МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентированные технологии разработки программного обеспечения»

Тема: Разработка программы для составления расписания

Студент гр. 5303 Допира В.Е.

Преподаватель Спицын А.В.

Санкт-Петербург
2020

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Допира В.Е.

Группа 5303

Тема практики: Разработка программы для составления расписания

Исходные данные:
Сведения о предметах, группах, кабинетах, времени проведения занятий и академических часах в учебном заведении. Хранятся в формате JSON.

Содержание пояснительной записки:
«Содержание», «Введение», «План разработки», «Разработка программного обеспечения», «Завершение разработки и тестирование», «Материалы», «Заключение»

Предполагаемый объем пояснительной записки:
Не менее 81 страниц.

Дата выдачи задания: 10.02.2020

Дата сдачи реферата: 04.06.2020

Дата защиты реферата: 04.06.2020

Студент		Допира В.Е.
Преподаватель		Спицын А.В.

АННОТАЦИЯ

В ходе курсовой работы разработано приложение для составления расписания, а также осуществлена визуализация полученного результата. Графический интерфейс пользователя позволяет удобно и быстро работать с системой. Реализованы контейнер - многодольный граф, аллокатор, итератор, механизм исключений и некоторые шаблоны проектирования. Реализована возможность открытия, редактирования, удаления и сохранения файла JSON с данными о группах, предметах, кабинетах, времени проведения занятий. Реализован алгоритм построения расписания, результат которого отображается в приложение, а также его можно сохранить в JSON. Граф, визуализирующий расписание, выводится в приложении. Приложение проходит все тесты.

SUMMARY

During the course work, an application for scheduling was developed, and the result obtained was visualized. The graphical user interface allows you to conveniently and quickly work with the system. A container is implemented - a multi-part graph, allocator, iterator, exception mechanism, and some design patterns. Implemented the ability to open, edit, delete and save the JSON file with data on groups, items, classrooms, time classes. A scheduling algorithm has been implemented, the result of which is displayed in the application, and it can also be saved in JSON. A graph visualizing the schedule is displayed in the application. The application passes all the tests.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Переход Российской Федерации на путь инновационной и инвестиционной стратегии развития экономики, повышение качества услуг и разрабатываемых технологий требует существенных изменений в организации управления деятельностью хозяйствующих субъектов. Учебные заведения, а в частности и университеты, не могут быть в стороне от этих процессов и должны соответствовать потребностям развития отраслей экономики регионов. В настоящее время необходимым условием для успешного функционирования образовательных учреждений становится разработка, внедрение и сопровождение информационных систем, обеспечивающих эффективное функционирование всех внутренних процессов.

Использование информационных систем в образовательных учреждениях активно внедряется. Спектр применения информационных технологий широк и варьируется от автоматизации отдельно взятых областей до полной автоматизации деятельности учебного заведения.

Вне зависимости от области автоматизации, внедряемые информационные системы имеют конечную цель: повышение качества образования. Значительное влияние на автоматизацию учебных процессов оказывают наличие денежных средств, готовность использования предлагаемых рынком информационных услуг и программных продуктов.

Составление расписания является сложной и требующей большой ответственности от сотрудников университета, выполняющих эту обязанность. Из-за большого количества студентов составление расписания отнимает большое количество времени. Автоматизация составления расписания занятий позволит сократить время и оптимизировать работу, поэтому выбранная тема актуальна.

Целью курсовой работы является разработка приложения для составления расписания и визуализация полученного результата.

Поставлены следующие задачи:

1. Обеспечение минимальных усилий пользователя при составлении расписания.

2. Создание удобного интерфейса.

3. Обеспечение быстрого доступа к данным, хранящимся в программе.

4. Обеспечение возможности добавления, изменения и удаления данных в программе.

5. Отображение расписание в виде многодольного графа.

6. Обеспечение сериализации данных при работе с программой.

1. План разработки

1.1. Определения и термины

Артефакт (синоним. Рабочий продукт, Ресурс) – некоторые продукты проекта, порождаемые или используемые в нем при работе над окончательным продуктом.

Интерфейс – граница между двумя функциональными объектами, требования к которой определяются стандартом; совокупность средств, методов и правил взаимодействия (управления, контроля и т. д.) между элементами системы.

Персональный компьютер – настольная микро-ЭВМ, имеющая эксплуатационные характеристики бытового прибора и универсальные функциональные возможности.

Прикладная программа, или приложение – программа, предназначенная для выполнения определенных задач и рассчитанная на непосредственное взаимодействие с пользователем.

Прецедент – последовательность действий, выполняемых системой для получения наблюдаемого результата.

Расписание занятий - документ, определяющий педагогически целесообразную последовательность учебных занятий в образовательном учреждении на каждый день учебной недели и конкретизирующий таким образом учебный план.

Учебное заведение — учреждение для (чего) обучения, организация образования. К ним относятся: школы, колледжи, пансионы, интернаты, гимназии, лицеи, техникумы, политехникумы, семинарии, медресе, курсы, институты, университеты.

Учебная дисциплина (предмет) - система знаний, умений и навыков, отобранных из определенной отрасли науки, техники, искусства, производственной деятельности для изучения в образовательном учреждении.

Преподаватель - работник высших, средних специальных и профессионально-технических учебных заведений, ведущий какой-либо предмет и воспитательную работу.

Факультет — отделение высшего учебного заведения, обнимающее науки, относящиеся к одной какой нибудь отрасли знаний.

Деканат — административно учебное управление факультета при декане.

Учебная группа (группа) - определенное число лиц с примерно одинаковым уровнем подготовки, изучающих одно и то же в одно и то же время под руководством одних и тех же преподавателей на протяжении одинакового для всех периода; обособленная часть контингента образовательного учреждения, являющаяся для ее членов первичным коллективом.

Учебное помещение (кабинет) - специально оборудованное помещение, для обучения студентов. Обычно номер кабинета имеет четырехзначное значение и нумеруется по типу: КЭНН, где К - номер корпуса, Э - номер этажа в определенном корпусе, Н - номер кабинета на определенных этажах и корпусах.

Этаж - уровень здания над (или под) уровнем земли.

Корпус - Одно из нескольких зданий, принадлежащих учебному заведению.

Учебная неделя - дни, по которым проходят занятия по определенным дисциплинам. Воскресенье является выходным днем.

Академический час - традиционное название учебного часа в вузах и других учреждениях профессионального образования. Академический час не равен астрономическому и устанавливается нормативными документами. Обычно равен 45 минутам.

Время начала занятий - время в которое проходят занятия по учебным дисциплинам.

Семестр - в высших учебных заведениях: полугодие учебного года; различают осеннее и весеннее.

Четность недель - способность номера недели, который является целым числом, делиться без остатка на 2. Принято считать учебные недели с начала четверти/семестра. В зависимости от четности недели расписание занятий может быть разным.

Многодольный граф - граф, множество вершин которого можно разбить на k независимых множеств (доль).

Вершина графа (узел, точка) - элемент графа, обозначающий объект любой природы, входящий в множество объектов, описываемое графом.

Ребро графа - термин теории графов, линия, соединяющая пару смежных вершин графа.

Qt - кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++.

C++ - компилируемый, статически типизированный язык программирования общего назначения.

1.2. Видение

Одна из основных составляющих учебного процесса - расписание занятий. Составленное расписание влияет на трудовой ритм, творческую отдачу преподавателей, поэтому его можно рассматривать как фактор оптимизации использования ограниченных трудовых ресурсов: преподавательского состава. Технологию же разработки расписания следует воспринимать не только как трудоемкий технический процесс, объект механизации и автоматизации с использованием ЭВМ, но и как акцию оптимального управления. Таким образом, это проблема разработки оптимальных расписаний занятий с очевидным экономическим эффектом, поскольку интересы участников учебного процесса многообразны, задача составления расписания многокритериальная.

Представляемый программный продукт призван:

1. Облегчить ручной и умственный труд сотрудников при составлении учебного расписания. От сотрудника требуется только ввести необходимые данные и проверить корректность итогового расписания.
2. Сгенерировать максимально удобное и оптимальное расписание как для студентов и школьников, так и для преподавателей.
3. Удобный интерфейс для взаимодействия программы с пользователем.

Использование данного продукта позволит сэкономить время и силы потраченные на выполнение этого трудоёмкого процесса. Конечно, не только учебные заведения заинтересованы в составлении оптимального расписания, но и крупные компании и предприятия, где нужно составить эффективный

распорядок дня как для определенных людей, так и для групп людей. В наши дни это все более актуально, так как динамика жизни становится все интенсивнее и интенсивнее.

На данный момент в образовательных учреждениях остро стоит вопрос о составлении учебных расписании. Это связано с тем, что итоговое расписание зачастую является не эффективными, а кроме этого требует больших трудозатрат пропорциональных масштабам учебного учреждения.

В данной области лучше всего подходит создание программного обеспечения по автоматизированному составлению оптимального расписания.

Постановка проблемы представлена в табл. 1.

Таблица 1 - Постановка проблемы

Проблема	Повышение сложности составления эффективного учебного расписания
На кого влияет	Составители расписаний в школах/Вузах, преподаватели, школьники и студенты.
Как влияет	Увеличение временных затрат на составление расписания. Нерациональное использование времени учащихся. Высокая стоимость внесения изменений в расписание.
Решение	Увеличится эффективность учебного процесса, увеличится продуктивность преподавателей, студентов и школьников, появится хорошо распланированный порядок дня.

1.2.1 Обзор продукта

Данное приложение предназначено для оптимизации процесса составления расписания в образовательных учреждениях.

Данный программный продукт может использоваться без привязки к определенному учебному заведению. Программный продукт будет распространяться бесплатно под лицензией MIT.

Предоставляемый продукт является системой автоматизированного составления расписания.

Аналоги данного ПО на данный момент либо слабо развиты, либо предоставляют только косвенный функционал, к примеру, такой как простое представление расписания в электронном виде.

В сравнении с аналогами наш продукт имеет много схожих возможностей, а именно возможность постоянно пополнять данные по учебным дисциплинам, учебным группам, учебным помещениям, учебным неделям, времени начала лекций.

Но отличительной чертой, а также особенностью нашего продукта, является наличие уникального функционала по генерации оптимального расписания. В этом аспекте продукт сильно выигрывает.

Пользователи:

- Составители расписаний
- Студенты
- Преподаватели
- Рекламодатели

Описание заинтересованных лиц представлено в таблице 2.

Таблица 2 - Описание заинтересованных лиц

Заинтересованное лицо	***Роль***
IT Executive	Следит за ходом разработки проекта
Пользователи системы	В качестве пользователя системой выступает человек отвечающий за составление расписания. Пользователь имеет возможность добавлять и удалять элементы расписания (дисциплина, время, помещение, группы учащихся)
Заказчик	Представляет интересы образовательной организации, заинтересованной нашем продукте.

Описание пользователей представлено в таблице 3.

Таблица 3 - Описание пользователей

Имя	***Описание***
Пользователи системы	Редактирование и просмотр данных: учебные дисциплины, учебные группы, учебные помещения, учебные недели, время начала лекций, на основе которых генерируется расписание, которое можно просматривать в виде таблиц или многодольного графа

Для нормальной работы с ПО пользователю достаточно иметь обычный стационарный компьютер с установленной системой по генерации расписания. Отличительной чертой является то, что системе не требуется выход в интернет, что означает программой можно пользоваться когда и где угодно.

Ключевые потребности заинтересованных лиц представлены в таблице 4.

Таблица 4 - Потребности заинтересованных лиц

Потребность	***Приоритет***	***Текущее решение***	***Предлагаемое решение***
Автоматизированное составление расписания	Высокий	Многие пользователи вынуждены составлять расписание вручную	Система предлагает на основе загружаемых данных генерировать учебное расписание

1.2.2 Особенности продукта

Система будет построена по принципу монолитной архитектуры, которое означает, что данное приложение будет представлять большой связанный

модуль, где все компоненты спроектированы так, чтобы работать вместе с друг с другом, общая память и ресурсы.

Приложение будет работать в автономном режиме без подключения к интернету, что облегчит работу пользователям в местах без доступа к интернету.

Система генерирования расписания может стать конкурентоспособной за счет функциональности и расчета оптимальных вариантов расписаний.

Конечно, система не ограничится текущим функционалом и в будущем планируется дополнить систему клиент-серверной технологией для обеспечения многопользовательского режима работы системы, а также централизации хранения данных. Это позволит скачать актуальное расписание на локальный компьютер, после редактирования отправить копию на сервер.

Также клиент-серверная архитектура позволяет генерировать расписание используя вычислительные мощности серверного компьютера, который как правило на порядок мощнее. Это ускорит процесс генерации для больших учебных заведений, большим объемом данных.

В следующих поколениях системы возможно добавление дополнительного функционала. Такого как составление расписания не на неделю, а на семестр, с учетом всех праздничных дней и учебной сессии.

Потенциально возможно создание мобильного приложения для студентов и школьников с расписание на неделю.

Данная программа будет предоставлять два режима отображения расписания:

- В виде таблицы;
- В виде многодольного графа.

Разрабатываемый программный продукт будет иметь следующие возможности:

1. Обеспечение оперативного доступа к данным, хранящимся в программе.
2. Обеспечение возможности изменения и дополнения данных в программе.
3. Отображение расписание в виде многодольного графа.
4. Обеспечение сериализации данных при работе с программой.
5. Кроссплатформенность: разрабатывается под Windows и Linux.

Разрабатываемый программный продукт имеет следующие ограничения:

1. Расписание составляется один раз без предоставления альтернативных вариантов.

В таблице 5 указаны основные возможности системы генерирования расписания с точки зрения преимуществ и возможностей.

Таблица 5 - Основные возможности

Преимущество для пользователей	***Поддерживающие функции***
Представление расписания в виде электронной таблицы	Система выводит составленное расписание в виде электронной таблицы
Представление расписания в виде многодольного графа	Система выводит составленное расписание в виде многодольного графа
Фильтрация расписания, представленное в виде многодольного графа	Система позволяет отфильтровать многодольный граф по: учебной дисциплине, времени начала занятий, учебным помещениям, учебным группам
Сохранение расписания в файл и загрузка расписания из файла	Система позволяет сохранять исходные и выходные данные в файл, который потом можно будет загрузить при новом сеансе работы с системой.
Легкость освоения продукта	Отсутствие авторизации и интуитивно понятный интерфейс

Следующие предположения и зависимости относятся к возможностям системы генерации расписания, изложенным в данном документе:

- Система зависит от корректных исходных данных, загружаемых пользователем;
- Предполагается, что сгенерированное расписание системой полностью удовлетворит потребности пользователя;
- Успешные продажи системы автоматизированной генерации расписания зависят от правильной рекламы и раскрутки, а также качества самого продукта.

Продукта должен иметь следующие возможности. Пользователь системы вход в системы без ввода логина и пароля. В системе должны иметься отдельные таблицы для добавления, удаления, изменения следующих данных: учебные дисциплины, учебные группы, учебные помещения, время начала лекций, чтобы упростить процесс ввода данных. Система должна позволять просматривать расписание в виде электронной таблицы и многодольного графа. Система

должна позволять пользователям сохранять загруженные данные, сгенерированное расписание, для повторного использования. Система должна позволять пользователю фильтровать многодольный граф по: учебной дисциплине, дню, преподавателю и помещению.

Требования:

- *Доступность.* Доступность ограничивается только работоспособностью компьютера, на котором она запущена.
- *Удобство использования.* Система должна быть простой в использовании, с интуитивно понятным интерфейсом.
- *Удобство обслуживания.* Система должна быть разработана для простоты обслуживания. Входные данные должны быть модифицируемыми.
- *Совместимость с устройствами.* Приложение должно работать на платформах (Windows, Linux).

Система должна взаимодействовать с файлами формата json при сериализации.

Система не должна занимать более 50 мб на жестком диске.

Клиентский компонент системы не должен требовать более 128 МБ ОЗУ.

Время ответа не должно превышать 0,1 секунды, т.е. пользователь должен воспринимать работу системы как «мгновенную».

Пользователь не должен ощущать задержки при работе, каждый его запрос должен быть обработан в кратчайшие сроки.

1.2.3 Альтернативы и конкуренты

Конкурентами являются всевозможные приложения для составления расписания доступные в сети интернет. Но как говорилось выше функционал таких приложение ограничен визуализацией расписания.

Альтернативным решением будет передать задачу составления расписания другой организации, что является нецелесообразным.

1.2.4 Стоимость и цена продукта

Затраты на разработку системы не должны превышать 200 000 рублей.

Ожидается, что конечный продукт будет предложен заказчикам с условием оплаты – 10% от одного заказа.

1.3. Модель вариантов использования

Каталог Акторов (действующих лиц) представлен в таблице 6.

Таблица 6 - Список акторов

Имя	***Описание***
Пользователь	Физическое лицо, составляющее расписание, используя уже готовые данные или заполняющих их сам
Приложение	Приложение, позволяющее по введенным данным составить расписание

1.3.1 Сценарии использования

Ниже описаны сценарии, которые охватывают наиболее важные функции и значительное число элементов архитектуры.

2.1 Основной поток: составление расписания

1. Пользователь загружает данные с файла, содержащий информацию по учебным дисциплинам, группам, помещениям, времени начале занятий
2. После загрузки данных пользователь нажимает на кнопку “Составить расписание”
3. Приложение по полученным данным выстраивает расписание в виде электронной таблице и многодольного графа
4. Далее пользователь может посмотреть на полученную электронную таблицу и граф с возможностью фильтрации, переключая вкладки “Расписание” и “Граф” соответственно

Альтернативные потоки:

1. Добавление в таблицу данных

a. 1.Пользователь открывает таблицу, где расположена таблица с данными по учебным дисциплинам, учебным группам, учебным помещениям, время начала занятий.

b. 2. Пользователь нажимает кнопку “добавить” с одной из колонок

c. 3. Приложение вводит данные в таблицу

2. Удаление в таблице данных

a. 1. Пользователь открывает таблицу, где расположена таблица с данными по учебным дисциплинам, учебным группам, учебным помещениям, время начала занятий.

b. 2. Пользователь выбирает в таблице ячейку и нажимает кнопку “Удалить”

c. 3. Приложение удаляет из таблицы данные, выбранные пользователем

3. Изменение в таблице данных

a. 1. Пользователь открывает таблицу, где расположена таблица с данными по учебным дисциплинам, учебным группам, учебным помещениям, время начала занятий.

b. 2. Пользователь находит нужную ячейку и нажимает на кнопку “Изменить”

c. 3. Пользователь переписывает информацию в окне, которое вывело приложение

d. 4. Приложение меняет информацию в ячейке

4. Некорректные данные в таблице данных

a. 1.Пользователь вводит данные по учебным дисциплинам, учебным группам, учебным помещениям, времени начала занятий.

b. 2. Программа выдает сообщение об ошибке и не дает занести данные

c. 3. Пользователь продолжает работу

1.3.2 Предварительные условия:

У пользователя должны иметься исходные данные по: учебным дисциплинам, учебным группам, учебным помещениям, времени начала занятий для загрузки их в систему, для составления учебного расписания.

1.3.3 Постусловия

Готовое расписание сохраняется в файл в формате JSON.

1.3.4 Диаграмма варианта использования

Пользователь: при первой работе с системой, необходимо добавить данные в таблицы: предметы, группы, кабинеты, время на основе которых система

генерирует учебное расписание. Составленное расписание представляется в виде электронной таблицы и многодольного графа. Представление многодольного графа можно отфильтровать по: группе, предметам, времени, кабинетам. После окончания работы с системой, пользователь может сохранить полученные данные в файл, который можно использовать при следующем входе в систему (см. рис. 1).

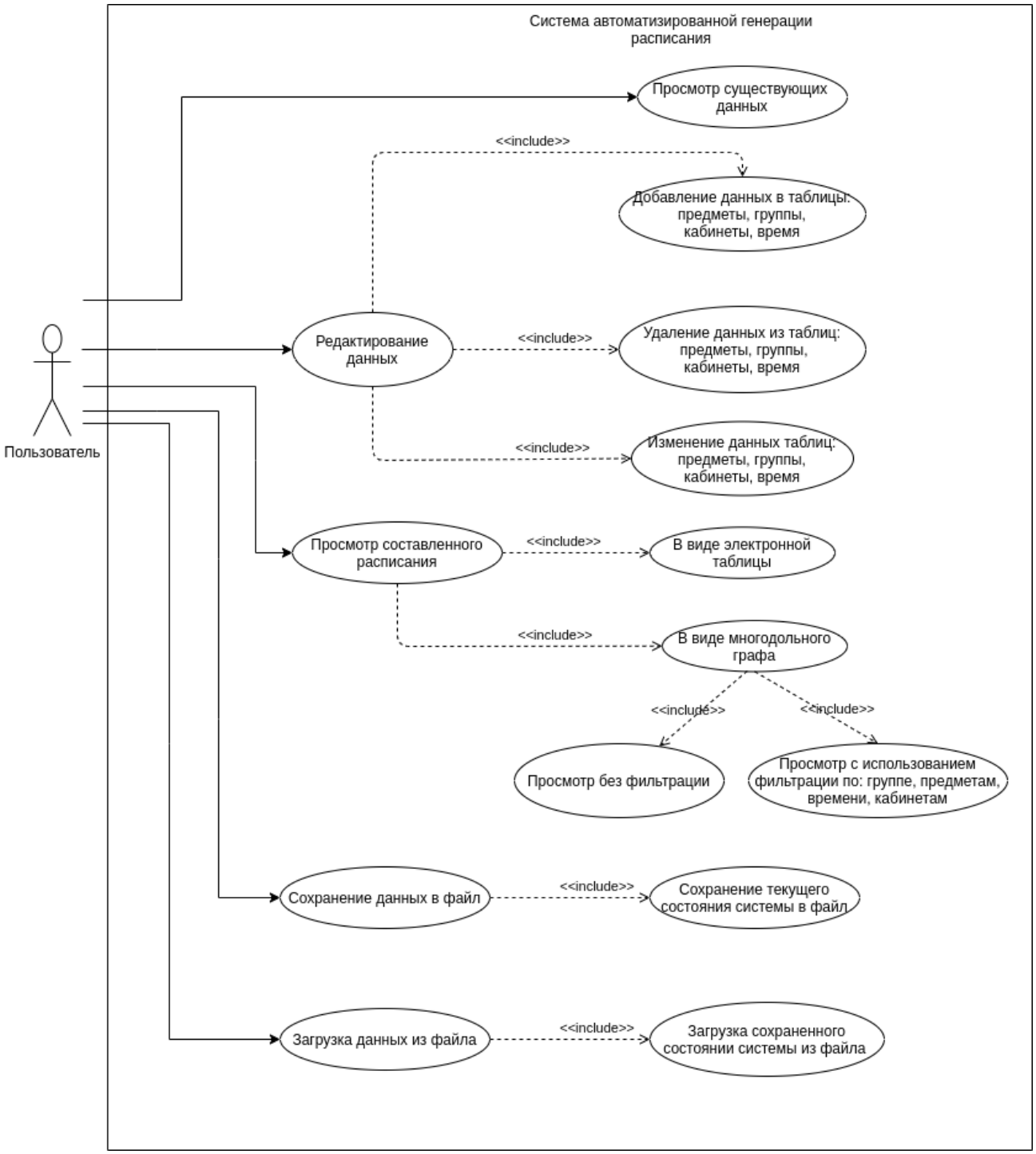


Рисунок 1 - Диаграмма вариантов использования

1.4. График разработки

Этот план будет обновляться по ходу разработки продукта. Целевые даты окончания каждого этапа показаны ниже (см. табл. 7).

Таблица 7 - Даты окончания каждого этапа

Название этапа	***Дата окончания***
Начало (Inception)	03.05.2020
Уточнение (Elaboration)	15.05.2020
Построение (Construction)	23.05.2020
Внедрение (Transition)	31.05.2020

Распределение ролей представлено в таблице 8.

Таблица 8 - Распределение ролей

Роль	**Фамилия**
Technical writer	Допира Валерия
Implementer	Бочкарев Иван
Design Reviewer	Бочкарев Иван
System Analyst	Павлов Данила
Deployment Manager	Павлов Данила
Configuration Manager	Бочкарев Иван
Implementer	Ильянов Вячеслав
Designer	Павлов Данила
Test Analyst	Бочкарев Иван
Tool Specialist	Допира Валерия
Tester	Ильянов Вячеслав
Integrator	Допира Валерия
Design Reviewer	Ильянов Вячеслав
System Analyst	Ильянов Вячеслав
Deployment Manager	Допира Валерия
Configuration Manager	Павлов Данила
Developer of requirements	Допира Валерия
Designer	Допира Валерия
Developer of architecture	Допира Валерия
Implementer	Павлов Данила
Integrator	Ильянов Вячеслав

Tester	Бочкарев Иван
Tester	Допира Валерия
Developer of requirements	Павлов Данила

Аналитики отвечают за обеспечение пригодности требований к тестированию и за ясность формулировок требований к выполняемым тестам. Разработчики должны помнить о тестировании при разработке приложений и нести ответственность за тестирование собственного кода.

Руководители должны обеспечить наличие планов тестирования и ресурсов, необходимых для формирования среды тестирования и выполнения необходимых тестов.

Испытатели - это эксперты по качеству. Они отвечают за все тестирование продукта (включая функциональное тестирование, тестирование системы и тестирование производительности) и должны лучше всех понимать, что такое качество и как его достичь.

Распределение ролей и обязанностей исполнителям представлено ниже (см. табл. 9).

Таблица 9 – Роли и обязанности

ФИО	***Роль***	***Обязанность***
Павлов Данила, Ильянов Вячеслав	Аналитик	Устанавливает требования к разрабатываемой системе путем определения необходимой функциональности
Допира Валерия, Павлов Данила	Разработчик требований	Формирует детальные требования к разрабатываемой системе
Допира Валерия, Павлов Данила	Проектировщик, Дизайнер	Проектирование компонентов системы с учётом требований, заданных проектом
Допира Валерия	Разработчик архитектуры	Управление разработкой программной архитектуры системы, что включает продвижение и поддержку ключевых технических решений, задающих рамки реализации проекта
Бочкарев Иван, Вячеслав Ильянов, Павлов Данила	Реализатор	Разработка программных компонентов и проверка их

		функциональности после интеграции в подсистемы
Бочкарев Иван, Ильянов Вячеслав	Рецензент дизайна	Защита качества дизайна, проверка удобства пользователя
Ильянов Вячеслав, Допира Валерия	Интегратор	Специализация на объединении компонентных подсистем в единое целое и обеспечении совместной работы этих подсистем
Допира Валерия, Бочкарев Иван, Ильянов Вячеслав	Разработчик тестов	Выполнение тестирования продукта и описание исхода тестирования. Определение общей стратегии тестирования и контроль успешности ее реализации
Бочкарев Иван	Аналитик тестирования	Защита качества тестирования, планирование ресурсов и управление ими, а также решение проблем, препятствующих процессу тестирования
Допира Валерия	Технический писатель	Составление отчетов, описание разработанного продукта
Павлов Данила, Допира Валерия	Менеджер по развертыванию	Организация одновременного централизованного развертывания продукта на компьютерах, повышение рентабельности обслуживания клиентов и повышение прибыли предприятия
Бочкарев Иван, Павлов Данила	Менеджер конфигураций	Управление наборами рабочих продуктов и их версиями
Допира Валерия	Руководитель проекта	Планирование затрат ресурсов, распределение ресурсов, выделение приоритетов, направление усилий коллектива
Спицын Александр Валентинович	Проверяющий	Оценивание планирования проекта и ценности рабочих продуктов на границах важнейших этапов жизненного цикла проекта

Разработка системы будет производиться с использованием поэтапного подхода.

Фазы и относительная временная шкала показаны ниже (см. табл. 10)

Таблица 10 – Фазы и относительная временная шкала

Фаза	***Дата начала***	***Дата окончания***
Начало	1 неделя	6 неделя
Уточнение	7 неделя	10 неделя

Построение	11 неделя	12 неделя
Внедрение	13 неделя	14 неделя

Рабочие продукты получаемые после завершения определенного этапа представлены ниже (см. табл. 11)

Таблица 11 – Рабочие продукты каждого этапа

Фаза	***Описание***	***Рабочие продукты***
Начало	Определение содержания проекта, разработка основных сценариев	План разработки ПО Видение Глоссарий Модель вариантов использования Инфраструктура разработки
Уточнение	Создание стабильной формальной архитектуры для выполнения разработки ПО	План разработки ПО Видение Глоссарий Модель вариантов использования Инфраструктура разработки Процесс разработки Диаграмма вариантов использования, Диаграмма классов, Диаграмма пакетов, Диаграмма компонентов, Диаграмма последовательностей, Диаграмма состояний Тестовый набор
Построение	Завершение разработки системы в соответствии с базовой архитектурой	План итерации Разработка системы Тестирование системы Материалы поддержки пользователей План развертывания
Внедрение	Обеспечение готовности программного обеспечения к представлению пользователям	План итерации Демонстрация работы системы Компоновка продукта Материалы поддержки пользователей Элементы реализации

Для иллюстрации плана, графика работ и занятости членов команды, работающих над проектом, удобно использовать диаграмму Ганта. Диаграмма Ганта со списком рабочих продуктов и исполнителями представлена ниже (см. рис. 2).

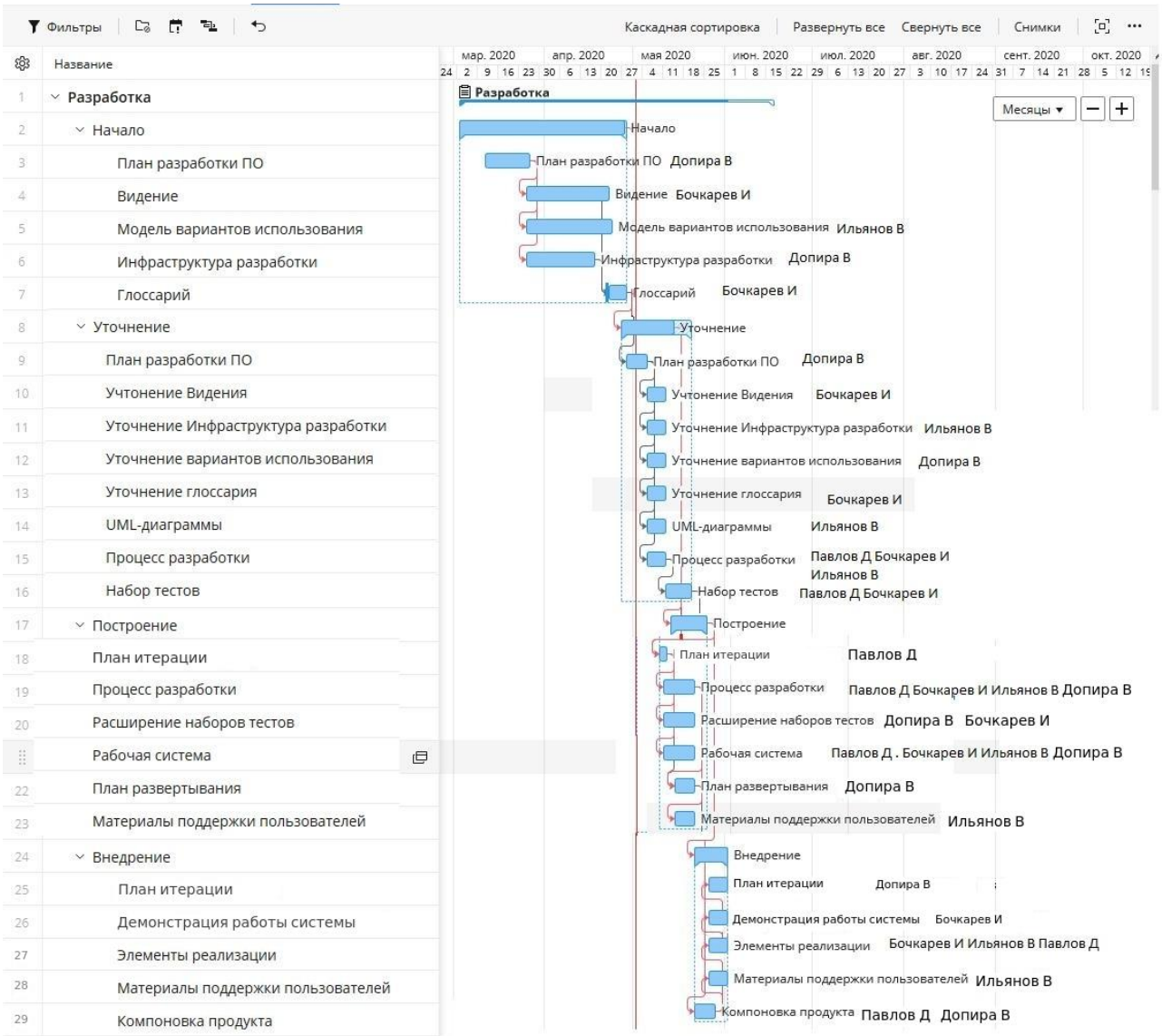


Рисунок 2 - Диаграмма Ганта

График сдачи каждой задачи проекта представлен ниже (см. табл. 12).

Таблица 12 – График проекта

Фаза	***Задача***	***Окончание работы***	***Исполнители***
Начало	План разработки ПО	26.03.2020	Допира Валерия
	Видение	22.04.2020	Бочкарев Иван
	Модель вариантов использования	22.04.2020	Павлов Данила
	Инфраструктура разработки	22.04.2020	Бочкарев Иван
	Глоссарий	27.04.2020	Ильянов Вячеслав
Уточнение	План разработки ПО	30.04.2020	Допира Валерия
	Видение	04.05.2020	Бочкарев Иван

	Модель вариантов использования	04.05.2020	Павлов Данила
	Инфраструктура разработки	04.05.2020	Бочкарев Иван
	Глоссарий	04.05.2020	Ильянов Вячеслав
	UML диаграммы	04.05.2020	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
	Набор тестов	14.05.2020	Допира Валерия, Бочкарев Иван
Построение	План итерации	22.05.2020	Павлов Данила
	Разработка системы	03.06.2020	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
	UML диаграммы	15.05.2020	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
Внедрение	План итерации	31.05.2020	Допира Валерия
	Демонстрация работы системы	31.05.2020	Бочкарев Иван
	Элементы реализации	31.05.2020	Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
	Компоновка продукта	31.05.2020	Павлов Данила, Допира Валерия
	Материалы поддержки пользователей	29.05.2020	Ильянов Вячеслав

Нагрузка исполнителей

Количество дней выделенных на каждый ресурс представлено ниже (см. табл. 13).

Таблица 13 – Загрузка ресурсов по дням

Ресурсы	***Количество дней***
План разработки ПО	14
Видение	35

Модель вариантов использования	35
Инфраструктура разработки	12
Глоссарий	6
UML диаграммы	6
Набор тестов	3
Материалы поддержки пользователей	7
Разработка продукта	25

Количество часов в неделю, которые исполнители тратят на проект описано ниже (см. табл. 14, 15, 16, 17). Нагрузка распределена равномерно на исполнителей на протяжении всего проекта. Роли распределены в соответствии с планом разработки программного обеспечения. Исполнитель может распределять время в течение недели, как ему удобно: работая каждый день по несколько часов или полноценный рабочий день. Синхронизация работы проходит раз в неделю или чаще, если это необходимо.

- Допира Валерия

Таблица 4 – График исполнителя Валерии

Фаза	***Задача***	***Окончание работы***	***Затраченное время***	***Исполнители***
Начало	План разработки ПО	26.03.2020	2 дня	Допира Валерия
Уточнение	План разработки ПО	30.04.2020	2 дня	Допира Валерия
	UML диаграммы	04.05.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
	Набор тестов	14.05.2020	3 дня	Допира Валерия, Бочкарев Иван
Построение	Разработка системы	03.06.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
	UML диаграммы	15.05.2020	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
Внедрение	План итерации	31.05.2020	1 день	Допира Валерия
	Компоновка продукта	31.05.2020	3 дня	Павлов Данила, Допира Валерия

- Павлов Данила

Таблица 15 – График исполнителя Данила

Фаза	***Задача***	***Окончание работы***	***Затраченное время***	***Исполнители***
Начало	Модель вариантов использования	22.04.2020	2 дня	Павлов Данила
Уточнение	Модель вариантов использования	04.05.2020	2 дня	Павлов Данила
	UML диаграммы	04.05.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
Построение	План итерации	22.05.2020	1 день	Павлов Данила
	Разработка системы	03.06.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
	UML диаграммы	15.05.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
Внедрение	Элементы реализации	31.05.2020	2 дня	Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
	Компоновка продукта	31.05.2020	3 дня	Павлов Данила, Допира Валерия

- Бочкарев Иван

Таблица 16 – График исполнителя Ивана

Фаза	***Задача***	***Окончание работы***	***Затраченное время***	***Исполнители***
Начало	Видение	22.04.2020	1 день	Бочкарев Иван

	Инфраструктура разработки	22.04.2020	1 день	Бочкарев Иван
Уточнение	Видение	04.05.2020	1 день	Бочкарев Иван
	Инфраструктура разработки	04.05.2020	1 день	Бочкарев Иван
	UML диаграммы	04.05.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
	Набор тестов	14.05.2020	3 дня	Допира Валерия, Бочкарев Иван
Построение	Разработка системы	03.06.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
	UML диаграммы	15.05.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
	Тестирование системы	22.05.2020	2 дня	Допира Валерия, Бочкарев Иван
Внедрение	Демонстрация работы системы	31.05.2020	1 день	Бочкарев Иван
	Элементы реализации	31.05.2020	2 дня	Бочкарев Иван, Павлов Данила, Ильянов Вячеслав

- Ильянов Вячеслав

Таблица 17 – График исполнителя Вячеслава

Фаза	***Задача***	***Окончание работы***	***Затраченное время***	***Исполнители***
Начало	Глоссарий	27.04.2020	2 дня	Ильянов Вячеслав
Уточнение	Глоссарий	04.05.2020	1 день	Ильянов Вячеслав
	UML диаграммы	04.05.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
Построение	Разработка системы	03.06.2020	7 дней	Допира Валерия, Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
	UML диаграммы	15.05.2020	7 дней	Допира Валерия, Бочкарев Иван,

				Павлов Данила, Ильянов Вячеслав
	Материалы поддержки пользователей	22.05.2020	2 дня	Ильянов Вячеслав
Внедрение	Элементы реализации	31.05.2020	2 дня	Бочкарев Иван, Павлов Данила, Ильянов Вячеслав
	Материалы поддержки пользователей	29.05.2020	1 день	Ильянов Вячеслав

2. Разработка программного обеспечения

2.1. Инфраструктура разработки

Инфраструктура разработки – этот рабочий документ описывает аппаратное и программное обеспечение, такое как компьютеры и операционные системы, на которых реализованы различные инструменты. Стандартная инфраструктура разработки существует для того, чтобы определить условия для разработки.

В качестве инструмента моделирования был выбран веб сервис Draw.io.

Разработка ведется на операционных системах Windows и Linux. Целевой платформой также являются данные операционные системы.

Разработка системы ведется на объектно-ориентированном языке C++.

Для более эффективной разработки система написано с использованием Фреймворка Qt. Данный Фреймворк имеет большой набор виджете, который облегчает разработку программного интерфейса.

В качестве среды разработки была взята IDE Qt Creator 5.13. Это кроссплатформенная свободная среда для разработки на C++, включающая в себя графический интерфейс отладчика и визуальные средства разработки интерфейса.

Был выбран доступный веб-инструмент GitHub, предоставляющий систему управления репозиториями кода, системой отслеживания ошибок, отслеживания заданий, системой уведомлений по группам, ссылки и доступ к конференциям из задач.

Большинство инструментов разработки программного обеспечения поддерживают совместную работу пользователей с возможностью одновременного обращения к информации из общего хранилища, такие как Github, draw.io.

Ряд инструментов: компиляторы, отладчики, редакторы и графические средства можно просто установить на компьютеры. Как было сделано с инструментов «Qt». Единственное что потребуется это активировать их. Для этого потребуется

создать учетную запись в соответствующем портале и получить ключ активации или бесплатную версию инструмента.

Большинство инструментов можно не конфигурировать, так как начальные настройки уже достаточны для нормальной работы.

2.2 Шаблоны проектирования

При разработке использовались следующие шаблоны проектирования.

Аллокатор является обёрткой, то есть, все экземпляры данного аллокатора являются взаимозаменяемыми, сравнительно равными и могут освободить память, выделенную любым другим экземпляром того же типа аллокатора.

Итератор относится к поведенческому типу. Представляет способ последовательного доступа к элементам множества, независимо от его внутреннего устройства.

Диаграммы представлены на рисунках 3, 4.

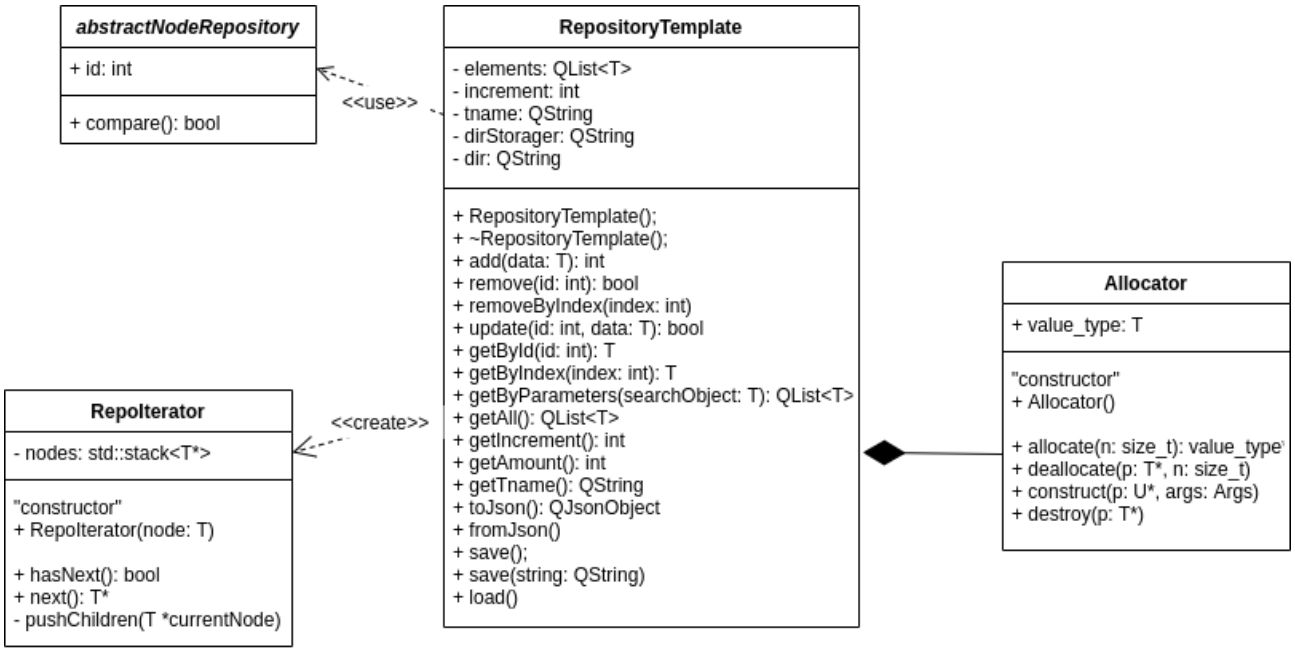


Рисунок 3 - Аллокатор и итератор

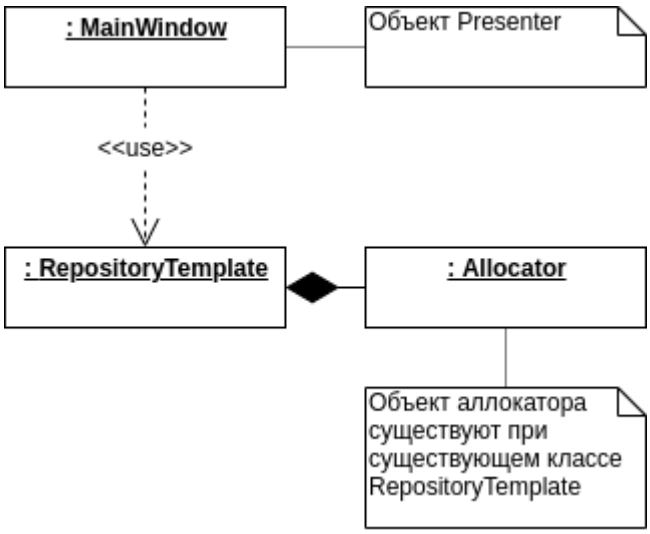


Рисунок 4 - Диаграмма объектов

Код для аллокатора:

```
template <class T>
class Allocator {
    typedef T value_type;
    Allocator() noexcept {}
    template <class U> Allocator (const Allocator<U>&) noexcept {
        std::cout << "Allocator<T>::Allocator(const Allocator<U>&):
" << std::endl;
    }
    T* allocate (std::size_t n) {
        std::cout << "Allocating "  << std::endl;
        return reinterpret_cast<T*>( ::operator new(n*sizeof(T)));
    }
    void deallocate (T* p, std::size_t n) {
        std::cout << "Deallocating "  << std::endl;
        ::operator delete(p);
    }
    template<typename U, typename... Args>
     void construct(U* p, Args&&... args) {
       std::cout << "Constructing " << std::endl;
       new (p) U(std::forward<Args>(args)...);
     }
     void destroy(T* p) {
       std::cout << "Destroying " << std::endl;
       p->~T();
     }
};
```

Код для итератора:

```
template<class T>
class RepoIterator
{
private:
    std::stack<T*> nodes;
```

```
    void pushChildren(T *currentNode);
public:
    RepoIterator(T *node) {
        pushChildren(node);
    }
    bool hasNext();
    T* next();
};
template<class T>
void RepoIterator<T>::pushChildren(T *currentNode)
{
    if (currentNode == nullptr) return;
    nodes.push(currentNode);
    for(auto     iter     =     currentNode->getChildren()->begin();
iter!=currentNode->getChildren()->end(); ++iter)
    {
        pushChildren(*iter);
    }
}
template<class T>
bool RepoIterator<T>::hasNext()
{
    return !nodes.empty();
}
template<class T>
T* RepoIterator<T>::next()
{
    if (!hasNext())
    {
        throw  stderr(this,  new  std::string("RepoIterator.h"),  46,
new std::string("next()"));
    }
    T* res = nodes.top();
    nodes.pop();
    return res->getValue();
}
```

Прототип - это порождающий шаблон. Определяет несколько видов объектов, чтобы при создании использовать объект-прототип и создает новые объекты, копируя прототип. Класс abstractNodeRepository является прототипом (см. рис. 5).

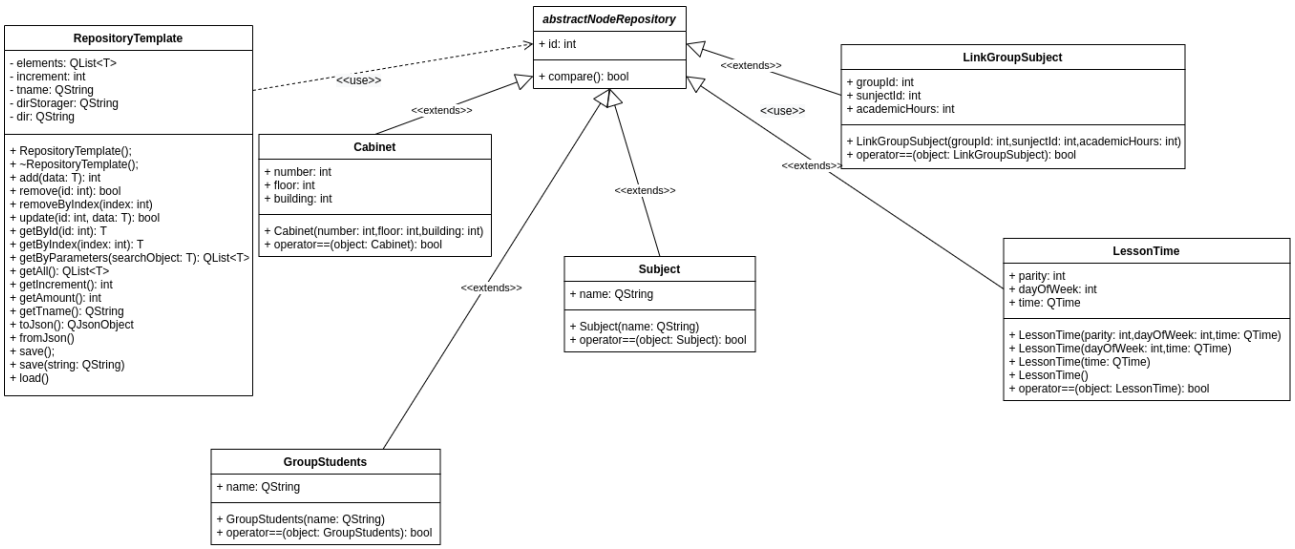


Рисунок 5 - Прототип

Фасад - структурный шаблон. Предоставляет единый интерфейс к группе интерфейсов подсистемы. Определяет высокоуровневый интерфейс, делая подсистему проще для использования.

Все классы диалогов используют класс QDialog за основу и изменены только данные, которые выводятся в диалоговое окно. QDialog - класс фасад (см. рис. 6).

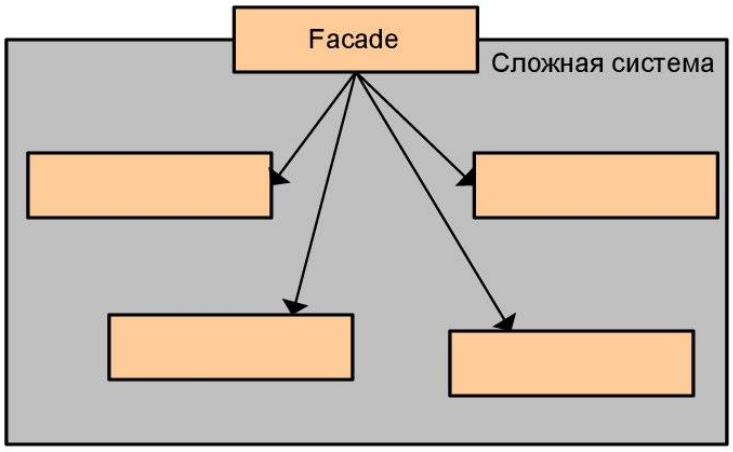


Рисунок 6 - Фасад

Шаблонный метод - это поведенческий шаблон. Определяет алгоритм, некоторые этапы которого делегируются подклассам. Позволяет подклассам переопределить эти этапы, не меняя структуру алгоритма.

Инстанцирование шаблона (см. рис. 7)

Для удобной работы работы с данными внутри приложения был спроектирован шаблонный классы RepositoryTemplate<T>. В основу его функциональности входит:

1. CRUD операции
2. Сохранение и чтение с диска
3. Упорядочивание данных
4. Сериализация и десериализация данных
5. Поиск по данным

Данный шаблонный класс используется для его инстанцирование в следующие специализации шаблона:

1. RepositoryTemplate<Cabinet>
2. RepositoryTemplate<GroupStudents>
3. RepositoryTemplate<LessonTime>
4. RepositoryTemplate<Subject>
5. RepositoryTemplate<LinkGroupSubject>

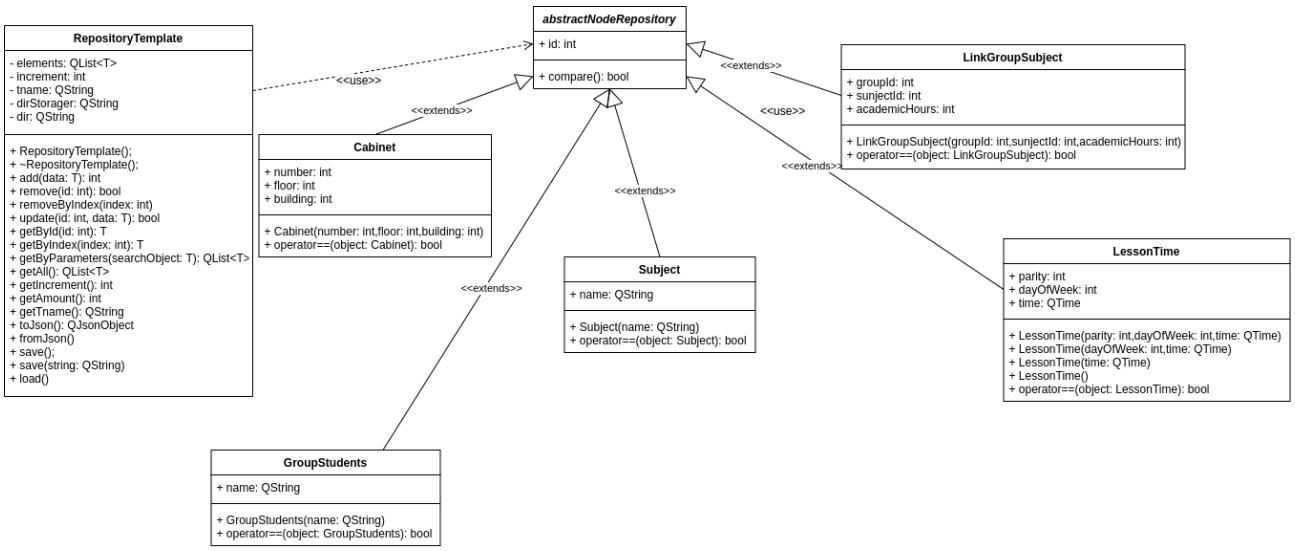


Рисунок 7 - Шаблонный метод

2.3 Результаты разработки

2.3.1 Сценарий использования программы

На диаграмме 7 представлен сценарий использования программы. Все сообщения асинхронные и отображаются сплошной линией с открытой

стрелкой, так как в программе поддерживается шаблон проектировании Модель-Вид-Контроллер.

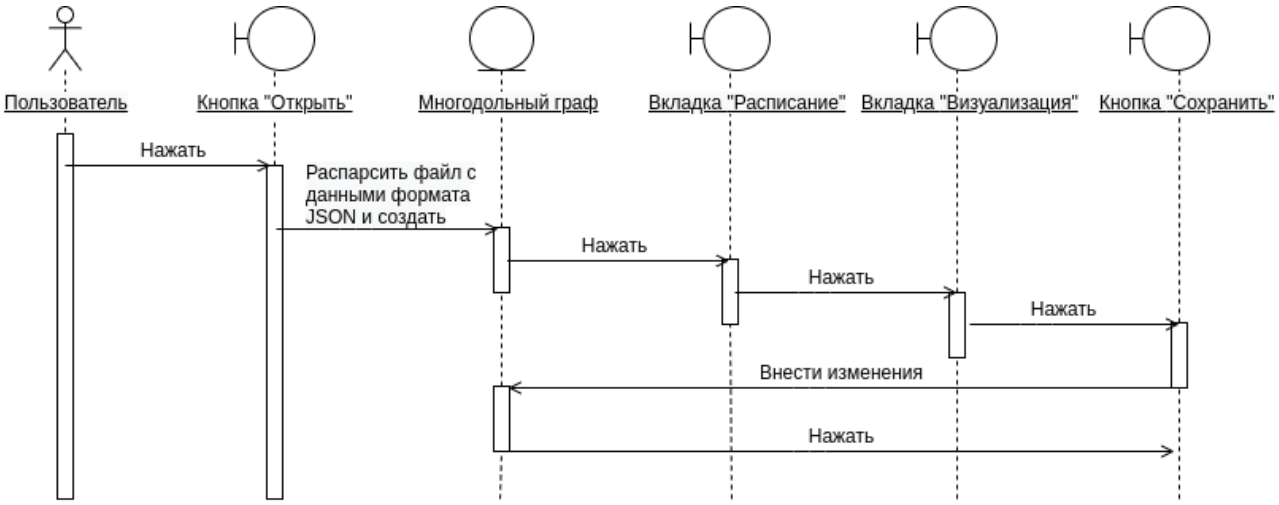


Рисунок 8 - Диаграмма последовательностей

2.3.2 Логическое представление

Для представления архитектуры используется диаграмма компонентов. Диаграмма компонентов показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п. Диаграмма компонентов представлена на рисунке 9 и в приложении А.

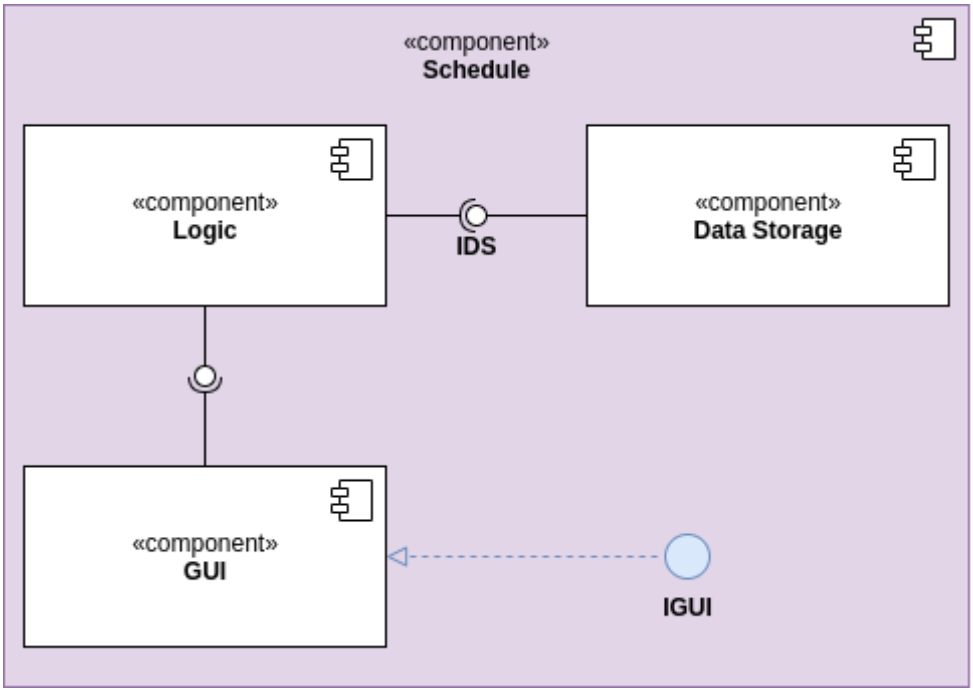


Рис. 9 Диаграмма компонентов

2.3.3 Построение расписания

Контейнер — многодольный граф. Граф задается используя списки смежности ребер и списки ребер:

```
    // Списки смежности
     QMap<int, QList<QPair<int, float>>> subjects;
     QMap<int, QList<QPair<int, float>>> groupsStudents;
     QMap<int, QList<QPair<int, float>>> cabinets;
     QMap<int, QList<QPair<int, float>>> times;
     // Список всех ребер
     QList<QPair<int, int>> groups_subjects;
     QList<QPair<int, int>> subjects_cabinets;
     QList<QPair<int, int>> cabinets_times;
```

Функция fit выделяет список индексов:

```
QList<QList<int>> Graph::fit()
{
     RepositoryTemplate<Cabinet> cabinets = this->repoCabinets;
     RepositoryTemplate<LessonTime> times = this->repoLessonTime;
     QMap<int, QList<QPair<int, int>>> ways;
     QList<QPair<int, int>> used_edges_local;
     QList<int> used_times;
     QList<QPair<int, int>> edges;
     QPair<int, int> edge;
     QList<QPair<int, float>> cabinets_w;
     QList<QPair<int, float>> times_w;
     QPair<int, float> pair;
     int k;
     int r;
     for (Cabinet cab : cabinets.getAll()) {
         edge.first = cab.id;
         for (LessonTime time : times.getAll()) {
             edge.second = time.id;
             edges.append(edge);
         }
     }
     for (GroupStudents group : this->repoGroupStudents) {
         cabinets_w.clear();
         times_w.clear();
         used_times.clear();
         used_edges_local.clear();
         auto            links_subjects            =            this-
>repoLinkGroupSubject.getByParameters(LinkGroupSubject(group.id,-
1,-1));
         srand(group.id+time(0));
         r = rand() % links_subjects.size();
         LinkGroupSubject start_link = links_subjects[r];
         r = rand() % cabinets.getAmount();
         Cabinet start_cabinet = cabinets.getByIndex(r);
         r = rand() % times.getAmount();
```

```
        LessonTime start_time = times.getByIndex(r);
        qSort(edges.begin(),  edges.end(),  [this,  start_cabinet,
start_time](QPair<int, int>& a, QPair<int, int>& b) {
            Cabinet cab1 = this->repoCabinets.getById(a.first);
            Cabinet cab2 = this->repoCabinets.getById(b.first);
            LessonTime           time1           =           this-
>repoLessonTime.getById(a.second);
            LessonTime           time2           =           this-
>repoLessonTime.getById(b.second);
            float c1 = sqrt(pow(cab1.number - start_cabinet.number,
2) + pow(cab1.floor - start_cabinet.floor, 2) + pow(cab1.building -
start_cabinet.building, 2));
            float c2 = sqrt(pow(cab2.number - start_cabinet.number,
2) + pow(cab2.floor - start_cabinet.floor, 2) + pow(cab2.building -
start_cabinet.building, 2));
            float     t1     =     sqrt(pow(time1.dayOfWeek     -
start_time.dayOfWeek,2) + pow(time1.parity - start_time.parity,2) +
pow((start_time.time.msecsSinceStartOfDay()/8640000+start_time.day
OfWeek*100)                                                    -
(time1.time.msecsSinceStartOfDay()/8640000+time1.dayOfWeek*100),2)
);
            float     t2     =     sqrt(pow(time2.dayOfWeek     -
start_time.dayOfWeek,2) + pow(time2.parity - start_time.parity,2) +
pow((start_time.time.msecsSinceStartOfDay()/8640000+start_time.day
OfWeek*100)                                                    -
(time2.time.msecsSinceStartOfDay()/8640000+time2.dayOfWeek*100),2)
);
            return c1 + t1 < c2 + t2;
        });
        for (LinkGroupSubject link : links_subjects) {
            for (int i = 0, amount = link.academicHours; i < amount;
++i) {
                for (k = 0; used_times.count(edges[k].second) != 0;
++k);
                pair = edges[k];
                edges.removeAt(k);
                used_times.append(pair.second);
                used_edges_local.append(pair);
            }
        }
        ways.insert(group.id, used_edges_local);
    }
    QList<QList<int>> result;
    QList<int>way;
    for (LinkGroupSubject link : this->repoLinkGroupSubject) {
       for (int i = 0; i < link.academicHours; ++i) {
            way.clear();
            pair = ways[link.groupId].front();
            ways[link.groupId].pop_front();
            way.append(link.groupId);
            way.append(link.subjectId);
```

```
                way.append(pair.first);
                way.append(pair.second);
                result.append(way);
                ++k;
            }
        }
        return result;
}
```

Класс граф , отвечающий за построение графа, имеет следующие связи с другими классами отображен на рисунке 10.

Агрегация – это когда экземпляр одно класса создается где-то в другом месте кода, и передается в конструктор другого класса в качестве параметра.

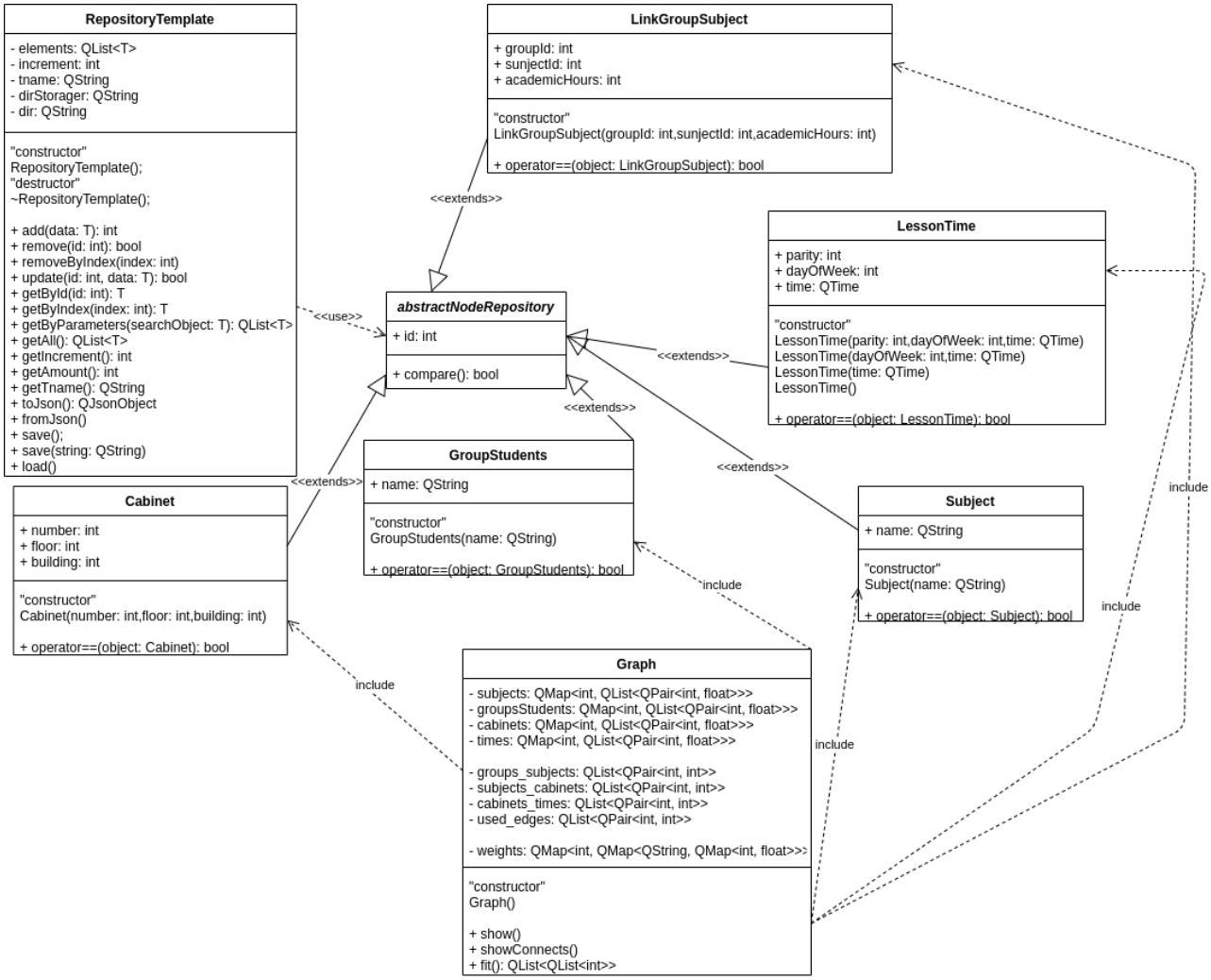


Рисунок 10 - Диаграмма графа

2.3.4 Алгоритм генерации расписания

Алгоритм построения расписания описан ниже. Он подразумевает, что используется четырех дольный ориентированный граф.

- **Получение всех необходимых данных**. Данные берутся из репозиторий «кабинетов», «предметов», «групп», «времени» и «связей группы с предметами».

```
Graph::Graph(
    RepositoryTemplate<Cabinet> repoCabinets,
    RepositoryTemplate<GroupStudents> repoGroupStudents,
    RepositoryTemplate<LessonTime> repoLessonTime,
    RepositoryTemplate<Subject> repoSubjects,
    RepositoryTemplate<LinkGroupSubject> repoLinkGroupSubject
)
{
    this->repoCabinets = repoCabinets;
    this->repoGroupStudents = repoGroupStudents;
    this->repoLessonTime = repoLessonTime;
    this->repoSubjects = repoSubjects;
    this->repoLinkGroupSubject = repoLinkGroupSubject;
    //this->initGraph();
}
```

- **Инициализация графа**. В первую очередь создаются доли графа на основе репозиторев «предметов», «групп», «кабинетов» и «времени». На основе репозитория «связей группы с предметами» выстраиваются ребра между предметами и группами, при чем количество ребер между группой и предметом определяется количеством академических часов, выделанных группе на предмет. Все вершины доли «предметы» соединяются со всеми вершинами доли «кабинеты», а те в свою очередь со всеми вершинами доли «время».

```
void Graph::initGraph()
{
    for (int i=0; i < this->repoLinkGroupSubject.getAmount(); ++i)
{
        LinkGroupSubject linkGroupSubject = this-
>repoLinkGroupSubject.getByIndex(i);
        QPair<int, int> item;
        item.first = linkGroupSubject.groupId;
        item.second = linkGroupSubject.subjectId;
        this->groups_subjects.append(item);
    }
    for (int i=0; i < this->repoSubjects.getAmount(); ++i) {
        Subject subject = this->repoSubjects.getByIndex(i);
        QList<QPair<int, float>> tmp = {};
        this->subjects.insert(subject.id, tmp);
    }
    for (int i=0; i < this->repoCabinets.getAmount(); ++i) {
        Cabinet cabinet = this->repoCabinets.getByIndex(i);
        QList<QPair<int, float>> tmp = {};
```

```
            this->cabinets.insert(cabinet.id, tmp);
        }
        for (int i=0; i < this->repoLessonTime.getAmount(); ++i) {
            LessonTime times = this->repoLessonTime.getByIndex(i);
            QList<QPair<int, float>> tmp = {};
            this->times.insert(times.id, tmp);
        }
}
```

- **Первичная генерация графа**. Для каждой вершины из доли «группы» случайным образом определяется один путь от «группы» до «времени», через «предмет» и «кабинет». Данный путь является корневым для «группы».

```
void Graph::firstStep()
{
    // Проходимся по всем группам учащихся
    QList<int> keys = this->groupsStudents.keys();
    for (int key : keys) {
        srand(time(0));
        // Получаем объект группы студентов
        QList<QPair<int,    float>>    groupStudents    =    *this-
>groupsStudents.find(key);
        // Выбираем случайный предмет
        int r = rand() % groupStudents.size();
        QPair<int, float> subject = groupStudents[r];
        // Выбираем случайный кабинет
        r = rand() % this->repoCabinets.getAmount();
        auto cabinet = this->repoCabinets.getByIndex(r);
        // Закрепляем за предметом кабинет
        QPair<int, float> item;
        item.first = cabinet.id;
        item.second = 0;
        QList<QPair<int, float>> tmp = { item };
        this->subjects.remove(subject.first);
        this->subjects.insert(subject.first, tmp);
        // Чтобы не было повторений
        while (true) {
            srand(time(0));
            int r = rand() % this->repoLessonTime.getAmount();
            auto time = this->repoLessonTime.getByIndex(r);
            auto cabtime = this->cabinets[cabinet.id];
            QPair<int, float> item;
            item.first = time.id;
            item.second = 0;
            if(cabtime.count(item) == 0) {
                tmp = { item };
                this->cabinets.remove(cabinet.id);
                this->cabinets.insert(cabinet.id, tmp);
                break;
            }
        };
```

```
    }
}
```

- **Инициализация весов**. Основываясь на корневом пути для каждой группы инициализируются веса ребер, таким образом, что веса от доли «группа» к доле «предметы» равны нулю, веса от доли «предметы» к доле «кабинеты» определяются по формуле:

$$\sqrt{\left(корпус_i - корпу_{корневой}\,c\right)^2 + \left(этаж_i - этаж_{корневой}\right)^2 + \left(кабинет_i - кабинет_{корневой}\right)^2} = вес$$

Данная формула помогает рассчитать подобие географического расстояния между аудиториями.

Веса для ребер между кабинетом и временем рассчитываются похожим способом, за тем лишь исключением, время предварительно нормируется:

$$\sqrt{\left(день_{неделиi} - день_{неделикорневой}\,c\right)^2 + \left(четность_i - четность_{корневой}\right)^2 + \left(время_i - время_{корневой}\right)^2} = вес$$

```
QList<QList<int>> Graph::fit()
{
    RepositoryTemplate<Cabinet> cabinets = this->repoCabinets;
    RepositoryTemplate<LessonTime> times = this->repoLessonTime;
    QMap<int, QList<QPair<int, int>>> ways;
    QList<QPair<int, int>> used_edges_local;
    QList<int> used_times;
    QList<QPair<int, int>> edges;
    QPair<int, int> edge;
    QList<QPair<int, float>> cabinets_w;
    QList<QPair<int, float>> times_w;
    QPair<int, float> pair;
    int k;
    int r;
    for (Cabinet cab : cabinets.getAll()) {
        edge.first = cab.id;
        for (LessonTime time : times.getAll()) {
            edge.second = time.id;
            edges.append(edge);
        }
    }
    for (GroupStudents group : this->repoGroupStudents) {
        cabinets_w.clear();
```

```
        times_w.clear();
        used_times.clear();
        used_edges_local.clear();
        auto            links_subjects              =            this-
>repoLinkGroupSubject.getByParameters(LinkGroupSubject(group.id,-
1,-1));
        srand(group.id+time(0));
        r = rand() % links_subjects.size();
        LinkGroupSubject start_link = links_subjects[r];
        r = rand() % cabinets.getAmount();
        Cabinet start_cabinet = cabinets.getByIndex(r);
        r = rand() % times.getAmount();
        LessonTime start_time = times.getByIndex(r);
        qSort(edges.begin(),  edges.end(),  [this,  start_cabinet,
start_time](QPair<int, int>& a, QPair<int, int>& b) {
            Cabinet cab1 = this->repoCabinets.getById(a.first);
            Cabinet cab2 = this->repoCabinets.getById(b.first);
            LessonTime            time1            =            this-
>repoLessonTime.getById(a.second);
            LessonTime            time2            =            this-
>repoLessonTime.getById(b.second);
            float c1 = sqrt(pow(cab1.number - start_cabinet.number,
2) + pow(cab1.floor - start_cabinet.floor, 2) + pow(cab1.building -
start_cabinet.building, 2));
            float c2 = sqrt(pow(cab2.number - start_cabinet.number,
2) + pow(cab2.floor - start_cabinet.floor, 2) + pow(cab2.building -
start_cabinet.building, 2));
            float     t1     =     sqrt(pow(time1.dayOfWeek     -
start_time.dayOfWeek,2) + pow(time1.parity - start_time.parity,2) +
pow((start_time.time.msecsSinceStartOfDay()/8640000+start_time.day
OfWeek*100)                                                    -
(time1.time.msecsSinceStartOfDay()/8640000+time1.dayOfWeek*100),2)
);
            float     t2     =     sqrt(pow(time2.dayOfWeek     -
start_time.dayOfWeek,2) + pow(time2.parity - start_time.parity,2) +
pow((start_time.time.msecsSinceStartOfDay()/8640000+start_time.day
OfWeek*100)                                                    -
(time2.time.msecsSinceStartOfDay()/8640000+time2.dayOfWeek*100),2)
);
            return c1 + t1 < c2 + t2;
        });
        for (LinkGroupSubject link : links_subjects) {
            for (int i = 0, amount = link.academicHours; i < amount;
++i) {
                for (k = 0; used_times.count(edges[k].second) != 0;
++k);
                pair = edges[k];
                edges.removeAt(k);
                used_times.append(pair.second);
                used_edges_local.append(pair);
            }
```

```
            }
            ways.insert(group.id, used_edges_local);
        }
        QList<QList<int>> result;
        QList<int>way;
        for (LinkGroupSubject link : this->repoLinkGroupSubject) {
           for (int i = 0; i < link.academicHours; ++i) {
                way.clear();
                pair = ways[link.groupId].front();
                ways[link.groupId].pop_front();
                way.append(link.groupId);
                way.append(link.subjectId);
                way.append(pair.first);
                way.append(pair.second);
                result.append(way);
                ++k;
            }
        }
        return result;
}
```

- **Поиск минимального пути (генерация расписания).** Все ребра сортируются по возрастанию весов. После чего запускается цикл. На каждой итерации выбирается ребро между двумя соединенными долями с минимальным весом и помечается как использованное. Таким образом выбранные ребра становятся ребрами нового пути.

```
void Graph::showConnects() {
    QMapIterator<int,     QList<QPair<int,     float>>>     it(this-
>groupsStudents);
    while (it.hasNext()) {
        it.next();
        auto subjects = it.value();
        for (auto subject : subjects) {
            auto cabinets = this->subjects[subject.first];
            for (auto cabinet : cabinets) {
                auto times = this->cabinets[cabinet.first];
                for (auto time : times) {
                    GroupStudents         g         =         this-
>repoGroupStudents.getById(it.key());
                    Subject           s           =           this-
>repoSubjects.getById(subject.first);
                    Cabinet           c           =           this-
>repoCabinets.getById(cabinet.first);
                    LessonTime          t          =          this-
>repoLessonTime.getById(time.first);
                    qDebug()  <<  g.toString().toLocal8Bit().data()
<<  "  ->  "  <<  s.toString().toLocal8Bit().data()  <<  "  ->  "  <<
```

```
c.toString().toLocal8Bit().data()    <<    "    ->    "         <<
t.toString().toLocal8Bit().data() << endl;
                    }
                };
            }
        }
}
```

- **Постобработка данных.** Перед выдачей результата формируется итоговый список ребер, который и передается дальше.

```
void Graph::show()
{
    for (int i=0; i < this->repoCabinets.getAmount(); ++i) {
        Cabinet cabinet = this->repoCabinets.getByIndex(i);
        qDebug() << cabinet.toString() << endl;
    }
    qDebug() << this->repoCabinets.getAmount() << endl;
    for (int i=0; i < this->repoSubjects.getAmount(); ++i) {
        Subject subject = this->repoSubjects.getByIndex(i);
        qDebug() << subject.toString() << endl;
    }
    qDebug() << this->repoSubjects.getAmount() << endl;
    for (int i=0; i < this->repoLessonTime.getAmount(); ++i) {
        LessonTime lessonTime = this->repoLessonTime.getByIndex(i);
        qDebug() << lessonTime.toString() << endl;
    }
    qDebug() << this->repoLessonTime.getAmount() << endl;
    for (int i=0; i < this->repoGroupStudents.getAmount(); ++i) {
        GroupStudents          groupStudents          =          this-
>repoGroupStudents.getByIndex(i);
        qDebug() << groupStudents.toString() << endl;
    }
    qDebug() << this->repoGroupStudents.getAmount() << endl;
    for (int i=0; i < this->repoLinkGroupSubject.getAmount(); ++i)
{
        LinkGroupSubject       linkGroupSubject       =       this-
>repoLinkGroupSubject.getByIndex(i);
        qDebug() << linkGroupSubject.toString() << endl;
    }
    qDebug() << this->repoLinkGroupSubject.getAmount() << endl;
    qDebug() << "groups-subjects" <<this->groups_subjects;
}
```

2.3.5 Диаграммы

Диаграмма состояний и классов приведена в приложении А.

На этой схеме представлена часть диаграммы классов для отображения MVP архитектуры и связей между основными классами. Далее будут описаны и другие составляющие.

Были использованы два вида связи:

1. Отношение обобщения — это наследование.

2. Зависимость используется для организации диалога одного класса с другим. Классы либо используют методы и объекты другого, либо создаются внутри другого класса.

2.4. Алгоритм построения графа для визуализации

Выглядит следующим образом:

1. Сформирован список параметров, по котором производится фильтрование по каждой доле (например, для доли "Группа" параметр "5304" для доли "предмет" параметр "АКЗ"). По очереди производится фильтрование по 1 доли
2. Фильтрование по одной доле:
1. В списке вершин, отвечающих за первый параметр удаляем все вершины, которые не подходят по параметру, со всеми смежными ребрами
2. Просматривается список вершин предшествующих долей: если от вершины нет ребра, которая ведет в следующую долю, то она тоже удаляется. Процесс длится до тех пор, пока не дойдет до самой первой доли
3. Просматривается список вершин следующей доли: если до вершины нет ребра от вершины предыдущей доли, то она удаляется. Процесс длится до тех пор, пока алгоритм не дойдет до самой последней доли.
3. Переход к следующего параметра и доли и пункту 1. Повтор алгоритма

Полученный результат отображен на рисунке 11.

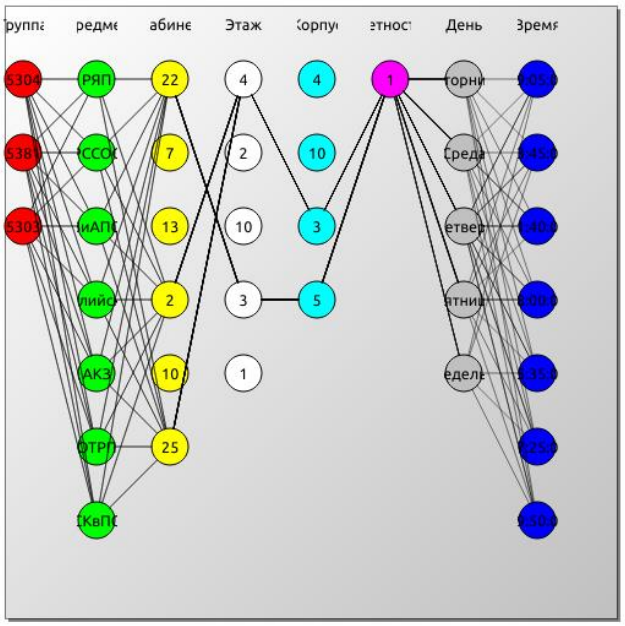


Рисунок 11 — Результат визуализации

2.5. Исключения

Обработка исключительных ситуаций – механизм языков программирования, предназначенный для описания реакции программы на ошибки времени выполнения и другие возможные проблемы (исключения), которые могут возникнуть при выполнении программы и приводят к невозможности (бессмысленности) дальнейшей отработки программой её базового алгоритма.

Реализовано пространство имен GraphContainer. В него входят все классы, связанные с графом и репозиториями.

Диаграмма для исключительных ситуаций содержится в приложении А.

2.6. Возможность сериализации

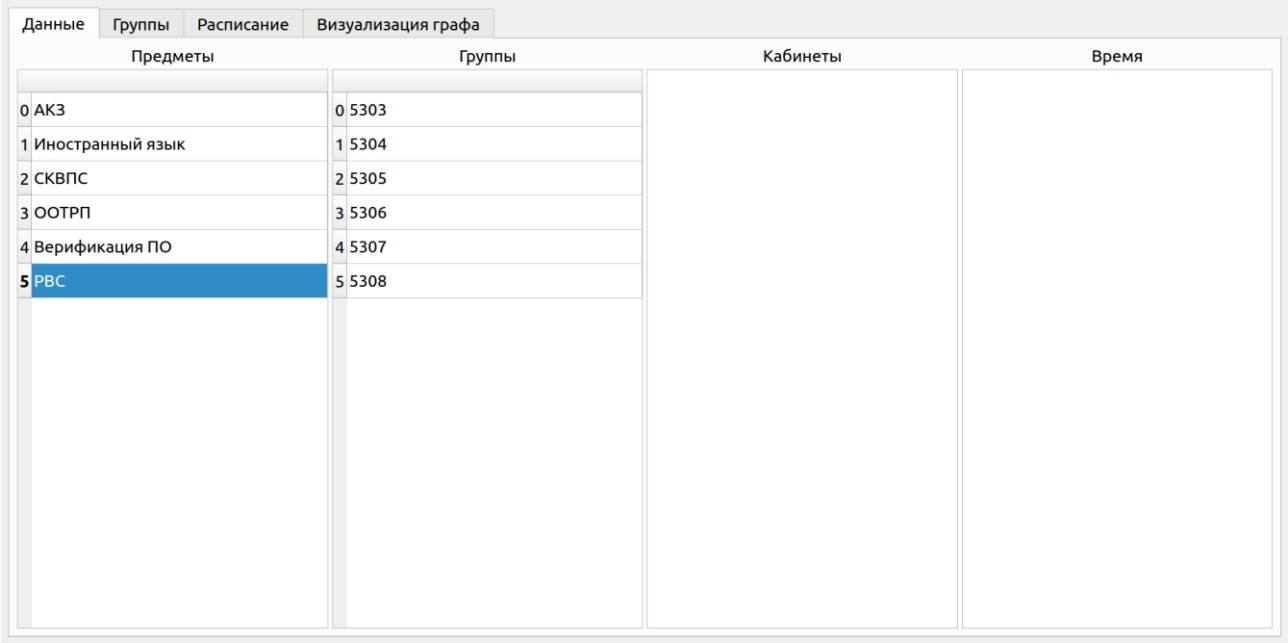
Сериализация – процесс перевода какой-либо структуры данных в последовательность битов. Обратной к операции сериализации является операция десериализации (структуризации) – восстановление начального состояния структуры данных из битовой последовательности.

Сериализация используется для сохранения данных в файл. Используется формат JSON. Можно сохранить как входные данные, так и выходные.

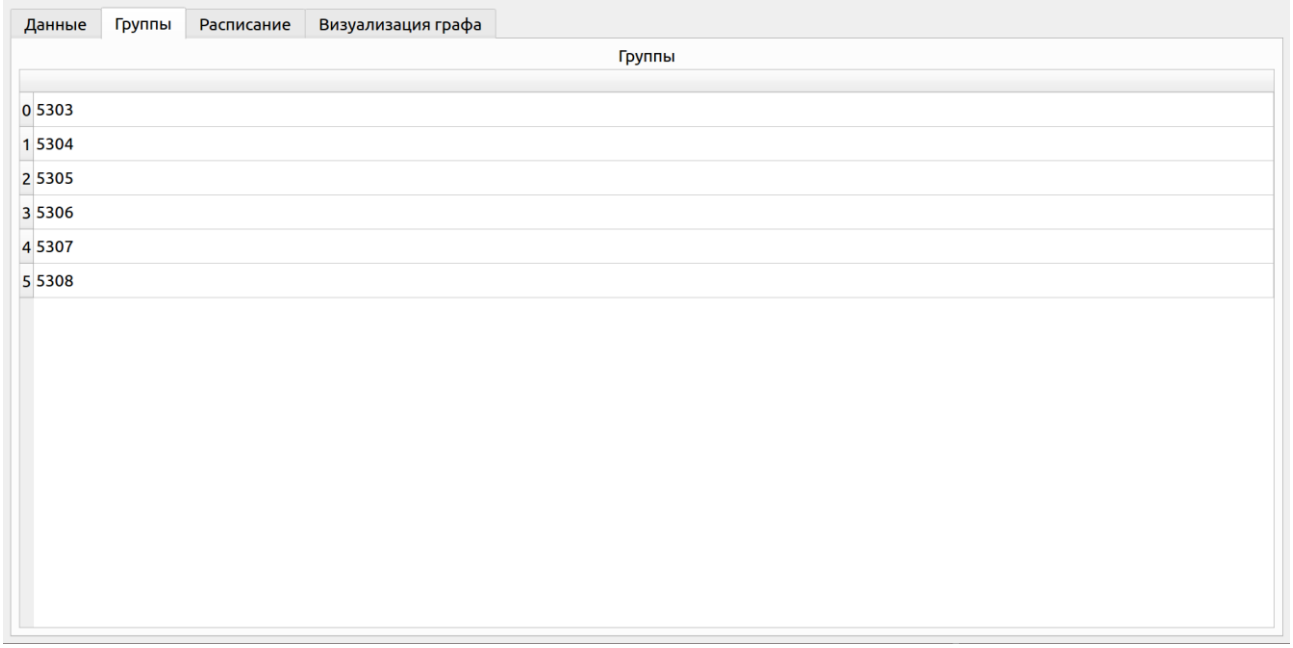
3. Завершение разработки и тестирование

3.1. Ручное тестирование приложения

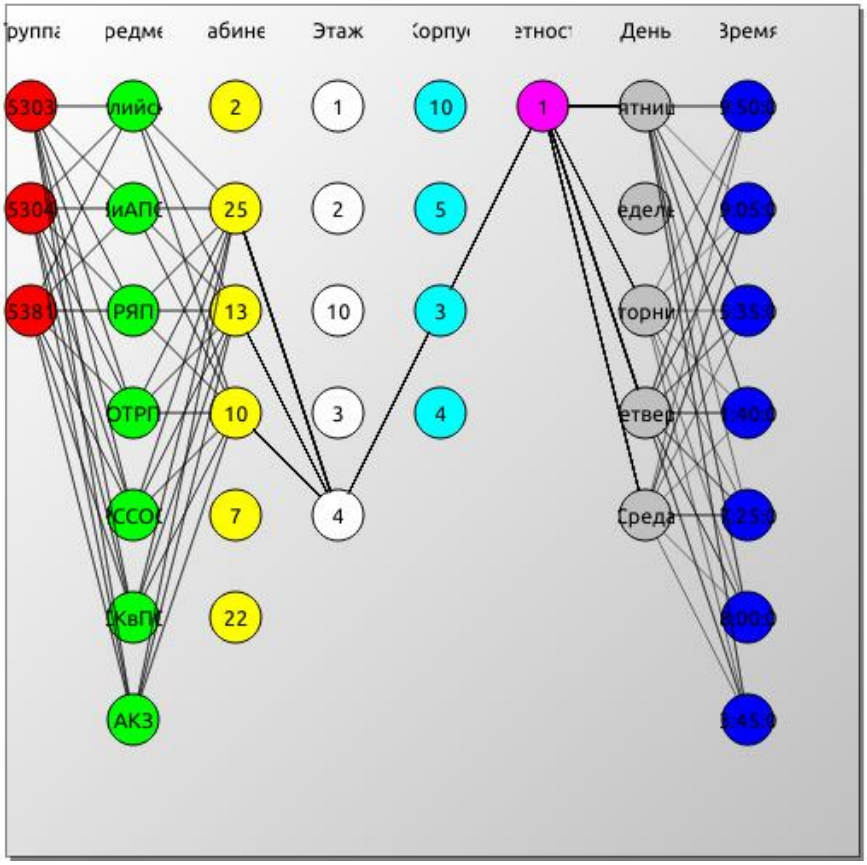
Произведено тестирование интерфейса разработанного приложения с помощью метода чёрного ящика. Приложение имеет следующий интерфейс с тестовыми данными. Главное окно:



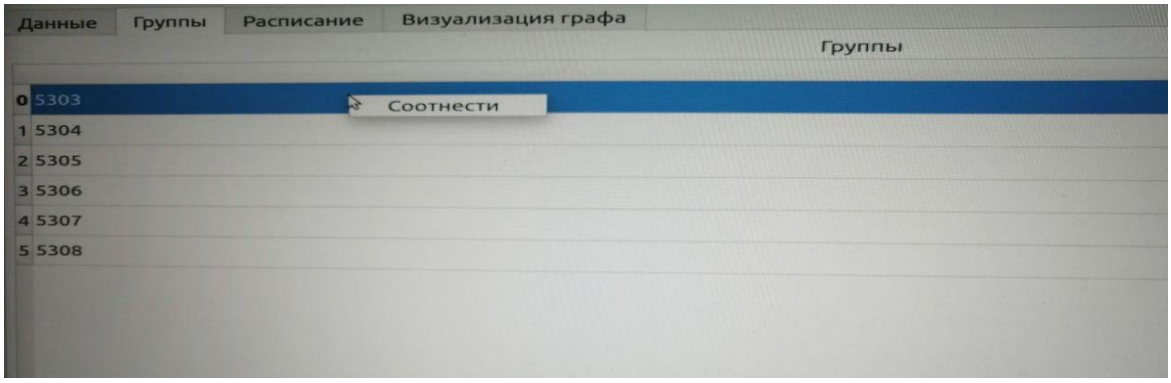
Данные групп:



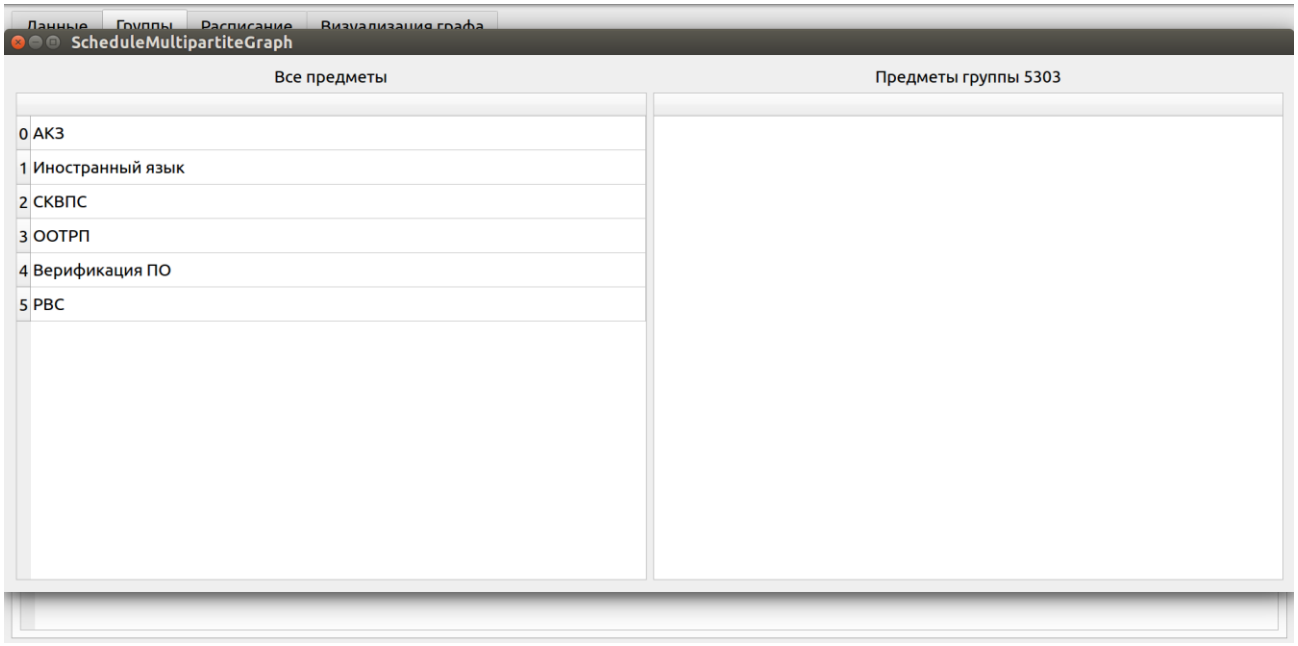
Визуализация графа:



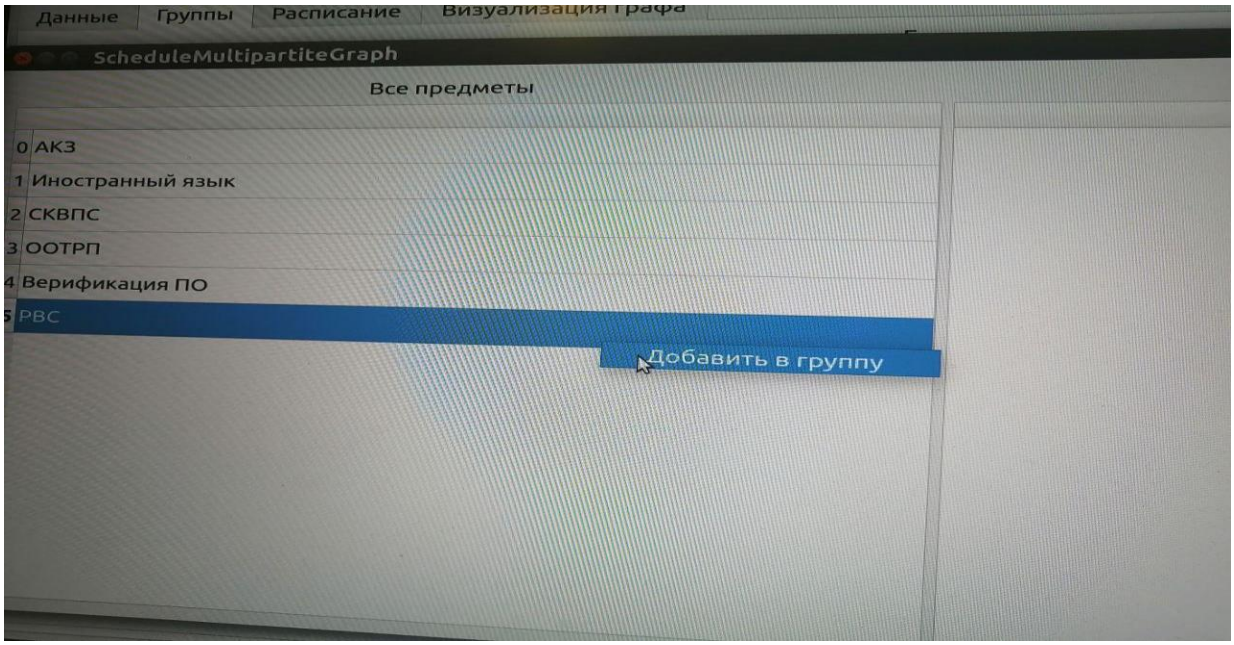
На вкладке группы, можно нажать на любую группу правой кнопкой, откроется контекстное меню: соотнести



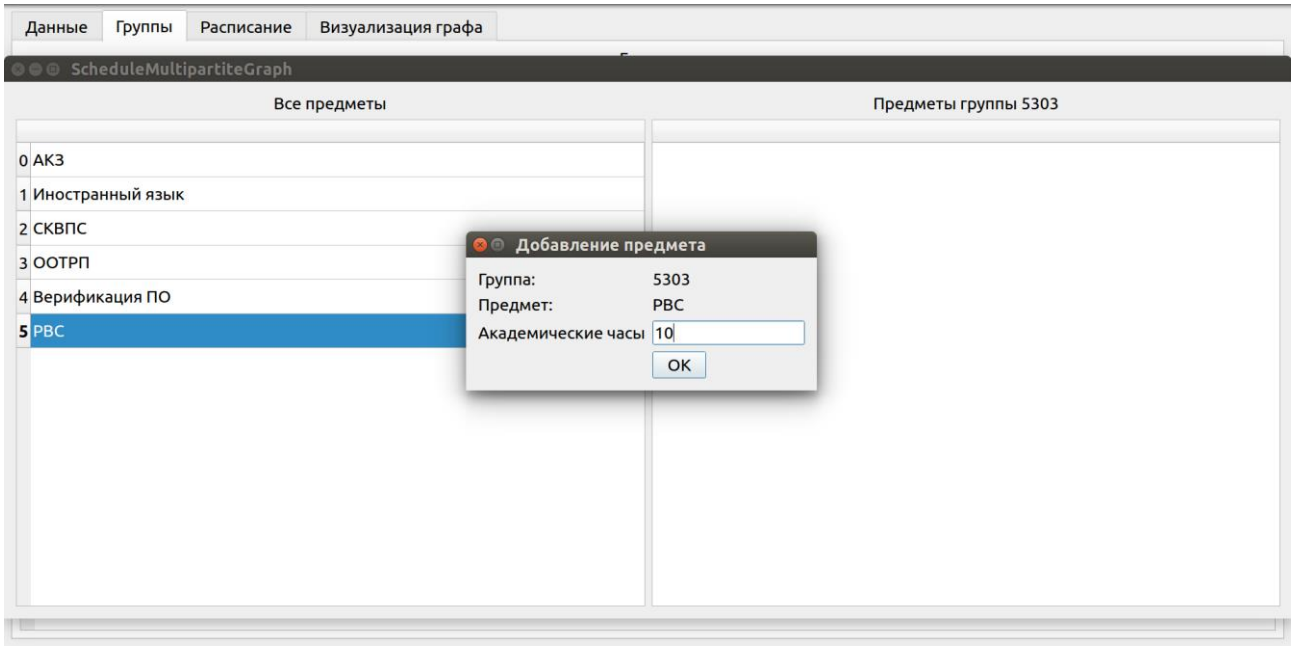
Далее откроется новое окно для добавления группировки предметов по группам с использованием контекстного меню:



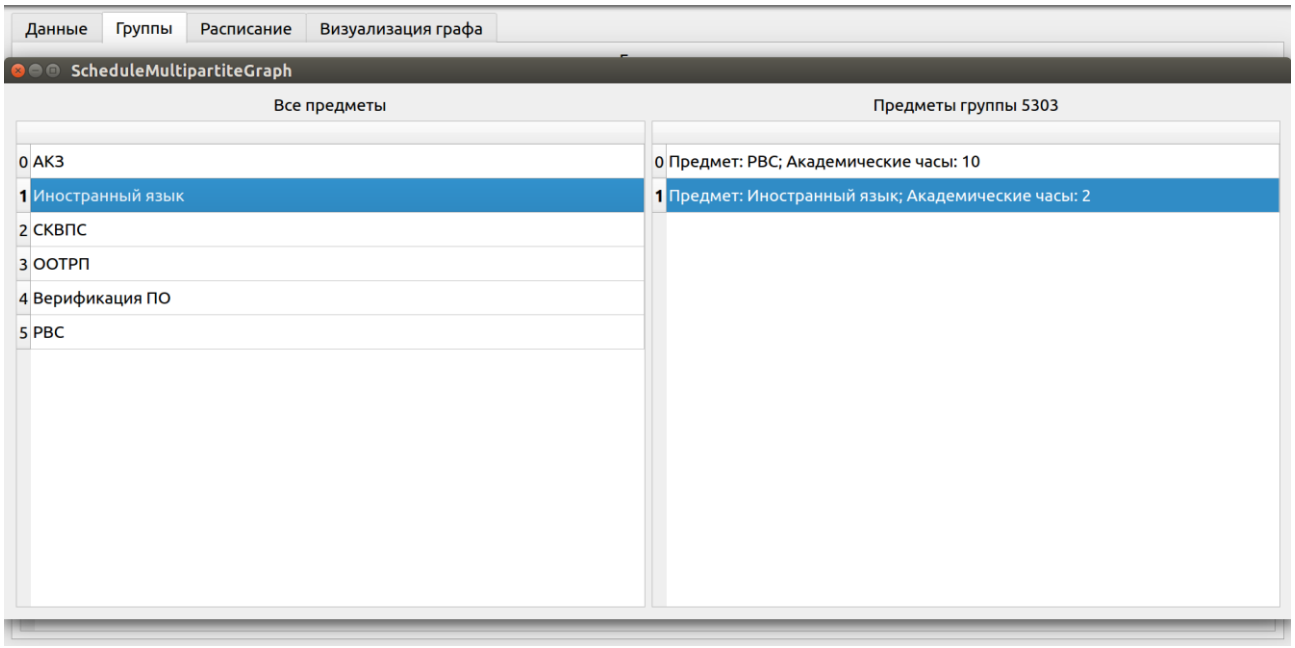
Добавить в группу:



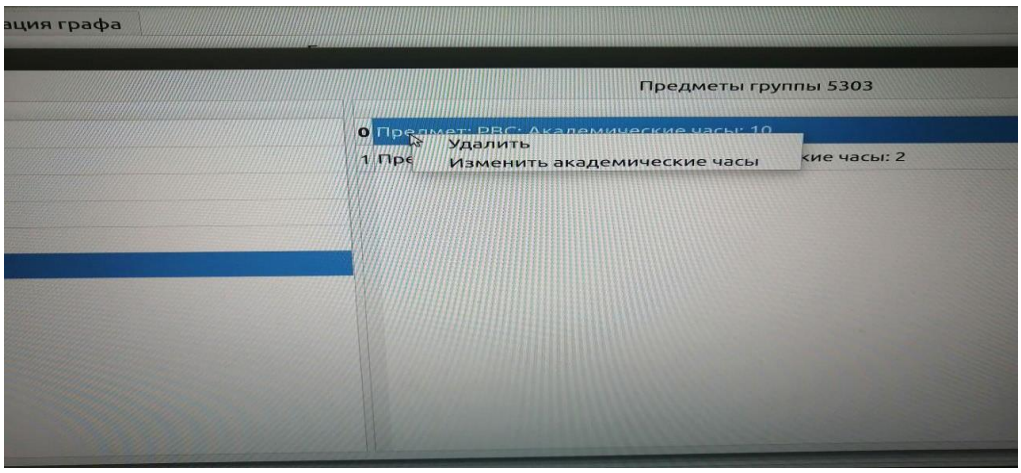
Добавление предмета:



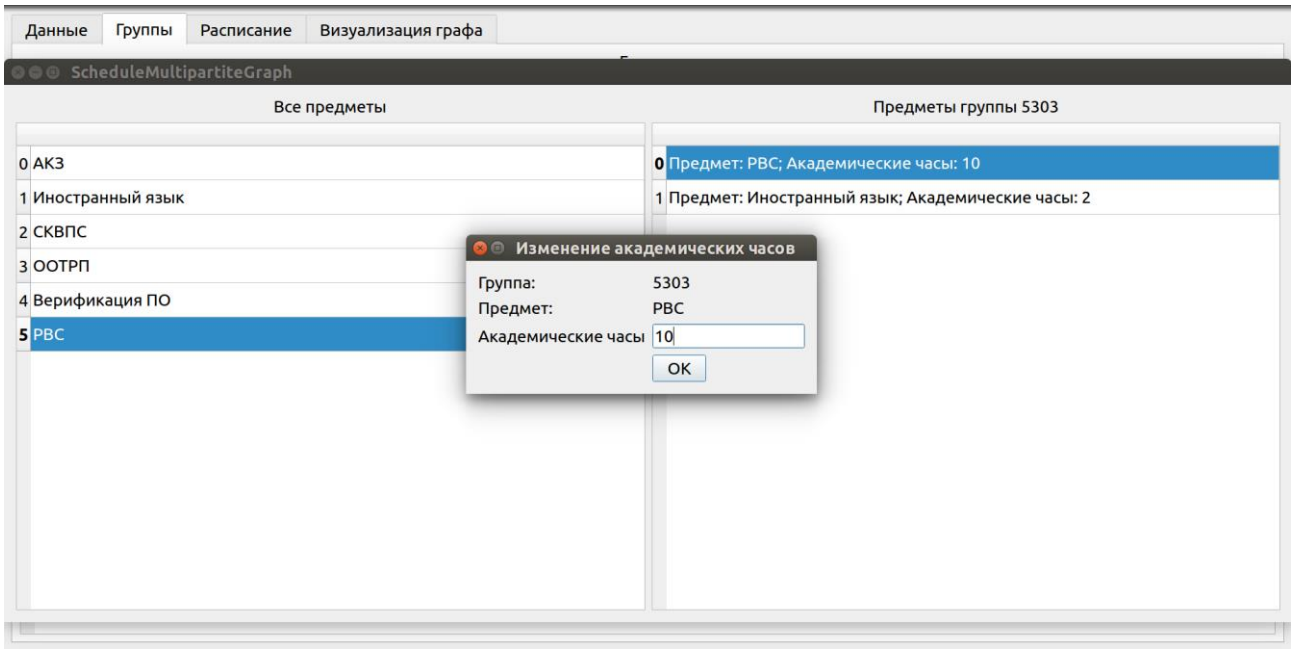
Предметы добавлены:

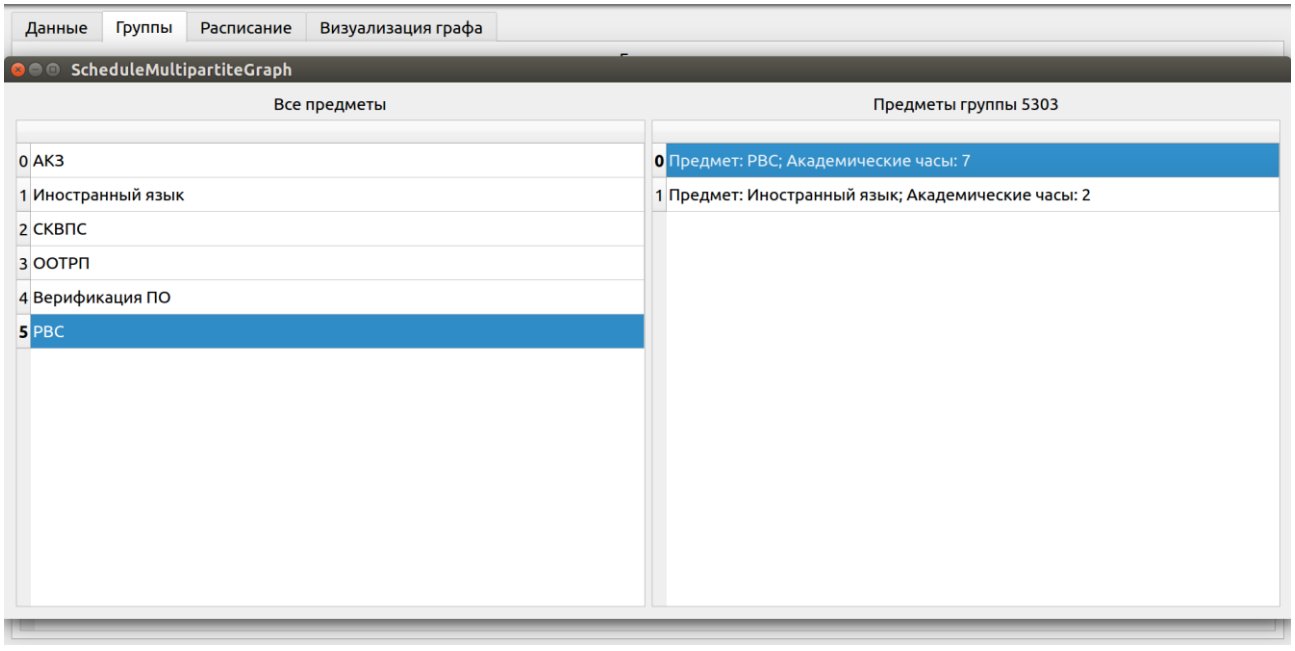


У добавленных предметов можно поменять академические часы и удалять предметы из групп с помощью контекстного меню:

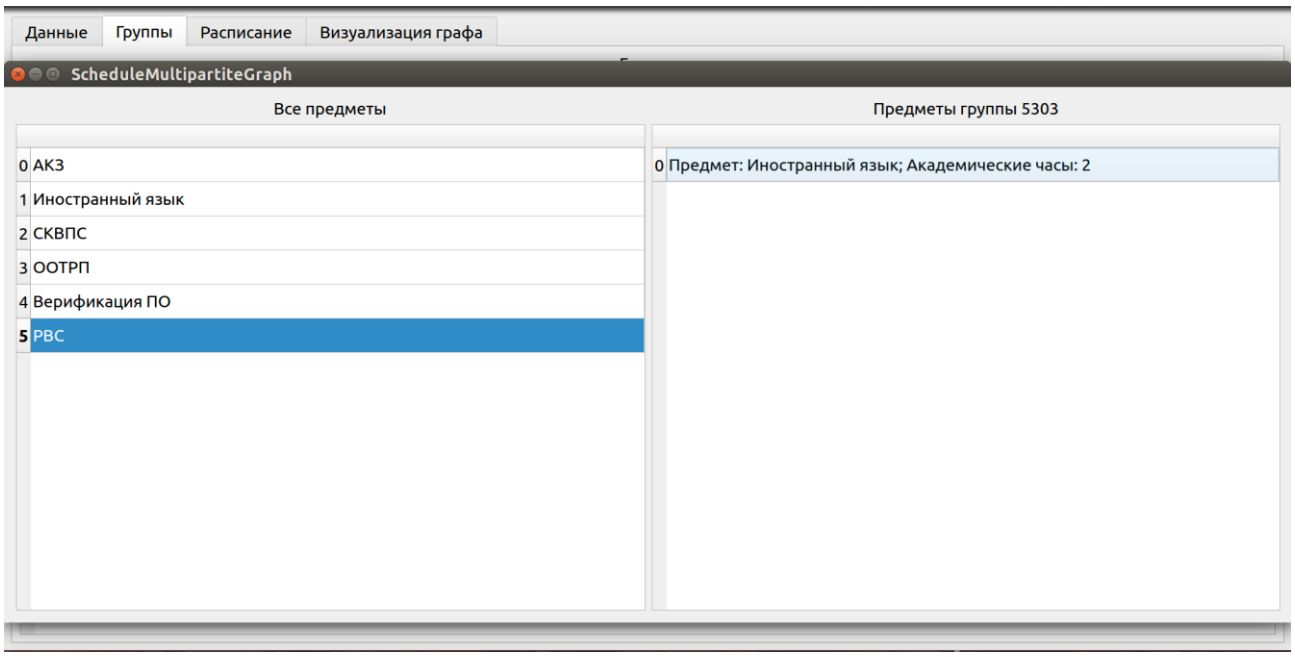


Изменить академические часы:

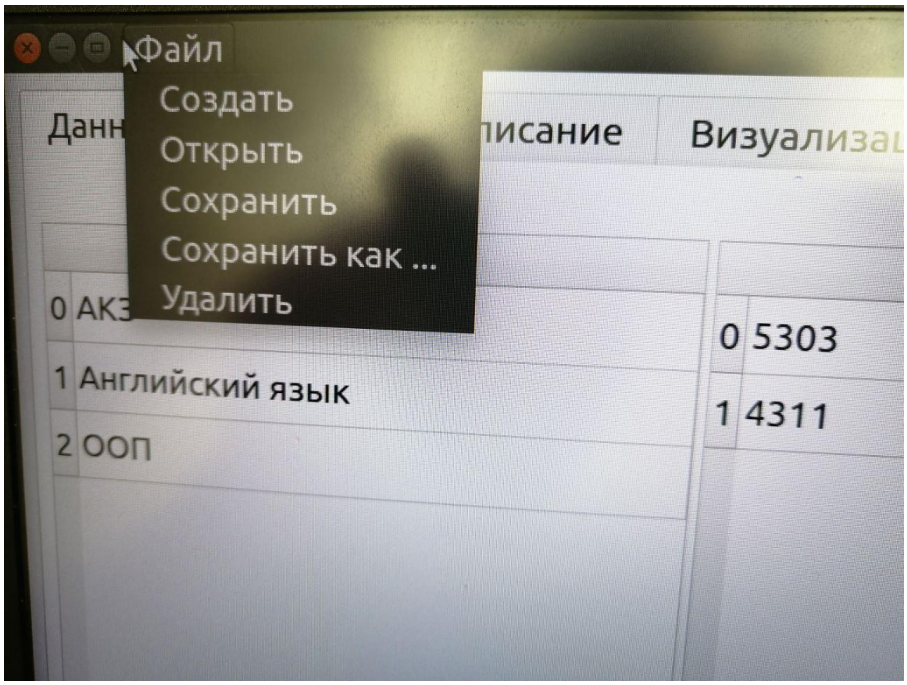




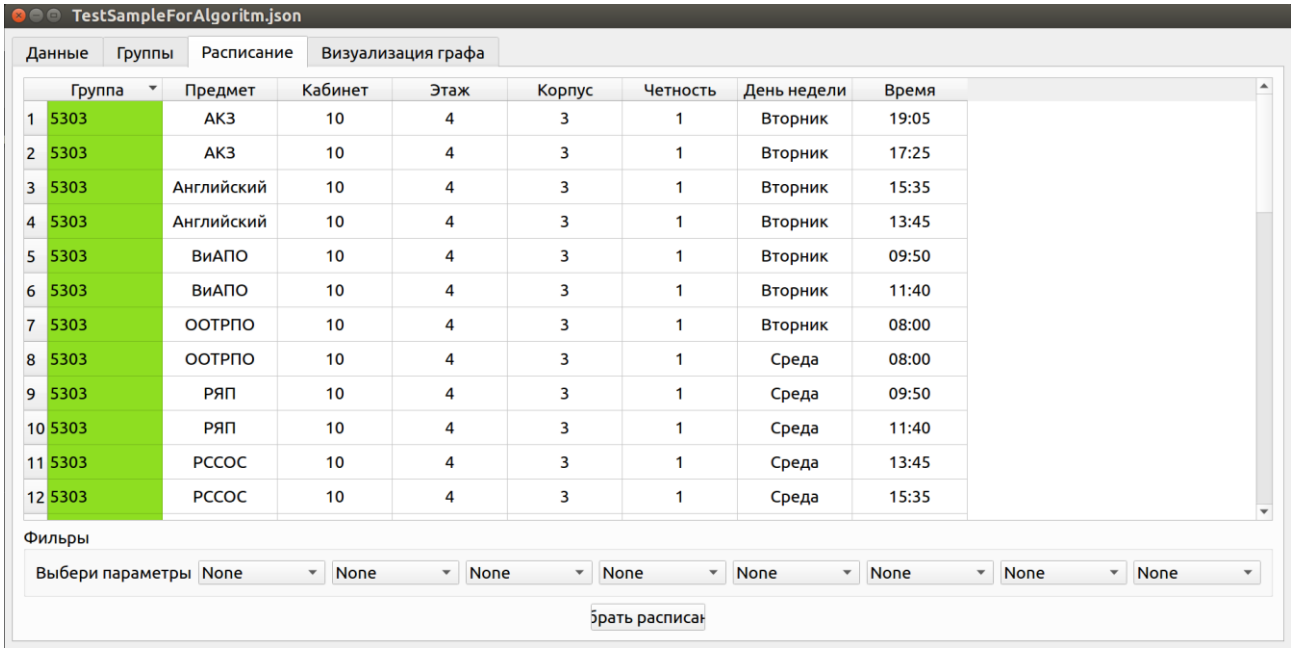
Удалить предмет:



Добавлено открытие, сохранение, сохранить как, удаление расписания в JSON — файл.



Страница составления расписания:



3.2 Стратегии и виды 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.

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 2. Ожидаемый результат соответствует полученному.

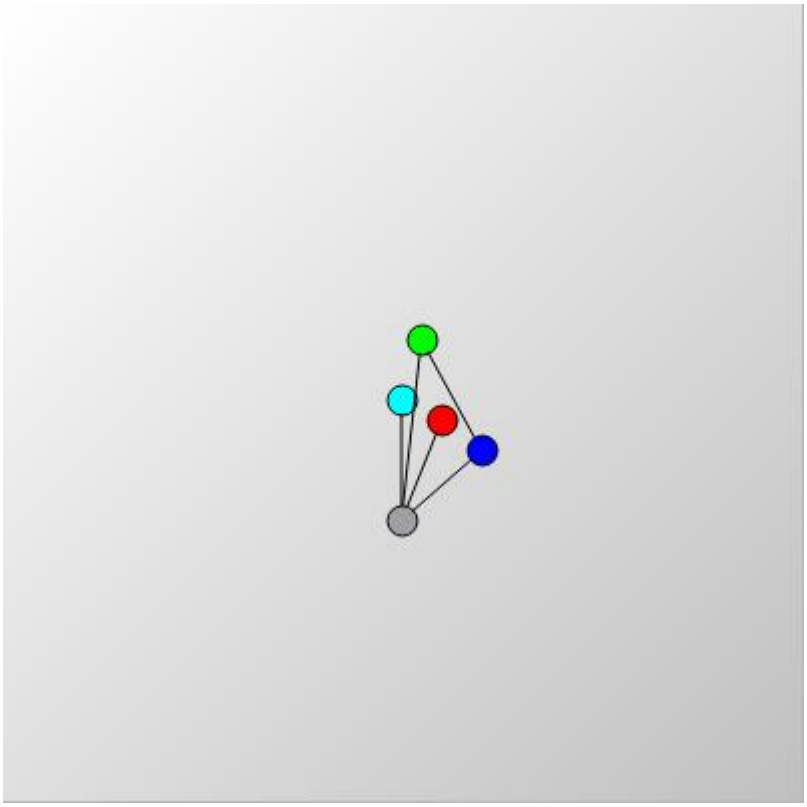


Рисунок 12 — Визуализация графа

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(repoRecGrou
pStudent.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).nam
e!=repoRecGroupStudent.getById(repoRecGroupStudent.getByIndex(i).i
d).name){
repoRecGroupStudent.update(repoRecGroupStudent.getById(repoRecGrou
pStudent.getByIndex(i).id).id,repoGroupStudents.getById(repoGroupS
tudents.getByIndex(i).id).name);
}
}
}
}
```

Ожидаемый результат соответствует полученному. Ошибок нет.

Для предмета:

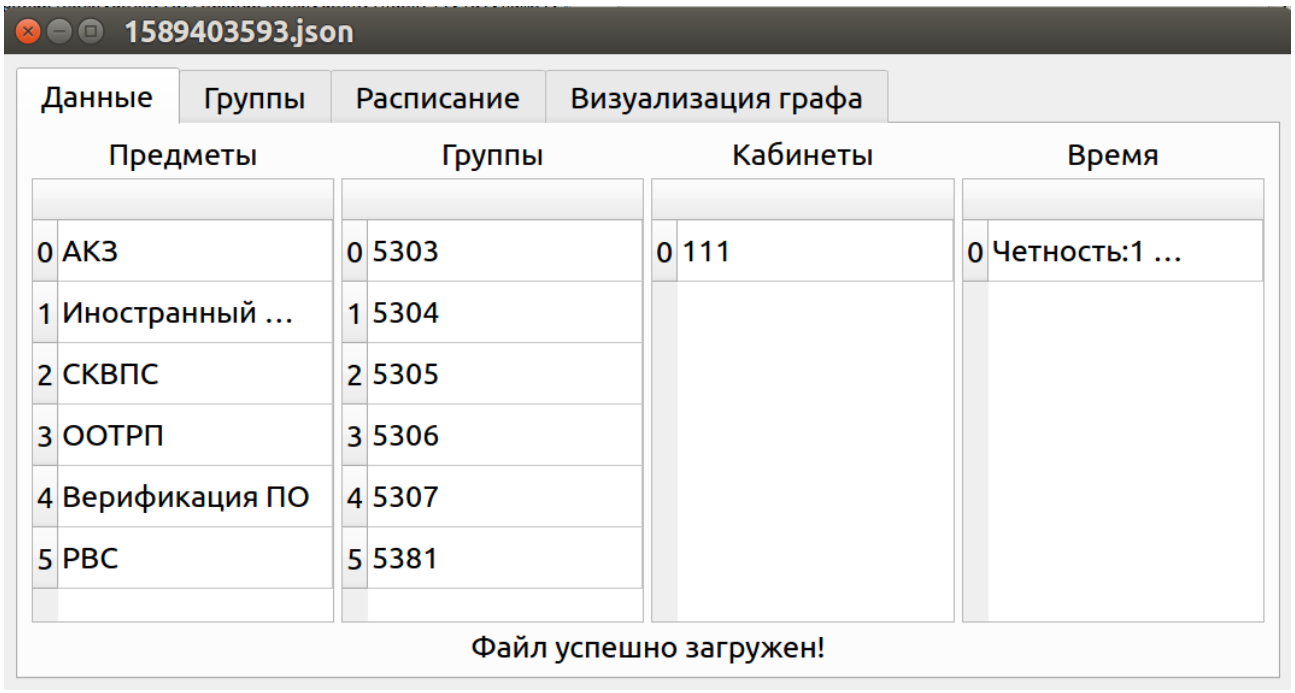
```
void
DialogLinkGroupSubjectWindow::editDataRepoSubject(RepositoryTempla
te<Subject> repoSubjects){
//Проверка на изменения(удаление, добавление, редактирование)
репозиториев в главной вкладке
if (repoRecSubject.getAmount()==0){
for (int i =0; i<repoSubjects.getAmount(); i++){
repoRecSubject.add(repoSubjects.getByIndex(i));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

}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.getByI
ndex(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!=repoRec
Subject.getById(repoRecSubject.getByIndex(i).id).name){
repoRecSubject.update(repoRecSubject.getById(repoRecSubject.getByI
ndex(i).id).id,repoSubjects.getById(repoSubjects.getByIndex(i).id)
.name);
}
}
}
}
```

Ожидаемый результат соответствует полученному. Ошибок нет.

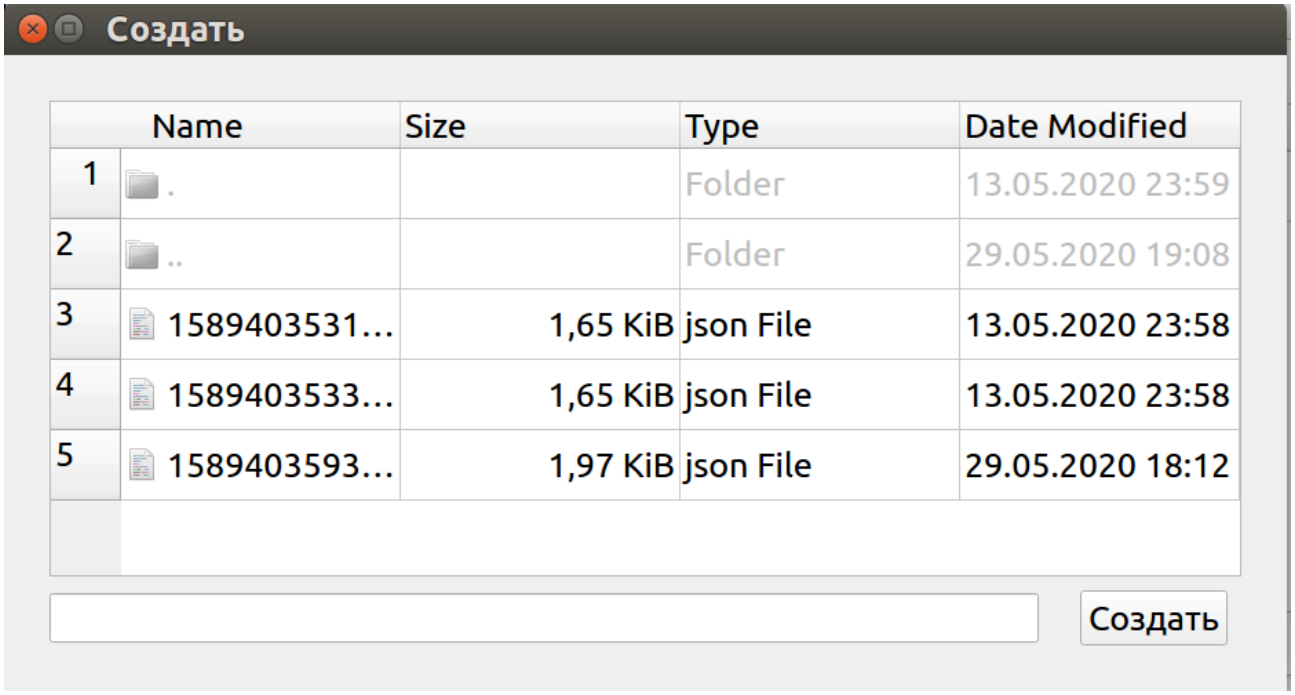
Регрессионное тестирование

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

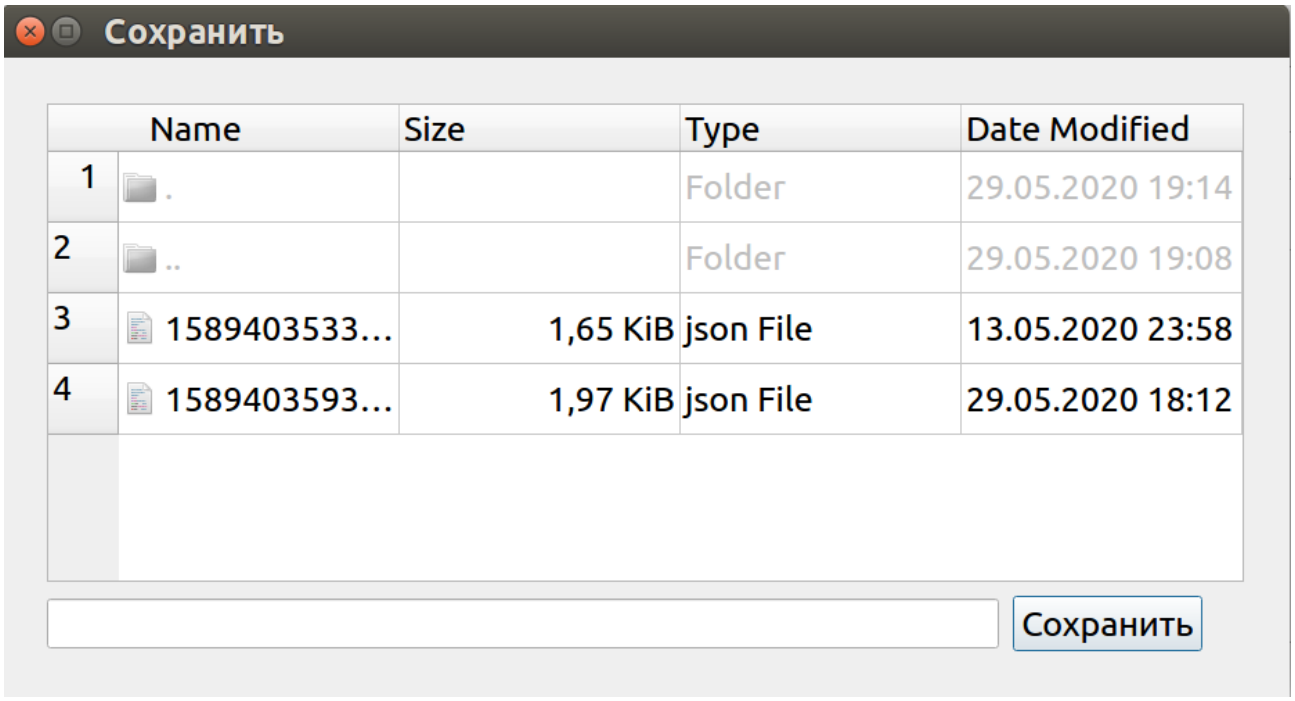
Открытие файла JSON при загрузке приложения и вывод сообщения:


1589403593.json
Данные
Группы
Расписание
Визуализация графа
Предметы
Группы
Кабинеты
Время
0 АКЗ
1 Иностранный …
2 СКВПС
3 ООТРП
4 Верификация ПО
5 РВС
0 5303
1 5304
2 5305
3 5306
4 5307
5 5381
0 111
0 Четность:1 …
Файл успешно загружен!

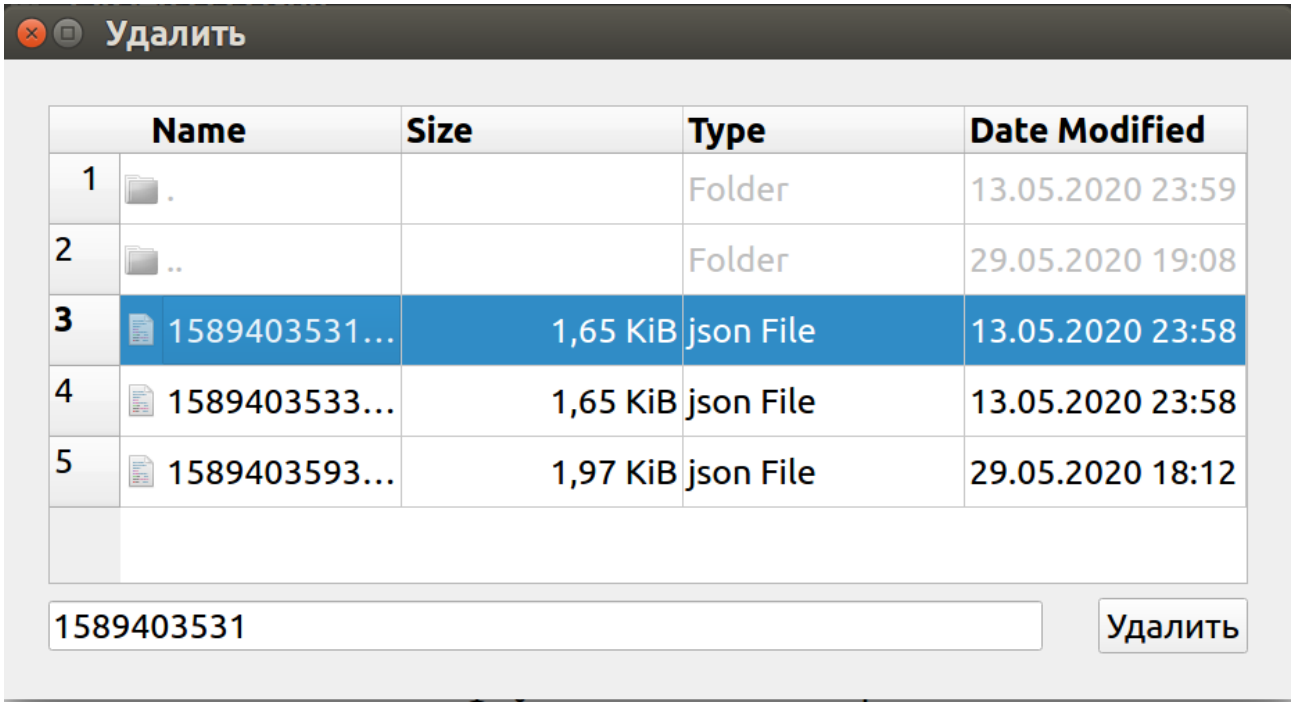
При создании открывается следующее окно:



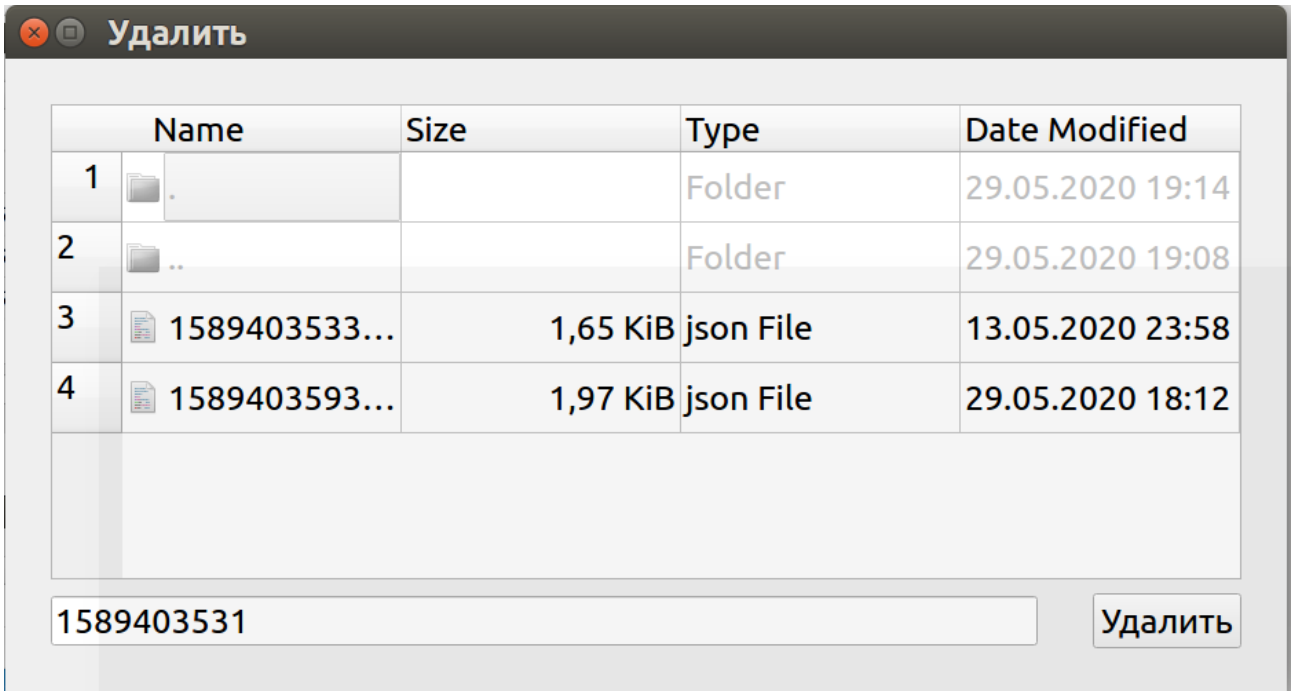
Сохранить как:



Удаление:



После удаления:



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.

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.

Тестирование стабильности (надежности)

Стабильность системы не зависит от времени работы приложения.

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.

3.3. Рефакторинг

Рефакторинг используется для изменения кода проекта с целью улучшения качества и читабельности кода.

Удаление функционально излишнего кода и избыточных комментариев. Исправление наименований функций и переменных для соответствия единому стилю именования.

Изменения:

- Удаление избыточного кода;
- Замена операторов на функции;
- Использование инструментов языка для сокращения типов;
- Удаление лишних пустых строк;
- Использование инструментов языка для сокращения типов;
- Добавление комментариев.

Далее показаны примеры рефакторинга:

1. Функция класса DialogAddlinkGroupSubject, упрощение метода копирования данных с объекта

```
158 121   void DialogAddLinkGroupSubject::editDataRepoSubject(RepositoryTemplate<Subject> repoSubjects){
159 122       //Проверка на изменения(удаление, добавление, редактирование) репозиториев в главной вкладке
160     -     if (receiveRepSubject.getAmount()==0){
161     -         for (int i =0; i<repoSubjects.getAmount(); i++){
162     -             receiveRepSubject.add(repoSubjects.getByIndex(i));
163     -         }
164     -
165     -     }else
166     -           if (repoSubjects.getAmount()>receiveRepSubject.getAmount()){
167     -             int raz = repoSubjects.getAmount()-receiveRepSubject.getAmount();
168     -             int addE = receiveRepSubject.getAmount();
169     -             for (int i =0; i<raz; i++){
170     -                 receiveRepSubject.add(repoSubjects.getByIndex(addE));
171     -                 ++addE;
172     -             }
173     -     }else
174     -           if (repoSubjects.getAmount()<receiveRepSubject.getAmount()){
175     -             //int raz = receiveRepSubject.getAmount()-repoSubjects.getAmount();
176     -             //int delE = receiveRepSubject.getAmount()-1;
177     -             for (int i =0; i<recdinSb.size(); i++){
178     -                 receiveRepSubject.remove(receiveRepSubject.getById(receiveRepSubject.getByIndex(recdinSb[i]).id).id);
179     -             //--delE;
180     -             }
181     -     }
182     -     else
183     -          if (repoSubjects.getAmount()==receiveRepSubject.getAmount()){
184     -             for (int i =0; i<repoSubjects.getAmount(); i++){
185     -                 if (repoSubjects.getById(repoSubjects.getByIndex(i).id).name!=receiveRepSubject.getById(receiveRepSubje
186     -                     receiveRepSubject.update(receiveRepSubject.getById(receiveRepSubject.getByIndex(i).id).id,repoSubje
187     -                 }
188     -             }
189     -          }
        123 +   receiveRepSubject=repoSubjects;
190 124     }
```

2. Функция класса DialogAddlinkGroupSubject, упрощение метод копирования данных с объекта

```
346 316   void DialogLinkGroupSubjectWindow::editDataRepoSubject(RepositoryTemplate<Subject> repoSubjects){
347     -     //Проверка на изменения(удаление, добавление, редактирование) репозиториев в главной вкладке
348     -     if (repoRecSubject.getAmount()==0){
349     -         for (int i =0; i<repoSubjects.getAmount(); i++){
350     -             repoRecSubject.add(repoSubjects.getByIndex(i));
351     -         }
        317 +     //изменение репозитория, согласно репозиторию на главной вкладке
        318 +      repoRecSubject=repoSubjects;
        319 +   }
352 320
```

3. Класс MainWindow, удаление дублирующихся полей

```
43  43          receiveDay[5]="Пятница";
44  44          receiveDay[6]="Суббота";
45  45
46      -
47      -       receiveDay[1]="Понедельник";
48      -       receiveDay[2]="Вторник";
49      -       receiveDay[3]="Среда";
50      -       receiveDay[4]="Четверг";
51      -       receiveDay[5]="Пятница";
52      -       receiveDay[6]="Суббота";
53      -
```

4. Класс MainWindow, удаление данных перемещено в отдельный метод из-за неоднократного использования удаления данных из файла.

```
881     -       RepositoryTemplate<Cabinet> lrepoCabinet;
882     -       RepositoryTemplate<Subject> lrepoSubject;
883     -       RepositoryTemplate<LessonTime> lrepoLessonTime;
884     -       RepositoryTemplate<GroupStudents> lrepoGroupStudent;
885     -       RepositoryTemplate<LinkGroupSubject> lrepoLinkGroupSubject;
886     -
887     -       repoCabinets= lrepoCabinet;
888     -       repoSubjects = lrepoSubject;
889     -       repoLessonTime = lrepoLessonTime;
890     -       repoGroupStudents = lrepoGroupStudent;
891     -       repoLinkGroupSubject = lrepoLinkGroupSubject;
892     -       //обнуление буфера группы_предметы
893     -       dialogLinkGroupSubject->repoLinkGroupSubjects = lrepoLinkGroupSubject;
894     -
895 886        object[this->repoCabinets.getTname()] = this->repoCabinets.toJson();
896 887        object[this->repoSubjects.getTname()] = this->repoSubjects.toJson();
897 888        object[this->repoLessonTime.getTname()] = this->repoLessonTime.toJson();
        @@ -1146,4 +1137,20 @@ void MainWindow::clearModel(){
1146 1137      clearTableView(ui->time_table,timeModel);
1147 1138      clearTableView(ui->gr_sub_table,gr_subModel);
1148 1139    }
     1140 + //очищение репозиториев
     1141 + void MainWindow::clearRepository(){
     1142 +     RepositoryTemplate<Cabinet> lrepoCabinet;
     1143 +     RepositoryTemplate<Subject> lrepoSubject;
     1144 +     RepositoryTemplate<LessonTime> lrepoLessonTime;
     1145 +     RepositoryTemplate<GroupStudents> lrepoGroupStudent;
     1146 +     RepositoryTemplate<LinkGroupSubject> lrepoLinkGroupSubject;
     1147 +
     1148 +     repoCabinets= lrepoCabinet;
     1149 +     repoSubjects = lrepoSubject;
     1150 +     repoLessonTime = lrepoLessonTime;
     1151 +     repoGroupStudents = lrepoGroupStudent;
     1152 +     repoLinkGroupSubject = lrepoLinkGroupSubject;
     1153 +     //обнуление буфера группы_предметы
     1154 +     dialogLinkGroupSubject->repoLinkGroupSubjects = lrepoLinkGroupSubject;
     1155 + }
```

5.Класс dialogAddlinkGroupSubject, упрощение метода копирования данных с объекта

```
312 310   void DialogLinkGroupSubjectWindow::editDataRepoGroup(RepositoryTemplate<GroupStudents> repoGroupStudents){
313       -     //Проверка на изменения(удаление, добавление, редактирование) репозиториев в главной вкладке
314       -     if (repoRecGroupStudent.getAmount()==0){
315       -         for (int i =0; i<repoGroupStudents.getAmount(); i++){
316       -             repoRecGroupStudent.add(repoGroupStudents.getByIndex(i));
317       -         }
      311 + //изменение репозитория, согласно репозиторию на главной вкладке
      312 +     repoRecGroupStudent = repoGroupStudents;
318 313
319       -     }else
320       -           if (repoGroupStudents.getAmount()>repoRecGroupStudent.getAmount()){
321       -             int raz = repoGroupStudents.getAmount()-repoRecGroupStudent.getAmount();
322       -             int addE = repoRecGroupStudent.getAmount();
323       -             for (int i =0; i<raz; i++){
324       -                 repoRecGroupStudent.add(repoGroupStudents.getByIndex(i));
325       -                 ++addE;
326       -             }
327       -     }else
328       -           if (repoGroupStudents.getAmount()<repoRecGroupStudent.getAmount()){
329       -             //int raz = repoRecGroupStudent.getAmount()-repoGroupStudents.getAmount();
330       -             //int delE = repoRecGroupStudent.getAmount()-1;
331       -             for (int i =0; i<dinGr.size(); i++){
332       -                 repoRecGroupStudent.remove(repoRecGroupStudent.getById(repoRecGroupStudent.getByIndex(dinGr[i]).id).id
333       -             //--delE;
334       -             }
335       -     }
336       -     else
337       -          if (repoGroupStudents.getAmount()==repoRecGroupStudent.getAmount()){
338       -             for (int i =0; i<repoGroupStudents.getAmount(); i++){
339       -                 if (repoGroupStudents.getById(repoGroupStudents.getByIndex(i).id).name!=repoRecGroupStudent.getById(re
340       -                     repoRecGroupStudent.update(repoRecGroupStudent.getById(repoRecGroupStudent.getByIndex(i).id).id,re
341       -                 }
342       -             }
343       -          }
344 314   }
```

3.4. Материалы поддержки пользователей

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.

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.

Техническая поддержка пользователей

Техническая поддержка (technical support) – это служба, в которую пользователи продукта или услуги могут обратиться за оказанием технической поддержки по решению возникшей проблемы, а также за получением дополнительной информации по интересующему вопросу.

Стоит также отметить, что служба технической поддержки может быть организована и для обслуживания сотрудников компании внутри организации, например, если сотрудникам нужна техническая помощь с компьютерной техникой (сломался компьютер/принтер) или неработающим программным обеспечением.

Отдельно можно выделить техническую поддержку клиентов (или клиентская поддержка). Такой вид поддержки имеет стратегическую направленность и нацелен на выстраивание долгосрочных отношений с клиентами.

Цели технической поддержки:

- оказание технической помощи пользователям при установке программного продукта
- оказание помощи в освоении и в решении проблем при использовании программного продукта
- сбор и систематизирование замечаний и пожеланий пользователей к программному продукту для улучшения качества использования данного продукта.

Обязанности службы:

- регистрация обращений пользователей в электронной системе
- перенаправление обращений/заявок пользователей к соответствующим специалистам техподдержки для решения проблемы
- ведение журнала с описанием выполненных действий и принятых решений с последующим занесением решений в единую базу
- оказание помощи и консультаций при установке, настройке и обновлении программного продукта, а также по использованию услуги

- оказание помощи в решении технических проблем при использовании продукта/услуги и дальнейшая ее координация
- оказание помощи по восстановлению работоспособности программного продукта после фатальных сбоев
- предоставление технической информации по функциональности продукта/услуги
- анализ проблем и разработка рекомендации по их устранению, которые были выявлены при использовании продукта/услуги
- отвечать на обращение пользователей в установленные сроки
- знание порядка и правил обработки обращений пользователей
- знание стандартных решений и ответов на наиболее часто задаваемые вопросы пользователей
- знание технических характеристик продукта

Виды технической поддержки

Техподдержку можно разделить на 3 направления:

- поддержка инфраструктуры
- поддержка пользователей
- сопровождение продуктов

Оказание услуг технической поддержки может быть предоставлена как на бесплатной, так и на платной основе. Платная техническая поддержка предоставляется на определенный срок по определенной цене и может включать в себя следующие услуги:

- круглосуточный мониторинг
- круглосуточная техподдержка
- техническая помощь с выездом специалиста т.е. помощь “на месте”
- резервное копирование, аварийное восстановление

Наиболее часто за оказанием технической поддержки можно обратиться через чат, электронную почту, форму “обратной связи”, телефон, панель управления, и, как правило, по рабочим дням.

В случае обращения пользователя, специалисты из техподдержки обязаны выяснить нюансы проблемы, найти и предложить способы решения этой проблемы, а также дать рекомендации по дальнейшему предотвращению подобной проблемы.

Обработка обращений пользователей и клиентов происходит с помощью программного обеспечения, например, Helpdesk и Salesforce.

Пока у разработчиков продукта нет достаточного количества средств, поэтому на данном этапе нет возможности нанять и организовать техническую поддержку пользователей. В дальнейшем планируется это сделать.

3.5. План развёртывания

Развёртывание программного обеспечения (Развёртывание ПО, Software deployment) — это все действия, которые делают программную систему готовой к использованию. Данный процесс является частью жизненного цикла программного обеспечения.

В целом процесс развертывания состоит из нескольких взаимосвязанных действий с возможными переходами между ними. Эта активность может происходить как со стороны производителя, так и со стороны потребителя. Поскольку каждая программная система является уникальной, трудно предсказать все процессы и процедуры во время развертывания. Поэтому «развертывание» можно трактовать как общий процесс, соответствующий определенным требованиям и характеристикам. Развертывание может осуществляться программистом и в процессе разработки программного обеспечения.

Определение стратегий обеспечения совместимости, преобразования и переноса

Если системе предстоит заменить существующую систему, то необходимо учесть аспекты, связанные с совместимостью, преобразованием и переносом. Конкретно:

- Данные из существующей системы должны быть перенесены (и, возможно, преобразованы в соответствующий формат) в новую систему.

- Существующие пользовательские интерфейсы (экранные форматы, команды и т.п.) должны поддерживаться в новой системе.
- Должны поддерживаться все существующие интерфейсы прикладных программ (API).
- Перенос из существующей системы в новую не должен нарушить работу пользователей более чем на заранее определенный период (конкретное значение зависит от вида бизнеса).
- У новой системы должна быть возможность работать параллельно со старой системой в течение переходного периода.
- Должна существовать возможность переключения обратно на старую систему, если это понадобится, в течение первых двух недель работы.
- Старые архивные данные могут понадобиться в новой системе. Если они зашифрованы, то шифровальные ключи потребуют особого внимания во время переноса.

Стратегии, выбранные для решения перечисленных вопросов, потребуют соответствующей поддержки в архитектуре и конфигурации системы

Определение расписания развертывания

Перенос системы в рабочую среду требует планирования и подготовки. Ниже перечислены технические факторы, которые нужно учесть:

- Может потребоваться подготовка пользователей системы.
- Необходимо подготовить рабочую среду и обучить персонал, чтобы он мог поддерживать работу системы.
- Необходимо разработать процедуры поддержки рабочего процесса, в том числе процедуры резервного копирования, восстановления и устранения неполадок.

Бизнес - факторы

Далее рассмотрены бизнес-факторы, влияющие на расписание развертывания:

- По причинам, связанным с бизнесом, систему может потребоваться развернуть к конкретной дате; несоблюдение этого условия может значительно снизить ценность системы.

- Могут быть периоды, когда развертывание системы невозможно по причинам, связанным с бизнесом или текущим рабочим процессом, включая, но не ограничиваясь этим, окончание отчетного финансового периода или период, в течение которого работа системы не должна прерываться.

Пики рабочей нагрузки и другие факторы в существующих системах и процессах могут препятствовать развертыванию в определенные моменты времени. Например:

1. Увеличенная нагрузка на систему: еженедельные, ежемесячные или ежегодные пики
2. Регулярные циклы обслуживания аппаратного или программного обеспечения - влияют и на готовность систем, и на персонал
3. Периоды выходных и отпусков
4. Плановые одноразовые нарушения рабочего процесса из-за модернизации аппаратного обеспечения или внедрения новых систем
5. Плановые реорганизации
6. Изменения устройств и средств.

- Некоторые системы должны работать постоянно (например, сетевые и телефонные коммутаторы); для таких систем развертывать новую версию нужно во время работы старой.

Определение последовательности развертывания

Некоторые системы должны развертываться постепенно, частями, по причинам, связанным с расписанием или готовностью. Если систему нельзя развернуть сразу целиком, то необходимо определить порядок установки компонентов и узлы, на которых они должны устанавливаться. Ниже перечислены общепринятые шаблоны планирования развертывания:

1. Географически - в зависимости от территории
2. Функционально - в зависимости от приложения
3. Организационно - в зависимости от подразделения или должностных обязанностей

Во время поэтапного развертывания приложения необходимо учитывать следующие аспекты:

- программное обеспечение должно работать и в условиях неполной конфигурации
- различные версии программного обеспечения должны быть совместимы
- должна существовать возможность возврата к предыдущей версии системы в случае обнаружения неполадок в новой системе

Определение необходимости обучения пользователей

Для каждой категории пользователей - администраторов, операторов и обычных пользователей - определите следующее:

- С какими системами информационных технологий они имеют дело в настоящий момент? Если при работе с данной системой какие-либо категории пользователей, как внутренних, так и внешних по отношению к организации, впервые столкнутся с той или иной информационной технологией, то отметьте это как пункт, требующий особого внимания.
- С какими новыми функциями они встретятся в данной системе?
- Чему потребуется их обучить, в широком смысле этого слова?
- Каковы требования относительно поддержки национальных языков (NLS)?

3.6. Компоновка продукта

Модульная организация программы - это разделение программы на более-менее независимые части (модули), их независимое проектирование и трансляция.

Иерархия

Любая сложная система не обходится без иерархии, без нее большая система превращается в нечто аморфное, необозримое и слабо управляемое.

Логическая иерархия отражает логических единиц программы, таких как функции, классы, библиотеки. Физическая иерархия касается физических единиц, на которые разбивается текст программы: файл (модуль), проект. Естественно, что между ними существует взаимосвязь, но не жесткая,

синтаксическая, а технологическая, соблюдаемая программистом. Иерархия программных единиц имеет три уровня:

- элементом самого нижнего уровня является функция (в объектно-ориентированном программировании – метод класса). Это автономная синтаксическая единица языка. В традиционной технологии структурного программирования под модульным программированием понимают именно это: представление программы в виде системы взаимодействующих функций;
- несколько функций, объединенных общем описанием обрабатываемых ими структур данных, составляют библиотеку функций (эквивалент в ООП - класс). Все это – элементы логической иерархии. В физическом представлении им соответствует модуль (в интегрированных, закрытых системах) или файл исходного текста. Особенность модульного программирования в том и состоит, что отдельные модули могут разрабатываться, транслироваться и частично отлаживаться отдельно друг от друга. Но для этого им могут потребоваться описания интерфейсов взаимодействия (в Си – заголовочные файлы);
- вся программа в целом образуют проект. В интегрированных системах проект и все его модули могут быть представлены одним файлом. В традиционных системах программирования (к ним относится и C/C++) проект состоит из файлов исходного текста – модулей, файла проекта, содержащего список модулей, настройки транслятора и т.п., а также вспомогательных файлов. В этом случае под проект отводится отдельная папка.

Фазы трансляции и выполнения программы

Подготовка программы начинается с редактирования файла, содержащего текст этой программы, который имеет стандартное расширение для данного языка. Затем выполняется его трансляция (компиляция), которая включает в себя несколько фаз: препроцессор, лексический, синтаксический, семантический анализ, генерация кода и его оптимизация. В результате трансляции получается объектный модуль. Файл объектного модуля имеет стандартное расширение obj. Компоновка (сборка) программы заключается в объединении одного или нескольких объектных модулей программы и объектных модулей, взятых из

библиотечных файлов, содержащих стандартные функции. В результате, будет получена исполняемая программа в виде файла, называемый загрузочный модуль или программный файл. В Windows стандартное расширение - .exe, а в Linux исполняемый файл определяется не по расширению, а по специальному флагу исполняемости. Полученный программный файл затем загружается в память и выполняется.

При модульном проектировании весьма важна разница между определением и объявлением объектов программы (переменных, функций, методов, классов). Определение переменной или функции – это фрагмент программы, в котором полностью задано содержание объекта и по которому происходит его трансляция во внутреннее представление. Объявление только упоминает объект языка и перечисляет его свойства, если он недоступен в данной точке программы. С учетом раздельного размещения определений и объявлений в проекте модульной Си-программы присутствуют три вида файлов (модулей):

- файлы исходного текста (с расширением - cpp), содержащие определения переменных, функций, методов;
- заголовочные файлы (с расширением - h), содержащие объявления для соответствующих файлов исходного текста;
- объектные модули (с расширением – obj), полученные в результате независимой трансляции файлов исходного текста.

Назначение заголовочных файлов заключается в том, что содержащиеся в них объявления позволяют сформировать правильный программный код для обращения к объекту языка, который определен в другом модуле. Для обращения к такому внешнему объекту необходимо подключить соответствующий заголовочный файл с его объявлением директивой include. То же самое касается данных и функций, содержащихся в библиотеках и библиотечных классах.

Для C++ это заголовочные файлы .h и файлы исходных коов .cpp. Также из-за того, что программа создана в виде проекта с использованием фреймворка Qt Creator, то есть еще файлы проекта с расширением .pro. Разработанный

программный продукт должен поддерживаться на операционных системах Linux и Windows.

Модульное программирование, компоновка

При независимой трансляции модулей получается объектный модуль, содержащий часть программы во внутреннем представлении, а также информацию о некоторых элементах программы в исходном (символьном) виде:

- программный код, использующий в своей работе только объекты языка (типы данных, переменные, функции), определенные в текущем модуле, полностью переводится во внутреннее (двоичное) представление;
- если объект языка допускает внешний доступ из других модулей, то в объектом модуле создается точка входа, содержащая его имя и внутренний адрес в пространстве объектного модуля;
- при трансляции обращения к внешнему объекту языка объявление, полученное из заголовочного файла позволяет сформировать программный код для обращения к нему. Но все равно неизвестным остается его адрес. Поэтому вместо адреса транслятор оставляет внешнюю ссылку, содержащую исходное (символическое) имя объекта.

Библиотека объектных модулей - это файл (библиотечный файл), содержащий набор объектных модулей и собственный внутренний каталог. Объектные модули библиотеки извлекаются из нее целиком при наличии в них требуемых внешних функций и переменных и используются в процессе компоновки программы.

Компоновка (редактирование связей) - это процесс сборки программы из объектных модулей и библиотек, который включает в себя:

- объединение адресных пространств отдельных модулей в единое адресное пространство программного файла
- соединение внешних ссылок и соответствующих им точек входа
- при отсутствии необходимых точек входа для внешних ссылок их поиск производится в указанных библиотечных файлах. Если точка входа найдена в библиотеке объектных модулей, то весь объектный модуль, содержащий эту

точку, компонуется в программу и для него повторяется описанный выше процесс.

UML-диаграмма развертывания представлена на рисунке 13.

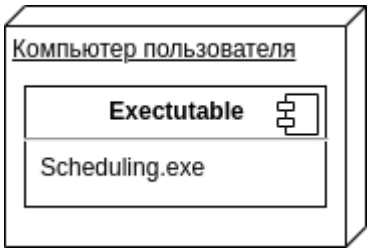


Рисунок 12 - Диаграмма развертывания

4. Материалы

Ссылка на репозиторий проекта:

https://github.com/valeriefuerte/ScheduleMultipartiteGraph

Видео демонстрация работы приложения:

https://drive.google.com/file/d/1cHE5ZCw6jKQCN3yos4GRHoIT0jHsTXPB/vie

Описание продукта

На видео продемонстрирована программа, содержащая ключевые компоненты (контейнер, также реализованы аллокатор и итератор, механизм исключений и прочее). 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. Имеются проверки на корректность выбранных данных. Все действия влияют на используемую модель (двудольный граф), добавляя и удаляя ребро между двумя долями. Расписание выводится, если перейти на вкладку «Расписание». Визуализируется граф во вкладке «Визуализация графа». Доступно удаление вершин и фильтрация.

ЗАКЛЮЧЕНИЕ

В ходе курсовой работы разработано приложение для составления расписания, а также осуществлена визуализация полученного результата.

Графический интерфейс пользователя позволяет удобно и быстро работать с системой. Реализованы контейнер - многодольный граф, аллокатор, итератор, механизм исключений и некоторые шаблоны проектирования, описанные выше. Реализована возможность открытия, редактирования, удаления и сохранения файла JSON с данными о группах, предметах, кабинетах, времени проведения занятий. Реализован алгоритм построения расписания, результат которого отображается в приложение, а также его можно сохранить в JSON. Граф, визуализирующий расписание, выводится в приложении. Приложение проходит все тесты.

Приложение А

Диаграмма компонентов

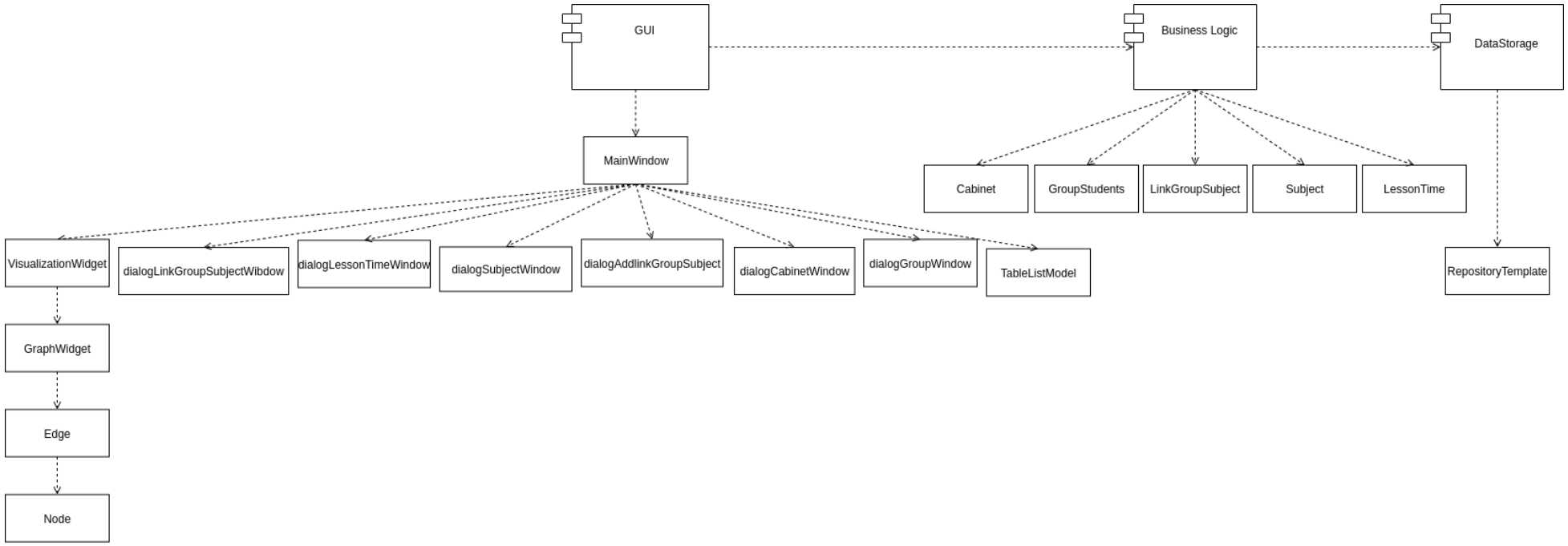


Диаграмма состояний

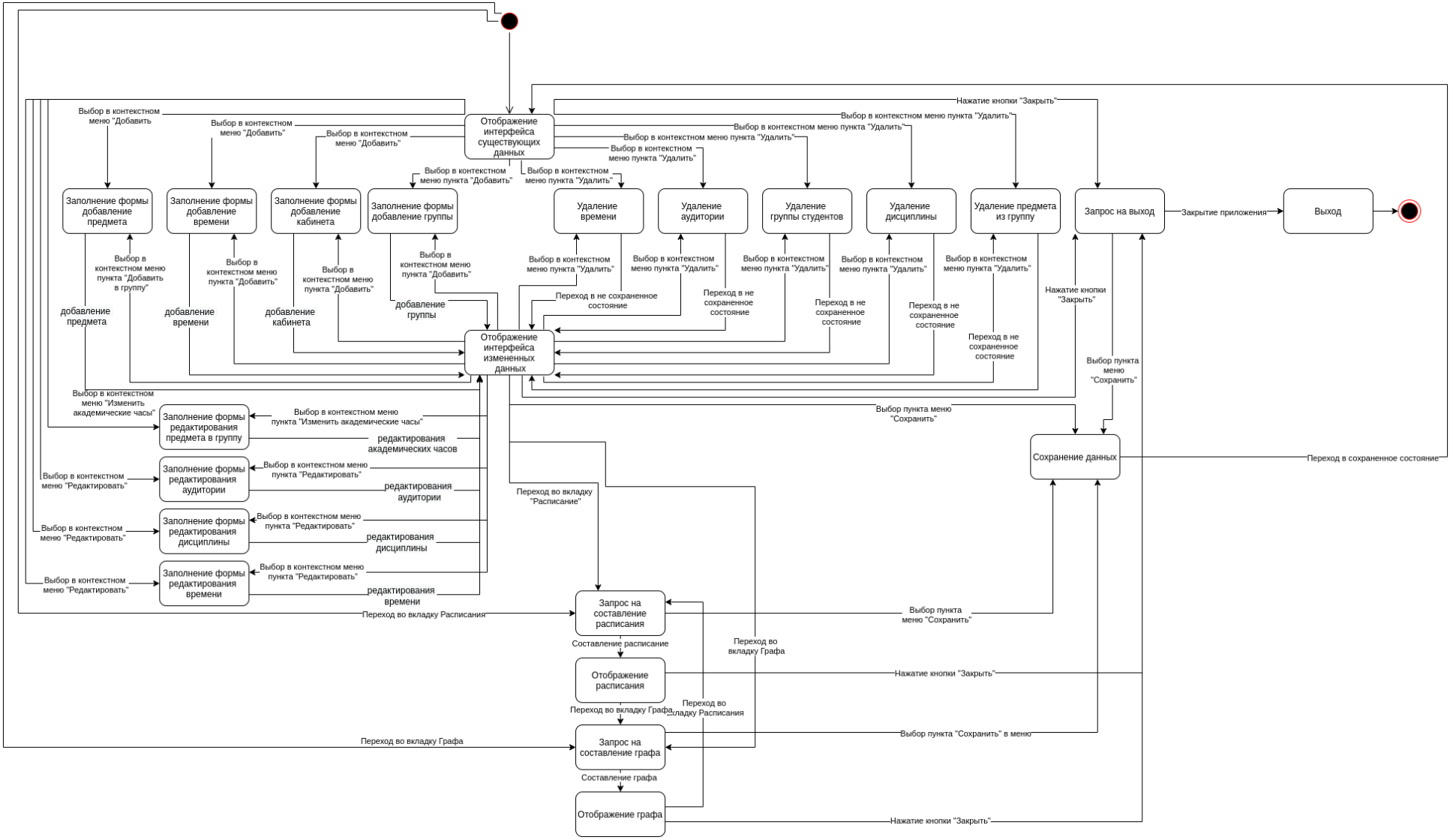


Диаграмма классов

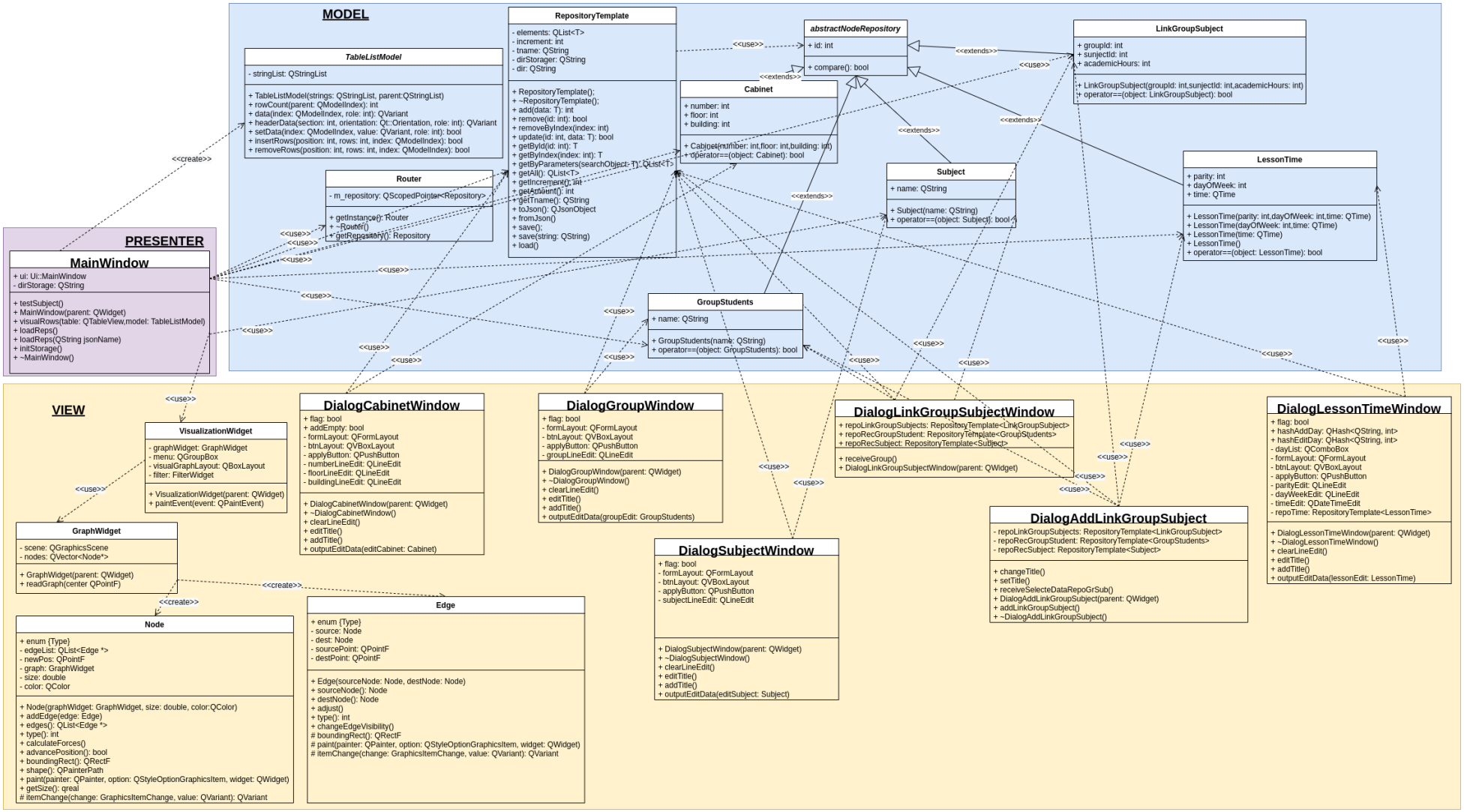


Диаграмма исключений