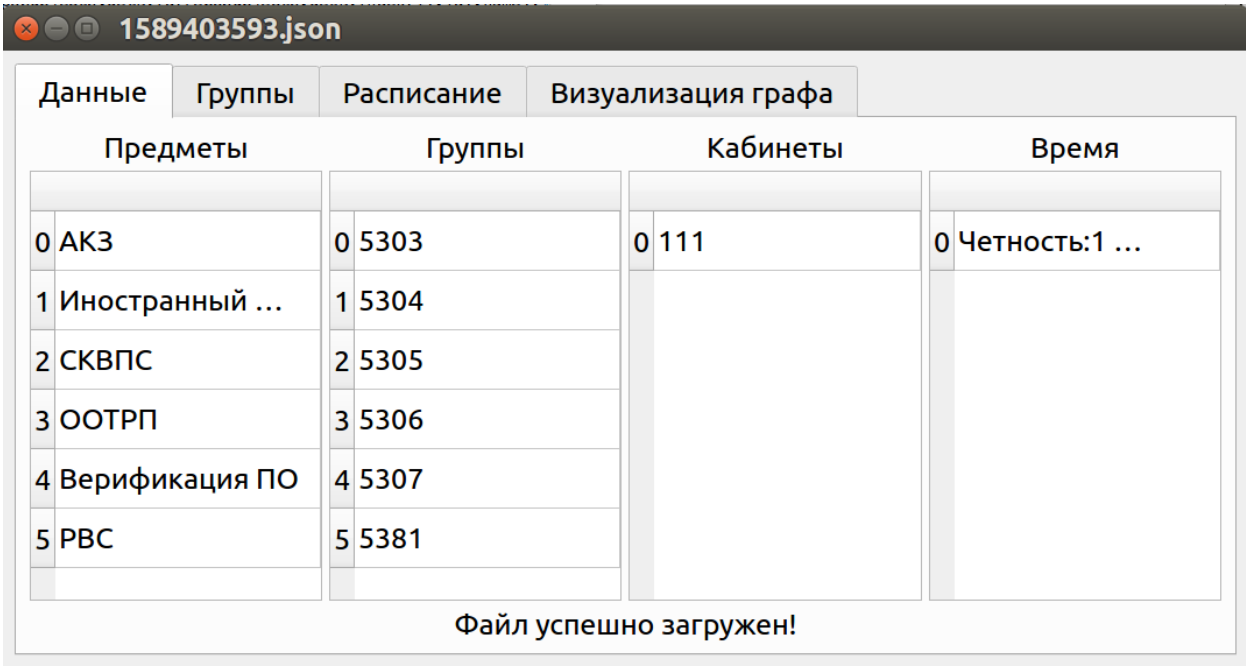
```
                    repoRecSubject.update(repoRecSubject.getById(repoRecSubject.
getByIndex(i).id).id,repoSubjects.getById(repoSubjects.getByIndex(i).id).name);
                }
            }
        }
    }
```

Ожидаемый результат соответствует полученному. Ошибок нет.

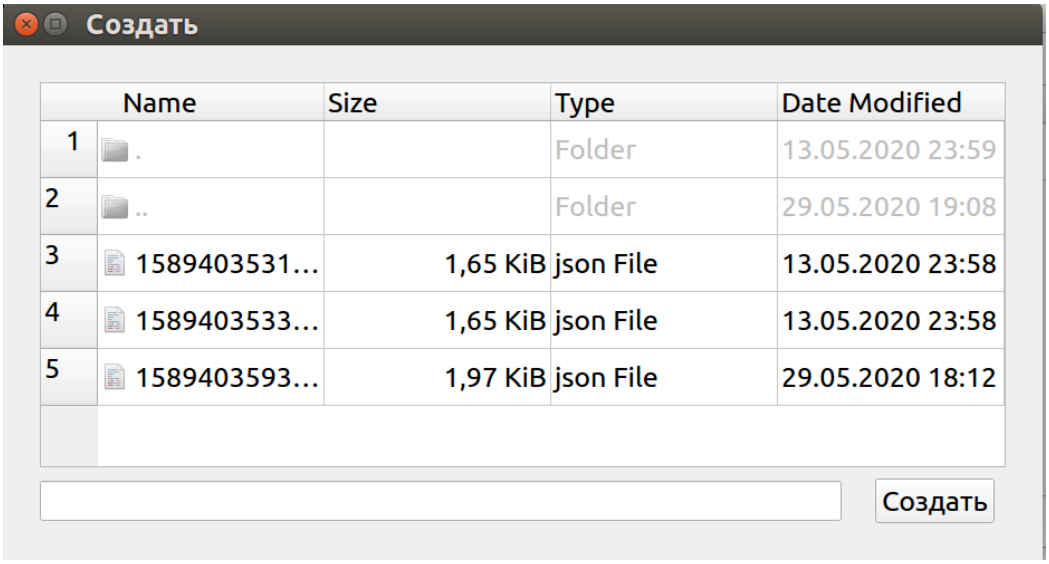
3.2. Регрессионное тестирование

Ручное тестирование описано во 2 итерации. После этого были реализованы следующие функции:

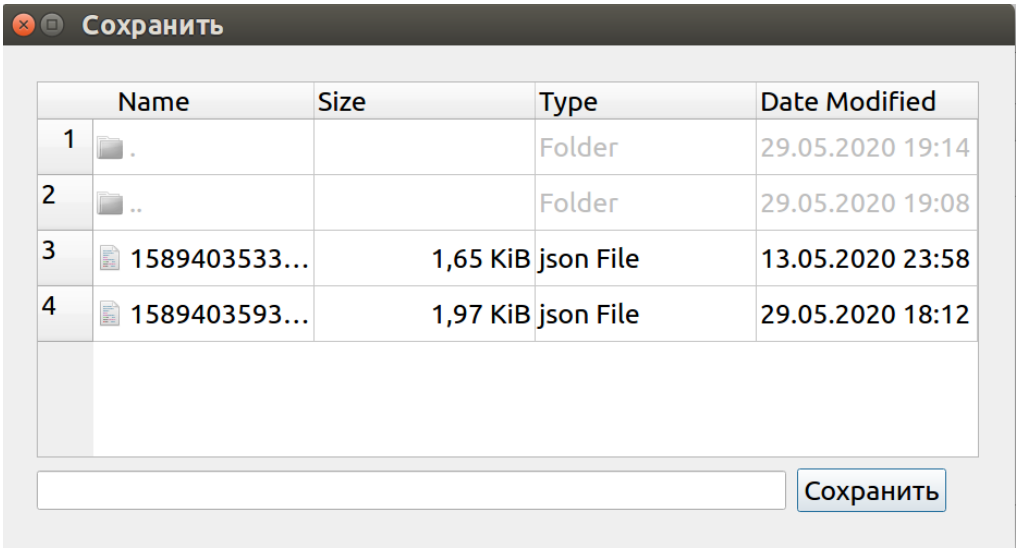
Открытие файла JSON при загрузке приложения и вывод сообщения:



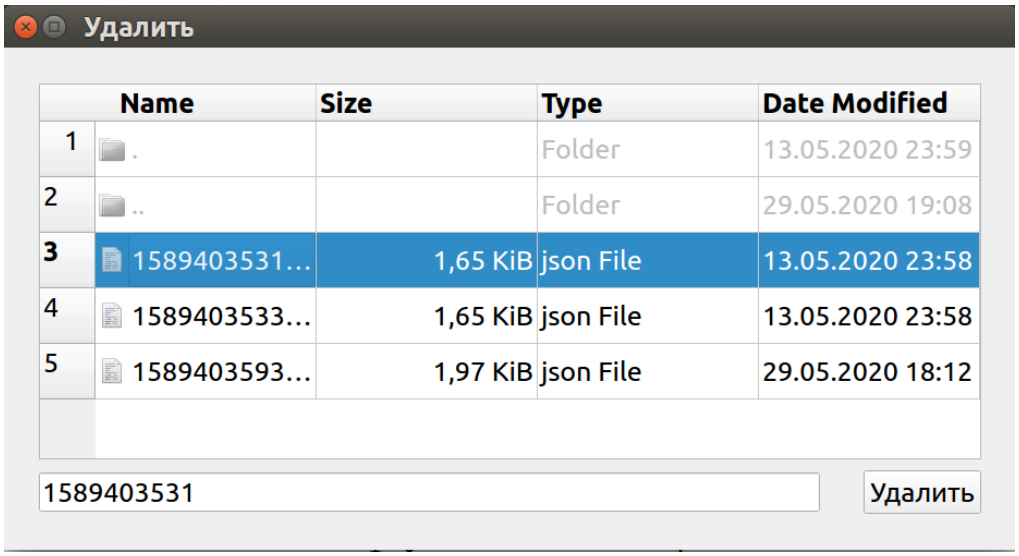
При создании открывается следующее окно:



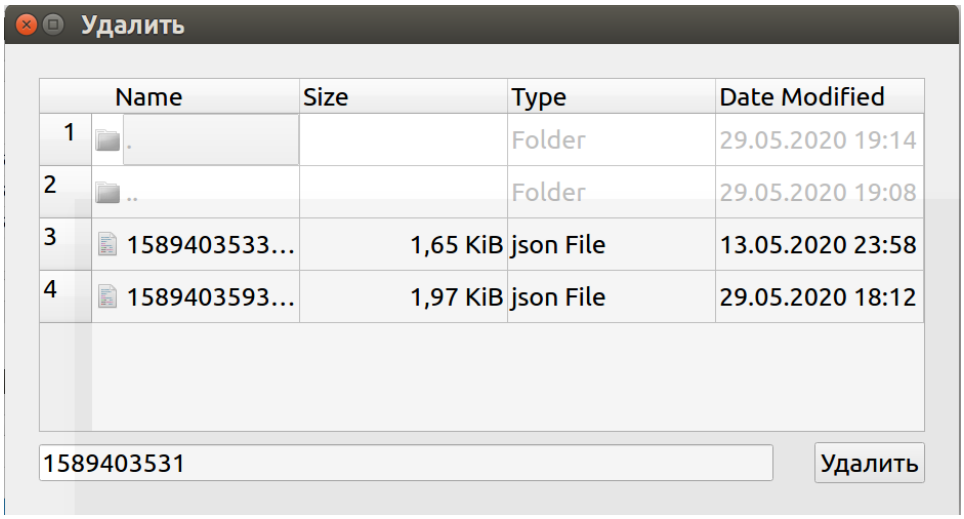
Сохранить как:



Удаление:



После удаления:



3.3. Тестирование удобства пользования

Если данные уже загружены и сохранены в JSON, то пользователю необходимо лишь перейти на вкладку Расписание для отображения расписания или на вкладку Визуализация для отображения графа. Для работы с файлом необходимо обратиться к меню приложения. Все действия перечислены подряд без вложенности, и при этом их немного, поэтому пользователь сможет легко найти необходимое. При работе с данными пользователю нужно открыть контекстное меню, ввести данные и нажать окей. Все действия занимают до 3-х шагов, поэтому можно сказать, что приложение имеет высокую эффективность.

Работа с данными осуществляется через контекстное меню, которое отображается при нажатии на правую мышку. Данный способ возможно не будет интуитивным для всех пользователей, но взаимодействие с приложением будет описано в документации, поэтому у пользователя не будет ошибок при работе с приложением.

Наличие документации обеспечивает воспроизводимость инструкций для работы с приложением, поэтому даже если пользователь не помнит, как работать из-за приостановки работы с ним на длительный период, пользователь всегда будет иметь указания.

Приложение направлено на осуществление определенной задачи — построение расписание. И если она будет выполнена, то пользователь будет удовлетворен. Он также сможет редактировать полученный результат, что тоже скажется на удобстве приложения.

3.4. Тестирование производительности

Приложение зависит от количества данных, которое ему необходимо будет обработать. Однако так как приложение рассчитано для работы со школой или факультетом, то оно сможет не заметно для пользователя обработать данные, которые обычно имеют такие организации.

Одновременно с приложением может работать 1 пользователь.

3.5 Тестирование стабильности (надежности)

Стабильность системы не зависит от времени работы приложения.

3.6 Тестирование на отказ и восстановление

При отключении электричества, выключении компьютера приложение будет завершено, а не сохраненные данные потеряны. Поэтому пользователю необходимо позаботиться о резервных источниках питания, периодически сохранять файл с данными.

Также добавлены проверки на ввод некоторых данных. Например, время проведения занятий. В некоторых местах важно, чтобы данные были только числовыми, например, количество академических часов. Дни недели пользователь выбирает из предложенного списка: Понедельник, вторник, среда, четверг, пятница, суббота, воскресенье. Предметы, названия групп могут иметь как числовые, так и буквенные значения.

4. Вывод

Конечным результатом является протестированное приложение без ошибок.

Система автоматизированной генерации расписания

Материалы поддержки пользователей

Версия 1.0

История Ревизий

Дата	Версия	Описание	Автор
22.05.2020	1.0	Первоначальная версия	Ильянов Вячеслав

1. Введение

1.1 Цель

Техническая поддержка служит для помощи конкретным пользователям решать возникающие конкретные проблемы с продуктом и его использованием, нежели задачи, связанные с обучением, индивидуальной настройкой или другими услугами поддержки.

2. Методы поддержки

Инструкция работы с приложением описано в документации и позволит решить многие задачи.

Пользователь сможет связаться как напрямую с разработками приложения, так и воспользоваться краудсорсингом. На гитхабе, где выложено приложение, будет возможность пообщаться с другими пользователями и найти способы решения проблемы.

Система автоматизированной генерации расписания

План развёртывания

Версия 1.0

История Ревизий

Дата	**Версия**	**Описание**	**Автор**
22.05.2020	1.0	Первоначальная версия	Допира Валерия