Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Дисциплина: Объектно-ориентированное программирование и проектирование

|  |
| --- |
|  |
|  |
|  |

Пояснительная записка

к курсовому проекту

на тему

**Проектирование системы поддержки составления расписания занятий**

БГУИР КП 1-53 01 02 06 018 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент |  | Шумигай В.В. |
| Руководитель |  | Ярмолик В.И. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Минск 2022

**РЕФЕРАТ**

ПРОЕКТИРОВАНИЕ СИСТЕМЫ ПОДДЕРЖКИ СОСТАВЛЕНИЯ РАСПИСАНИЯ ЗАНЯТИЙ: курсовой проект /В. В. Шумигай. – Минск: БГУИР, 2022, – п.з. –38 с., чертежей (плакатов) – 2 л. формата А1.

Курсовой проект на тему «Система поддержки и составления расписания занятий» разработан с целью автоматизации составления и управления расписанием учебного заведения.

Пояснительная записка к курсовому проекту состоит из введения, 3 разделов, включающих, анализ технического задания и создания спецификации к нему, проектирования системы, а именно создание различных диаграмм, реализацию системы, заключение, список использованных источников и приложение, содержащее листинг кода отдельных классов.

Для разработки проекта системы был выбран объектно-ориентированный язык проектирования UML. Проект представляет собой описание создания системы поддержки и составления расписания занятий. В нем присутствуют различные диаграммы, каждая из которых демонстрирует определенный аспект проекта.

В результате выполнения курсовой работы была спроектирована модель системы, написан глоссарий, а также составлены чертежи.

Результаты, полученные в ходе курсового проектирования, могут использоваться в организационной деятельности учебных заведений для автоматизации составления расписания учебных занятий и экзаменов, а также организации доступа к его редактированию.

Учреждение образования

«Белорусский государственный университет информатики   
и радиоэлектроники»

Факультет информационных технологий и управления

УТВЕРЖДАЮ

Заведующий кафедрой

*––––––––––––––––––––––––*

(подпись)

–––––––––––––––––––2022г.

ЗАДАНИЕ

по курсовому проектированию

Студенту  *Шумигаю Владиславу Викторовичу––––––––––––––*

1. Тема проекта *Система поддержки составления расписания занятий*

2. Срок сдачи студентом законченного проекта––––20 мая 2022 *г. ––––*

3. Исходные данные к проекту Система обеспечивает составление расписания некоторого учебного заведения, внесение в него изменений, выдачу полного расписания и дополнительной информации (например, по итоговому расписанию составляется расписание указанной группы или указанного преподавателя на заданный день или неделю).

В расписании фиксируются время и место проведения занятия, предмет и преподаватель, проводящий занятие, а также номер группы, для которой это занятие проводится. Некоторые занятия проводятся для нескольких групп: потока или всего курса.

Расписание не должно содержать коллизий (например, разные занятия не должны пересекаться друг с другом по месту и времени их проведения, один преподаватель не может вести одновременно два разных занятия, в одно и то же время, у одной и той же группы не допускается одновременно два различных занятия и т. д.).

Система по требованию пользователя должна осуществлять поиск свободных аудиторий на заданную дату и время.

При работе над этим вариантом задания необходимо разработать схему базы о расписании, курсах, преподавателях.*–––––––––––––––––––––––––––––*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке):

*Введение*

1. *Анализ технического задания на курсовое проектирование*
2. *Проектирование системы поддержки составления расписания занятий*
3. *Программная реализация системы поддержки составления расписания занятий*

*Заключение*

*Список использованных источников*

*Приложения*

*Ведомость курсового проекта*

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*1. Диаграмма вариантов использования––––––––––––––– –*

*2. Диаграмма последовательности. –––––––––––––––––-----------------------–––*

6. Консультант по проекту (с обозначением разделов проекта) *Ярмолик В. И.–*7. Дата выдачи задания *12 февраля 2022 г.* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_––– \_\_–

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

*раздел 1 к 15.03–15 %; –––––––––––––––– –––––––––––––––\_\_\_\_\_\_\_\_\_\_\_\_\_––*

*раздел 2к 15.04–50 %;–––––––––––––––– –––––––––––––––––– \_\_\_\_\_\_\_\_-––––*

*раздел 3к 15.05 –15 %;––––––––––––––––––––– –––––––––––––––\_\_\_\_\_\_\_\_––*

*оформление пояснительной записки и графического  
материала к 20. 05 – 20 %*

*Защита курсового проекта с* ***23.05*** *по* ***06.06.2022*** *г.––––––––––––––––––––––*

Руководитель*– ––––––––––*

(подпись)

Задание принял к исполнению *–––––––\_\_\_\_\_\_\_\_\_\_\_\_––*

(дата и подпись студента)

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc8657323)

[1 Анализ технического задания на курсовое проектирование 8](#_Toc8657324)

[1.1 Постановка задачи проектирования 8](#_Toc8657325)

[1.2 Составление глоссария проекта 9](#_Toc8657326)

[1.3 Описание дополнительных спецификаций к проекту 9](#_Toc8657327)

[2 Проектирование системы 11](#_Toc8657328)

[2.1 Создание диаграммы вариантов использования 11](#_Toc8657329)

[2.2 Добавление описаний к вариантам использования 12](#_Toc8657331)

[2.3 Создание диаграммы взаимодействия 22](#_Toc8657332)

[2.4 Создание диаграммы классов 24](#_Toc8657333)

[2.5 Создание диаграммы «сущность-связь»](#_Toc8657334) 27

[2.6 Создание диаграммы активности 27](#_Toc8657335)

[2.7 Создание диаграммы базы данных 28](#_Toc8657336)

[3 Программная реалиазация системы 30](#_Toc8657337)

[3.1 Создание диаграммы компонентов 30](#_Toc8657338)

[3.2 Создание диаграммы размещения 31](#_Toc8657339)

[3.3 Генерация кода C++ 32](#_Toc8657340)

[Заключение 33](#_Toc8657341)

[Список использованных источников 35](#_Toc8657342)

[Приложение А(обязательное) Листинг кода 36](#_Toc8657343)

[Ведомость курсового проекта 38](#_Toc8657346)

# ВВЕДЕНИЕ

В силу специфичности производства ПО (практически нулевая стоимость тиражирования, очень быстрый процесс устаревания и т. д.) технология его создания очень сильно завязана на человеческий ресурс и поэтому должна включать в себя организационный и управленческий аспекты. На сегодняшний день в мире существует огромное количество различных процессов для создания ПО. Тем не менее, именно технологий, рассматривающих полный жизненный цикл проекта разработки ПО, сочетающих в себе научный подход, серьезную базу исследований и имеющих историю реального использования и адаптации, относительно немного.

Для коммерческих предприятий, мы используем язык проектирования *UML*. Он отлично подходит для данной задачи, так как обладает широким спектром различных инструментов.

Данная работа направлена на демонстрацию освоения *Rational* *Rose* и *UML*. В ней используются основные виды диаграмм и на примере задания, указаны их основные характеристики.

*UML* используется для проектирования систем. Он является унифицированным, общепризнанным языком объектно-ориентированного программирования. *UML* содержит стандартный набор диаграмм и нотаций самых разнообразных видов.

*UML* находится в процессе стандартизации, проводимом консорциумом *OMG* (*Object* *Management* *Group*), в настоящее время он принят в качестве стандартного языка моделирования и получил широкую поддержку. U*M*L принят на вооружение практически всеми крупнейшими компаниями – производителями программного обеспечения (*Microsoft*, *IBM*, *Hewlett*-*Packard*, *Oracle*, *Sybase* и др.). Кроме того, практически всемировые производители CASE-средств, помимо *Rational Software* (*Rational Rose*), поддерживают UML в своих продуктах (*Paradigm Plus* (*CA*)*, System Architect (Popkin Software), Microsoft Visual Modeler* и др.).

Так же можно выделить преимущества *UML*. *UML* объектно-ориентирован, в результате чего методы описания результатов анализа и проектирования семантически близки к методам программирования на современных объектно-ориентированных языках. *UML* позволяет описать систему практически со всех возможных точек зрения и разные аспекты поведения системы. Диаграммы *UML* сравнительно просты для чтения после достаточно быстрого ознакомления с его синтаксисом. *UML* расширяет и позволяет вводить собственные текстовые и графические стереотипы, что способствует его применению не только в сфере программной инженерии. *UML* получил широкое распространение и динамично развивается.

Задачами курсового проекта (работы) как этапа подготовки к дипломному проектированию являются:

– освоение, углубление и обобщение знаний, полученных студентами в процессе обучения;

– приобретение практических навыков и развитие творческих подходов к решению конкретной инженерной или инженерно-экономической задачи;

– формирование умений использовать справочную литературу, нормативную, правовую, нормативно-техническую документацию, осуществлять патентный поиск;

– приобретение навыков по оформлению текстовой и графической документации согласно требованиям государственных стандартов и стандарта предприятия Дипломные проекты (работы). общие требования. СТП 01-2017.

Целью курсового проекта является объектно-ориентированный анализ и проектирование программного обеспечения.

Темой курсового проекта является проектирование системы поддержки составления расписания занятий. Данная тема актуальна, потому что в современном мире всё больше задач переносится на электронные устройства, а основным потребителем данного продукта является молодёжь, для которой удобно пользоваться системой на электронном устройстве. Основное преимущество состоит в том, что автоматизированное составление устраняет массу рутинной работы, такой как: поиск возможных вариантов внесения очередных элементов в расписание, проверок выполнения требований, поиск случайных ошибок в готовом расписании, оформление расписания на бумаге в виде различных таблиц, оставляя человеку больше времени на более интеллектуальные действия. Компьютер в данном случае также является инструментом, существенно усиливающим способности человека, так как человек не в состоянии перебрать и проанализировать такое же количество вариантов расписаний, как компьютер.

# АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

## 1.1 Постановка задачи проектирования

Система обеспечивает составление расписания некоторого учебного заведения, внесение в него изменений, выдачу полного расписания и дополнительной информации (например, по итоговому расписанию составляется расписание указанной группы или указанного преподавателя на заданный день или неделю). В расписании фиксируются время и место проведения занятия, предмет и преподаватель, проводящий занятие, а также номер группы, для которой это занятие проводится. Некоторые занятия проводятся для нескольких групп: потока или всего курса.

Расписание не должно содержать коллизий (например, разные занятия не должны пересекаться друг с другом по месту и времени их проведения, один преподаватель не может вести одновременно два разных занятия, в одно и то же время, у одной и той же группы не допускается одновременно два различных занятия и т. д.).

Система по требованию пользователя должна осуществлять поиск свободных аудиторий на заданную дату и время.

При работе над этим вариантом задания необходимо разработать схему базы о расписании, курсах, преподавателях.

**При проектировании системы требуется:**

* создать иерархию классов системы;
* для классов указать стереотипы;
* разместить классы по пакетам;
* построить диаграммы классов системы, отображающие связи между классами;
* связать объекты на диаграммах взаимодействия с классами, а сообщения – с операциями;
* каждый класс снабдить описанием, которое должно включать в себя краткое описание;
* для описания поведения экземпляров отдельных классов построить диаграммы состояний;
* построить диаграммы деятельности для моделирования сложных операций;
* разработать схему базы данных и отобразить ее на диаграмме «сущность – связь».

**При реализации системы необходимо** построить диаграммы компонентов для каждого пакета и для системы в целом. Также следует разработать диаграмму размещения. В зависимости от варианта задания диаграмма размещения должна показывать расположение компонентов в распределенном приложении или связи между встроенным процессором и устройствами. Должна быть произведена проверка корректности модели средствами *Rational* *Rose*.

## 1.2 Составление глоссария проекта

Глоссарий предназначен для описания терминологии предметной области составим глоссарий проекта. Он может быть использован как неформальный словарь данных системы. Глоссарий приведен в таблице 1.

Таблица 1 – Глоссарий проекта

|  |  |
| --- | --- |
| Диспетчер | Человек, имеющий право создавать расписание, редактировать его. |
| Преподаватель | Человек, который занимается преподаванием какого-либо предмета в учебном заведении. |
| Пользователь | Человек, который пользуется данной системой. |
| Аудитория | Помещение, в котором проводится занятие. |
| Студент | Личность, проходящая обучение в университете. |
| Группа | Совокупность студентов, имеющих одинаковое расписание. |

## 1.3 Описание дополнительных спецификаций к проекту

**Функциональные возможности** Система должна обеспечивать многопользовательский режим работы. Если предлагаем курс оказывается заполненным в то время, когда студент формирует свой учебный график, включающий данный курс, то система должна известить его об этом. Система не должна содержать коллизий. В расписании фиксируются время и место проведения занятия, предмет и преподаватель, проводящий занятие, а также номер группы, для которой это занятие проводится.

**Удобство использования** Пользовательский интерфейс должен быть *Windows* *ХР*, 7-10-совместимым.

**Надежность** Система должна быть в работоспособном состоянии 24 часа в день 7 дней в неделю, время простоя – не более 10%.

**Производительность** Система должна поддерживать до 2000 одновременно работающих с центральной базой данных пользователей, и до 500 пользователей, одновременно работающих с локальными серверами.

**Безопасность** Только диспетчер имеет право создавать и вносить изменения в расписание.

Только преподаватель имеет право создавать запрос на перенос занятия.

**Проектные ограничения** Система должна быть интегрирована с существующей системой каталога курсов, функционирующей на основе реляционной СУБД.

# ПРОЕКТИРОВАНИЕ СИСТЕМЫ

## 2.1 Создание диаграммы вариантов использования

Вариант использования представляет собой последовательность действий, выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом.

Созданная модель диаграммы вариантов использования на основе полученного варианта задания представлена на рисунке 1.

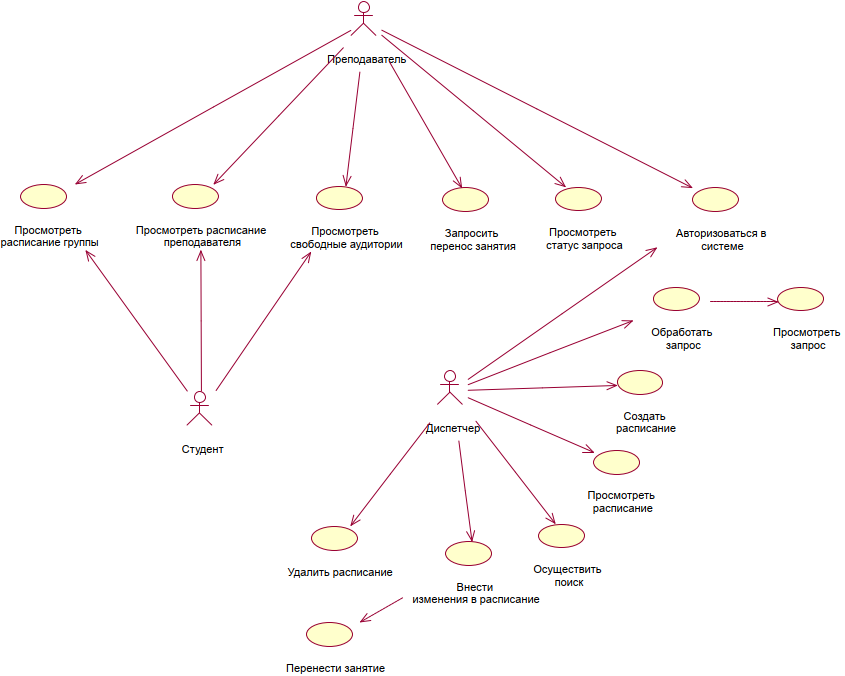


Рисунок 1 – Диаграмма вариантов использования системы

На диаграмме описаны основные требования к системе. Действия выполняются студентом, преподавателем, диспетчером. Программа ожидает событие, на которые будет откликаться тем или иным образом.

Действующие лица:

* Студент – личность, проходящая обучение в университете.
* Преподаватель – личность, которая преподаёт какой-либо предмет в университете.
* Диспетчер –личность, которая занимается составлением расписания, а также внесением в него каких-либо изменений;

Исходя из потребностей действующих лиц, выделяются следующие варианты использования:

* «Просмотреть расписание группы»;
* «Просмотреть расписание преподавателя»;
* «Просмотреть свободные аудитории»;
* «Запросить перенос занятия»;
* «Просмотреть статус запроса»;
* «Авторизоваться в системе»;
* «Просмотреть запрос»;
* «Обработать запрос»;
* «Создать расписание»;
* «Просмотреть расписание»;
* «Осуществить поиск»;
* «Внести изменения в расписание»;
* «Перенести занятие»;
* «Удалить расписание».

## Добавление описаний к вариантам использования

## Вариант использования «Просмотреть расписание группы»:

**Идентификатор и название**

Просмотреть расписание группы.

**Основные действующие лица**

Студент, преподаватель.

**Цель**

Получить расписание группы.

**Нормальные направления развития варианта использования**

Пользователь кликает «Расписание группы»;

Система открывает окно запроса;

Пользователь заполняет поля запроса: номер группы (выбирает нужную группу из предложенного списка), временной промежуток (выбирает необходимый временной промежуток из предложенного списка);

Система проверяет наличие группы в поле для выбора, а также временного промежутка;

Система активирует кнопку «ОК»;

Пользователь подтверждает запрос, нажав кнопку «ОК»;

Система находит по заданному запросу расписание группы в базе данных;

Система выводит на экран найденное расписание.

**Результат**

На экране отображается расписание выбранной группы (дата, день недели, номер учебной недели, смена, номер занятия, время начала и окончания занятия, дисциплина, преподаватель, учебный корпус и номер аудитории, вид занятия (лекция, практическое занятие, лабораторная работа)) на указанный временной промежуток.

**Исключения**

*Система обнаружила пустые поля в запросе.*

Отправление запроса невозможно (кнопка «ОК» неактивна);

Система отображает сообщение об ошибке.

**Вариант использования «Просмотреть расписание преподавателя»:**

**Идентификатор и название**

Просмотреть расписание преподавателя.

**Основные действующие лица**

Студент, преподаватель.

**Цель**

Получить расписание преподавателя.

**Нормальные направления развития варианта использования**

Пользователь кликает «Расписание преподавателя»;

Система открывает окно запроса;

Пользователь заполняет поля запроса: ФИO преподавателя (выбирает нужного преподавателя из предложенного списка), временной промежуток (выбирает необходимый временной промежуток из предложенного списка);

Пользователь кликает «Расписание преподавателя»;

Система открывает окно запроса;

Пользователь заполняет поля запроса: ФИO преподавателя (выбирает нужного преподавателя из предложенного списка), временной промежуток (выбирает необходимый временной промежуток из предложенного списка);

Система проверяет наличие ФИО преподавателя в поле для выбора, а также временного промежутка;

Система активирует кнопку «ОК»;

Пользователь подтверждает запрос, нажав кнопку «ОК»;

Система находит по заданному запросу расписание преподавателя в базе данных;

Система выводит на экран найденное расписание

**Результат**

На экране отображается расписание выбранного преподавателя (дата, день недели, номер учебной недели, смена, номер занятия, время начала и окончания занятия, дисциплина, учебный корпус.

**Вариант использования «Просмотреть свободные аудитории»:**

**Идентификатор и название**

Просмотреть свободные аудитории.

**Основные действующие лица**

Студент, преподаватель, диспетчер.

**Цель**

Получить список свободных аудиторий.

**Нормальные направления развития варианта использования**

Пользователь кликает «Свободные аудитории»;

Система открывает окно запроса;

Пользователь заполняет поля запроса: временной промежуток (выбирает необходимый временной промежуток из предложенного списка);

Система проверяет наличие временного промежутка в поле для выбора;

Система активирует кнопку «ОК»;

Пользователь подтверждает запрос, нажав кнопку «ОК»;

Система находит список свободных аудиторий в базе данных на указанный в запросе временной промежуток;

Система выводит на экран список свободных аудиторий.

**Результат**

На экране отображается дата, день недели, время и список свободных аудиторий на указанный в запросе временной промежуток.

**Исключения**

*Система обнаружила пустые поля в запросе.*

Отправление запроса невозможно (кнопка «ОК» неактивна);

Система отображает сообщение об ошибке.

**Вариант использования «Запросить перенос занятия»:**

**Идентификатор и название**

Запросить перенос занятия

**Основные действующие лица**

Преподаватель.

**Цель**

Создать запрос.

**Нормальные направления развития варианта использования**

Преподаватель кликает «Создать запрос»;

Система открывает окно запроса на перенос занятия;

Преподаватель заполняет поля запроса: дата, дисциплина, вид занятия, поток, курс, номер группы-все поля содержат список возможных вариантов. Преподаватель выбирает из предложенных списков нужный вариант и при необходимости указывает время;

Система проверяет наличие всех заполненных полей;

Система активирует кнопку «Отправить»;

Преподаватель кликает «Отправить»;

На экране появляется сообщение об успешном создании запроса.

**Результат**

На экране отображается сообщение об успешном создании запроса.

Созданный запрос сохраняется в «Мои запросы»

Созданный запрос отображается у диспетчера и находится в стадии обработки.

**Исключения**

*Система обнаружила пустые поля в запросе.*

Отправление запроса невозможно (кнопка «ОК» неактивна);

Система отображает сообщение об ошибке.

**Вариант использования «Просмотреть статус запроса»:**

**Идентификатор и название**

Просмотреть статус запроса.

**Основные действующие лица**

Преподаватель.

**Цель**

Просмотреть статус запроса.

**Нормальные направления развития варианта использования**

Преподаватель кликает «Мои запросы»;

Система отображает созданный раннее запрос преподавателем (дата, дисциплина, вид занятия, поток, курс, номер группы), а также присвоенный ему статус (принят, отклонен, находится в обработке).

**Результат**

На экране отображается созданный раннее запрос (данные, которые ввел преподаватель) с присвоенным ему на текущий момент статусом.

**Исключения**

*Запрос не был создан.*

Система отображает на экране сообщение об отсутствии созданных им запросов.

**Вариант использования «Авторизоваться в системе»:**

**Идентификатор и название**

Авторизоваться в системе.

**Основные действующие лица**

Диспетчер, преподаватель.

**Цель**

Пройти авторизацию для входа в личный кабинет.

**Нормальные направления развития варианта использования**

Пользователь кликает «Авторизация»;

Система открывает форму;

Пользователь вводит логин (обязательное) и пароль (обязательное);

Система активирует кнопку «Войти»;

Пользователь кликает «Войти»;

Система находит логин и пароль данного пользователя в базе данных;

Система выводит на экран сообщение об успешной авторизации;

Система в соответствии с правами доступа авторизованного пользователя предоставляет ему доступ к личному кабинету.

**Результат**

Авторизация пройдена.

Пользователь получил доступ к личному кабинету.

**Исключения**

*Система не нашла в базе данных логин и/или пароль*.

Система отображает на экране сообщение об ошибке.

Система предоставляет возможность изменить логин и/или пароль.

**Вариант использования «Обработать запрос»:**

**Идентификатор и название**

Просмотреть статус запроса.

**Основные действующие лица**

Диспетчер.

**Цель**

Присвоить статус запросу.

**Предварительные условия**

Необходимо выполнение варианта использования. Просмотреть запрос.

**Нормальные направления развития варианта использования**

Система активирует кнопку «Обработать запрос»;

Диспетчер кликает «Обработать запрос»;

Диспетчер присваивает статус запросу (принято либо отклонено);

Система отправляет запрос с поставленным статусом преподавателю.

**Результат**

Статус присвоен;

Запрос обработан;

Результат отправлен преподавателю.

**Вариант использования «Просмотреть запрос»:**

**Идентификатор и название**

Просмотреть запрос.

**Основные действующие лица**

Диспетчер.

**Цель**

Просмотреть запрос.

**Нормальные направления развития варианта использования**

Диспетчер кликает «Запросы»;

Система отображает на экране список отправленных диспетчеру запросов (ФИО преподавателя, время и день отправки запроса);

Диспетчер из предложенного списка выбирает нужный запрос и кликает на него;

Система активирует кнопку «Просмотреть запрос»;

Диспетчер кликает «Просмотреть запрос»;

Система отображает на экране запрос (дата, дисциплина, вид занятия, курс, номер группы), который выбрал диспетчер.

**Результат**

Запрос просмотрен и находится в стадии обработки.

**Исключения**

*Запрос не был создан.*

Система отображает на экране сообщение об отсутствии созданных им запросов.

**Вариант использования «Создать расписание»:**

**Идентификатор и название**

Создать расписание.

**Основные действующие лица**

Диспетчер.

**Цель**

Создать расписание.

**Предварительные условия**

Диспетчер должен пройти авторизацию

Авторизоваться в системе.

**Нормальные направления развития варианта использования**

Диспетчер кликает «Создать расписание»;

Система открывает окно запроса на создание расписания;

Преподаватель заполняет поля запроса: выбирает из предложенного списка номер группы, для которой будет создаваться, а также вводит дату начала и окончания действия созданного расписания;

Система проверяет наличие заполненных полей;

Система активирует кнопку «ОК»;

Диспетчер кликает «ОК»;

Система открывает окно запроса на создание расписания;

Преподаватель заполняет поля запроса: номер группы, для которой будет создаваться расписание (выбирает из предложенного списка), а также вводит дату начала и окончания действия созданного расписания;

Система проверяет наличие заполненных полей;

Система активирует кнопку «Создать»;

Диспетчер кликает «Создать»;

Система проверяет созданное расписание на наличие коллизий;

Система сохраняет расписание в базе данных;

Система выводит на экран сообщение об успешном создании расписания

**Результат**

Расписание создано;

Расписание сохранено в базе данных.

**Исключения**

*Система обнаружила пустые поля в запросе.*

Отправление запроса невозможно (кнопка «ОК» неактивна);

Система отображает сообщение об ошибке.

*Диспетчер не заполнил форму.*

Кнопка «Создать» не активна;

Система отображает сообщение об ошибке.

*Система обнаружила коллизии.*

Система отображает сообщение об ошибке, а также место в котором она допущена;

Система предоставляет возможность изменить данные.

**Вариант использования «Просмотреть расписание»:**

**Идентификатор и название**

Просмотреть расписание.

**Основные действующие лица**

Диспетчер.

**Цель**

Просмотреть расписание.

**Нормальные направления развития варианта использования**

Диспетчер кликает «Расписание»;

Система отображает расписание всех созданных диспетчером групп (номер группы, дисциплина, преподаватель, аудитория);

Система предоставляет диспетчеру форму, в которой дает возможность выбора группы или преподавателя, а также временного промежутка;

Диспетчер при необходимости заполняет форму, в которой из предоставленных списков выбирает номер группы или фамилию преподавателя, а также временной промежуток;

Если поля формы заполнены, система активирует кнопку «Найти»;

Диспетчер кликает «Найти»;

Система группирует расписание по заданному запросу;

Система отображает расписание по заданному запросу.

**Результат**

На экране отображается расписание

**Исключения**

*Система обнаружила пустые поля в форме.*

**Вариант использования «Осуществить поиск»:**

**Идентификатор и название**

Просмотреть запрос.

**Основные действующие лица**

Диспетчер.

**Цель**

Найти необходимую информацию.

**Нормальные направления развития варианта использования**

Диспетчер кликает «Поиск»;

Система выводит на экран форму, в которой диспетчер из предложенных списков выбирает необходимые данные для получения нужной информации;

Система отображает информацию по заданному запросу.

**Результат**

На экране отображается информация по заданному поиску.

**Вариант использования «Внести изменения в расписание»:**

**Идентификатор и название**

Внести изменение в расписание.

**Основные действующие лица**

Диспетчер.

**Цель**

Просмотреть запрос.

**Предварительные условия**

Диспетчер должен просмотреть расписание

Просмотреть расписание.

**Нормальные направления развития варианта использования**

Диспетчер кликает «Редактировать»;

Система предоставляет доступ для внесения изменений в расписание;

Диспетчер изменяет данные в расписании.

Система проверяет внесенные изменения в расписание на наличие коллизий;

Система обновляет базу данных.

**Результат**

Изменения внесены в расписание.

**Исключения**

*Система обнаружила коллизии.*

Система отображает сообщение об ошибке, а также место, в котором она допущена;

Система предоставляет возможность изменить данные.

**Вариант использования «Перенести занятие»:**

**Идентификатор и название**

Перенести занятие.

**Основные действующие лица**

Диспетчер.

**Цель**

Перенести занятие.

**Предварительные условия**

Диспетчер должен просмотреть расписание

Просмотреть расписание.

**Нормальные направления развития варианта использования**

Диспетчер кликает «Перенос занятия»;

Система предоставляет возможность диспетчеру выбрать нужную запись(пару) в расписании, которую необходимо перенести на другой день.

Диспетчер кликает на нужную запись;

Система активирует кнопку «Перенести занятие»;

Диспетчер кликает «Перенести занятие»;

Система предоставляет возможность выбрать диспетчеру в расписании поле для переноса выбранной записи(пары);

Диспетчер находит в расписании необходимое свободное поле и щелчком мыши добавляет в него выбранную запись(пару);

Система обновляет базу данных.

**Результат**

Занятие перенесено.

**Исключения**

*Система не обнаружила выбранной диспетчером записи в запросе.*

Отправление запроса невозможно (кнопка «Перенести занятие» неактивна);

**Вариант использования «Удалить расписание»:**

**Идентификатор и название**

Удалить расписание.

**Основные действующие лица**

Диспетчер.

**Цель**

Удалить расписание.

**Предварительные условия**

Диспетчер должен просмотреть расписание

Просмотреть расписание.

**Нормальные направления развития варианта использования**

Диспетчер кликает «Удалить расписание»;

Система открывает окно с запросом подтверждения удаления диспетчером расписания («Вы уверены, что хотите удалить расписание *номер группы/номера групп*? –да/нет);

Диспетчер кликает «Да»;

Система удаляет расписание из базы данных

**Результат**

Р-1. Расписание удалено.

**Исключения**

Диспетчер кликает «Нет».

Расписание остаётся неизменным (Расписание не удаляется).

## Создание диаграммы взаимодействия

Диаграммы взаимодействия описывают поведение взаимодействующих групп объектов.

Как правило, диаграмма взаимодействия описывает поведение объектов в рамках только одного варианта использования. На такой диаграмме отображаются ряд объектов и те сообщения, которыми они обмениваются между собой.

**2.3.1 Диаграмма последовательности**

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.

Студент кликает «Свободные аудитории». Открывается окно запроса «Поиск свободных аудиторий», в котором студенту предлагается выбрать временной промежуток (условие запроса). После сделанного студентам выбора идет формирование запроса к серверу и передача данных. Сервер обрабатывает запрос и осуществляет поиск в базе данных. После нахождения свободных аудиторий в базе данных, сервер формирует ответ и отправляет результаты поиска. У студента на экране отображается результат поиска.

Для описания процесса просмотра свободных аудиторий была создана диаграмма последовательности, представленная на рисунке 2.

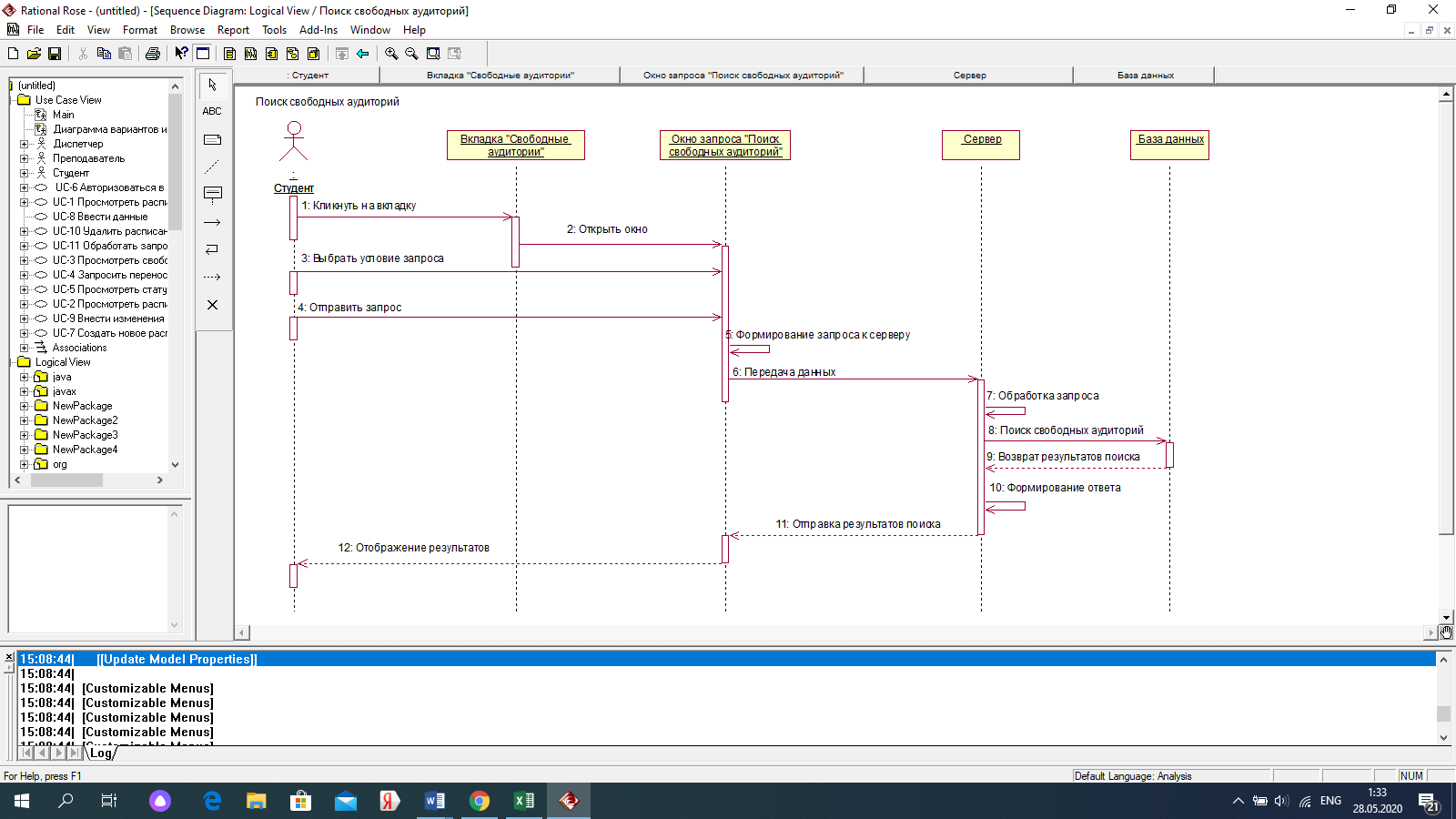


Рисунок 2 – Диаграмма «Просмотреть свободные аудитории»

**2.3.2 Создание кооперативной диаграммы**

Для лучшего представления процесса взаимодействия объектов в процессе внесения изменений в расписание используется кооперативная диаграмма.

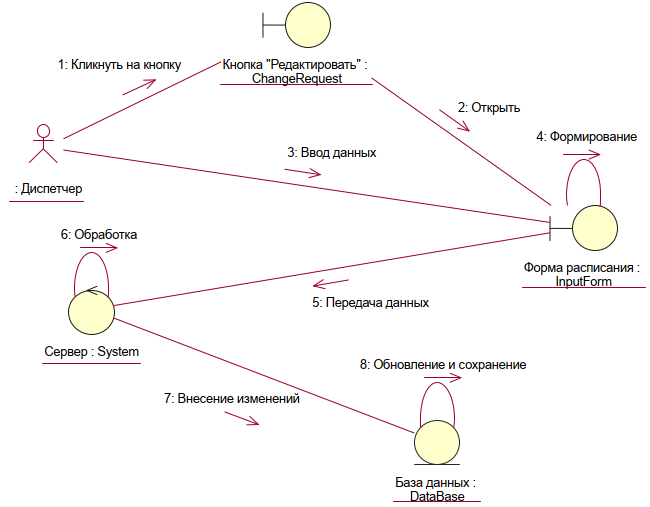


Рисунок 3 – Кооперативная диаграмма «Внести изменения в расписание»

На данной диаграмме легко определить связи взаимодействия в системе. Главное взаимодействие происходит с базой данных. Диспетчер кликает «Редактировать». Открывается форма расписания. Диспетчер вносит изменения в расписание. Расписание формируется и данные передаются серверу. Сервер в свою очередь обрабатывает данные и вносит изменения в базу данных. В конечном итоге обновленное расписание сохраняется в базе данных.

## 2.4 Создание диаграммы классов

Все объекты, участвующие в системе, продемонстрируем с помощью диаграммы классов. Для начала распределим объекты по их типу по папкам.

На рисунке 4 продемонстрировано распределение классов по папкам, в соответствии с их стереотипом.

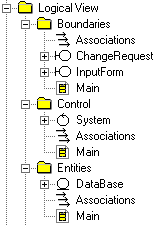


Рисунок 4 – Распределение классов по пакетам

Разделение было сделано на 3 основных типа: границы, сущности и управление. Границы – классы, граничащие с внешней средой, используемые для взаимодействия в основном с пользователями. Сущности – элементы взаимодействия в системе, которые обладают уникальными данными и возможностями. Управление – элементы, используемые для управления взаимодействием между, преимущественно, сущностями.

Главная диаграмма классов создана для демонстрации взаимодействия между пакетами классов системы и представлена на рисунке 5. Активное взаимодействие происходит между классами управления и сущностей. Классы управления корректируют взаимодействие между сущностями.

На рисунке 6 находится основная диаграмма классов.

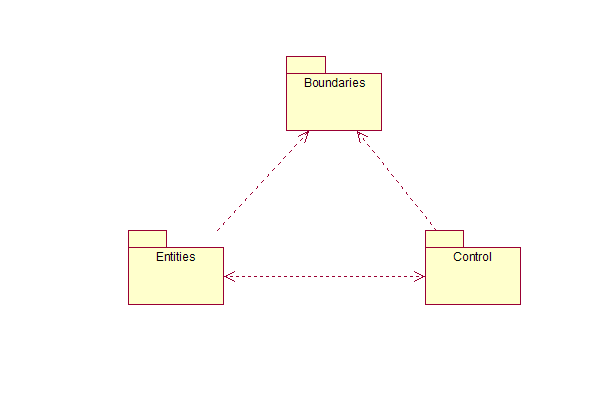


Рисунок 5 – Диаграмма пакетов «Главная диаграмма классов»

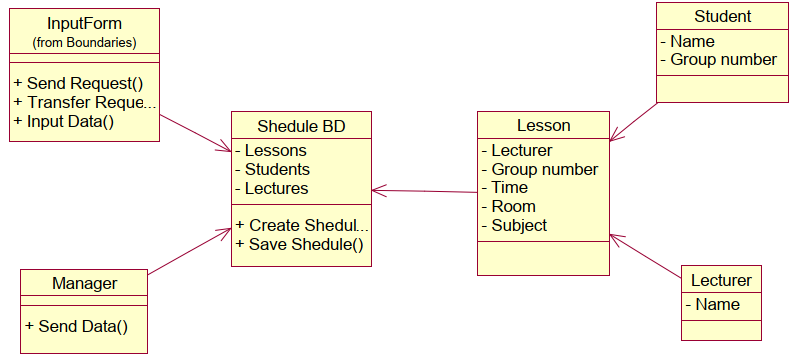


Рисунок 6 – Диаграмма классов

## Создание диаграммы «сущность-связь»

Диаграмма «сущность-связь» (ERD) дает абстрактную модель

предметной области, используя следующие основные понятия: сущности (entities), взаимосвязи (relationships) между сущностями и атрибуты (attributes) для представления свойств сущностей и взаимосвязей.

На рисунке 7 отображена диаграмма «сущность-связь» (ERD).

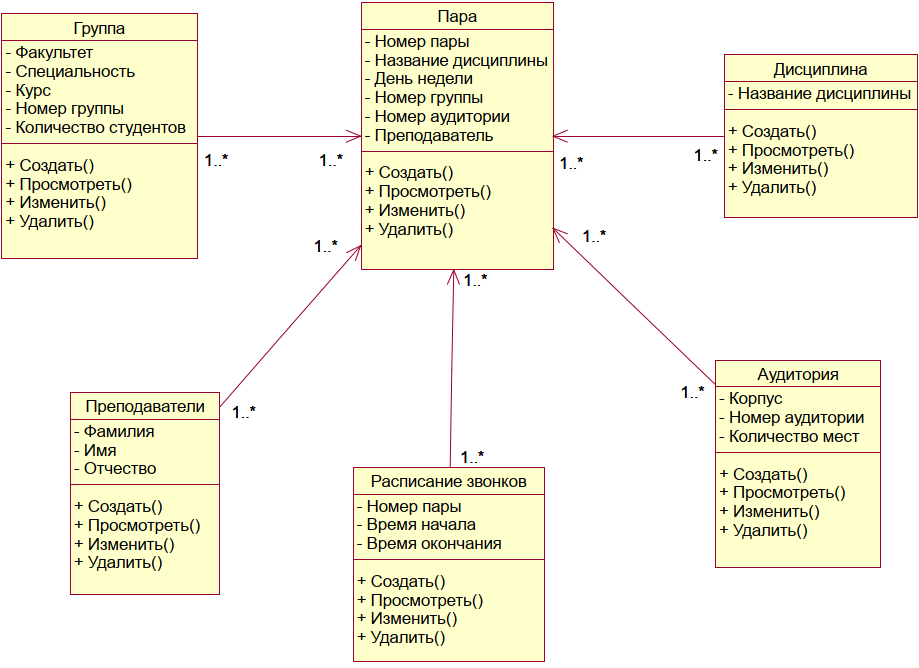


Рисунок 7 – Диаграмма «Сущность-связь»

На данной диаграмме отображены сущности (группа, преподаватели, аудитории и т.д.). Каждая сущность имеет свой набор атрибутов. В данной диаграмме используется связь типа «один-ко-многим».

Связь типа «один-ко-многим» означает, что один экземпляр первой сущности связан с несколькими экземплярами второй сущности.

## 

## 2.6 Создание диаграммы активности

На данной диаграмме будет показан процесс обработки запроса. Диаграмма представлена на рисунке 8.

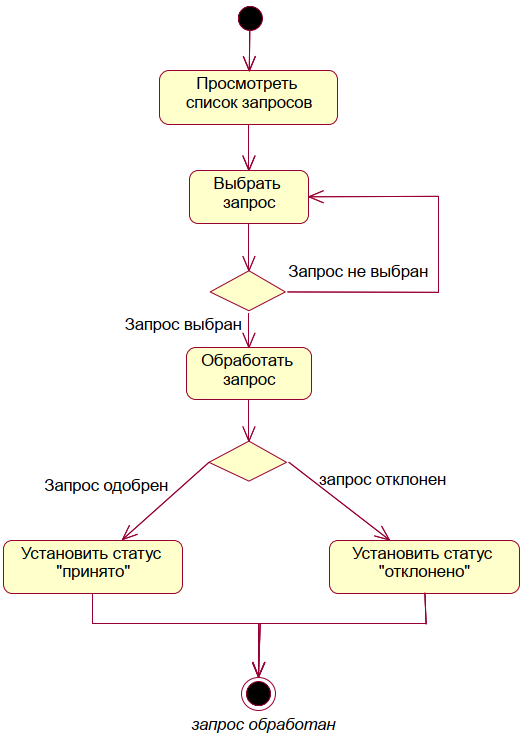


Рисунок 8 – Диаграмма активности

## 2.7 Создание диаграммы базы данных

В программе необходимо постоянно хранить данные о расписании, курсах, преподавателях. Поэтому необходимо создать базу данных с полями, содержащими все необходимые функции.

По условию сказано, что нужно построить модель базы данных программы на диаграмме. На рисунке 9 представлена модель построенной базы данных

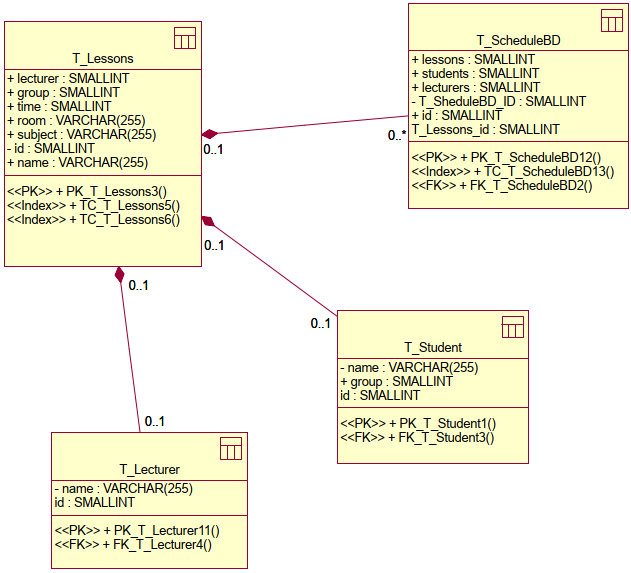


Рисунок 9 – Диаграмма БД

Диаграмма базы данных была сгенерирована с помощью модуля *Data* *Model* в *Rational* *Rose*. Стоит отметить, что подобная функция является весьма удобным инструментом при работе с большими системами. Также система позволяет генерировать *SQL* код. На диаграмме указаны основные связи между таблицами базы данных, первичные и вторичные ключи каждой из них. Также добавлены таблицы, наличие которых необходимо для других таблиц, что показано идентифицирующими связями.

# 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ

## 3.1 Создание диаграммы компонентов

Диаграмма показывает основные спецификации и их взаимодействие. Это необходимо для того, чтобы программисты знали какие элементы системы с чем взаимодействуют. Это позволяет правильно организовать написание кода, тем самым увеличивая продуктивность и уменьшая шанс возникновения ошибки. При этом на такой диаграмме выделяют два типа компонентов: исполняемые компоненты и библиотеки кода.

На рисунке 10 представлена построенная диаграмма компонентов. Диаграмма является начальным этапам реализации любой системы.



Рисунок 10 – Диаграмма компонентов

Диаграмма показывает основные спецификации и их взаимодействие. Это необходимо для того, чтобы программисты знали какие элементы системы с чем взаимодействуют. Это позволяет правильно организовать написание кода, тем самым увеличивая продуктивность и уменьшая шанс возникновения ошибки.

Со всеми спецификациями связаны их тела. В их телах описаны ключевые моменты создания, а также их код написания. В дальнейшем это пригодится при генерации кода, т.к. это будет происходить автоматически.

Ключевую роль в создании программы играет ее ядро и база данных. Поэтому следует заострить внимание на создании этих компонентов. В дальнейшем, когда архитектура программы уже будет готова, ее следует дополнить остальными классами, которые позволят управлять ей. В частности, необходимо спланировать более тщательно разработку интерфейса и взаимодействия с ним.

Из диаграммы видно, что конечная цель сводиться к спецификации запросов. Это можно сказать своеобразное ядро системы. У ядра системы всегда самый высокий приоритет. Его созданием следует заняться в первую очередь. Остальные части системы следует разрабатывать по ходу создания ядра. Также следует отдавать приоритет спецификациям, связанным со спецификацией запросов.

## Создание диаграммы размещения

Диаграммы размещения показывают, как выглядит модель на физическом уровне. На них изображены используемые устройства и связи между ними.

Для отображения устройств, на которых будет размещена система, представлена диаграмма на рисунке 11.



Рисунок 11 – Диаграмма размещения

Имеется три основных устройства. Каждое отдано под отдельную часть общей системы. Сервер приложений используется для реализации основных интерфейсов системы. Сервер базы данных используется для функционирования базы данных системы. На нем будут отображаться все данные расписания университета. Пользовательская часть отображает связь различных девайсов, которые могут использовать студенты и преподаватели для доступа к интерфейсам. К их числу относятся как стационарные рабочие станции, так и мобильные устройства.

## 3.3 Генерация кода C++

В *Rational* *Rose* есть возможность сгенерировать макеты кода. Воспользуемся этим и сгенерируем код для всех классов. Для этого в спецификациях классов укажем, какой код мы хотим сгенерировать. Установим *C++* и папку назначения. В результате сгенерируются файлы с некоторым начальным кодом, от которого мы можем начать написание программы. Все файлы после генерации отображены на рисунке 12.

Всего генерируется два типа файлов: .*cpp* и .*h*, что позволяет сразу же начать написание программы.

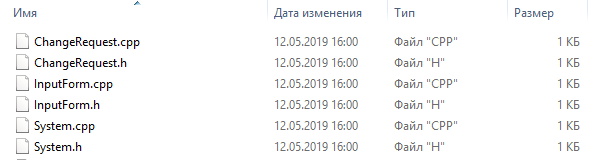
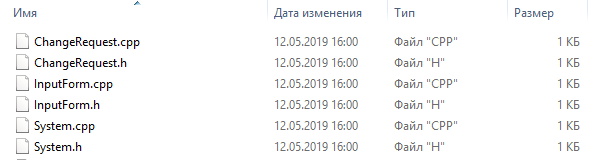


Рисунок 12 – Сгенерированные файлы

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта разработана автоматизированная система поддержки и составления расписания занятий учебного заведения, ставшая основой для написания программы на языке *C++*. Для этого были использованы программа *RationalRose* и язык объектно-ориентированного проектирования *UML*. В процессе работы созданы основные диаграммы языка и описание к диаграммам.

В начале работы проведен анализ поставленной задачи. На основе полученных рекомендаций и условия к заданию был произведен общий расчёт системы. В соответствии с ним нужно было сделать несколько диаграмм взаимодействия, диаграмму компонентов, размещения, активности, состояний, вариантов использования и классов.

Первой построена диаграмма вариантов использования. На ней были показаны основные требования к системе. Эта диаграмма является первым шагом к проектированию системы. Она изображает основные способы использования системы и основных пользователей.

Затем создана диаграмма последовательности, отображающая последовательность выполняемых пользователем действий и возможный ответ системы. На диаграмме показано, как и когда пользователь выполняет действие и какой ответ системы приходит.

На созданной диаграмме классов отображаются все объекты, участвующие в системе. Все объекты были разделены на три типа: границы, сущности и управление, что позволяет детально видеть, что должен выполнять каждый созданный нами класс.

Диаграмма активности показывает процесс «Получить расписание на заданный день». Она показывает, какие действия выполняются в ответ на действия пользователя и каковы зависимости в поведении программы при этом спроектирована ее модель на диаграмме.

По условию необходимо наличие базы данных. Данная модель создана с помощью модуля *Data Modeler* в *Rational* *Rose*. Также он позволяет генерировать *SQL* код, что удобно при начале написания программы на языке программирования.

Важным этапом является создание диаграмма компонентов, которая является начальным этапом реализации любой системы. Она показывает основные спецификации и их взаимодействие. Это необходимо прежде всего для написания кода. Диаграмма компонентов позволяет правильно его организовать, что уменьшает количество и шанс возникновения ошибок.

Диаграмма размещения отображает устройства, на которых будет размещена система.

Сгенерированный *C++* код, создал некоторый каркас для написания всех частей программы. Генерация включает в себя создание файлов с расширениями .*cpp* и .*h*.

В результате была построена система с помощью языка *UML*. В ней были использованы диаграммы различного вида. Это позволило нам детально спроектировать программу, начиная от начальных требований диаграммы вариантов использования и заканчивая генерацией кода.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Г. Буч, Дж. Рамбо, И. Якобсон, Язык UML. Руководство пользователя, 2-е издание, 2013. – 252 с. – ISBN 0-321-26797-4.

[2] У. Боггс, М. Боггс, UML и Rational Rose, 2002. – 307 c.

[3] СТП 01-2017. Стандарт предприятия. Дипломные проекты (работы). Общие требования. - Минск: БГУИР, 2017. - 169 с.

[4] Вендров, А.М. Практикум по проектированию программного обеспечения экономических информационных систем : учебное пособие / А.М. Вендров. – Москва : Финансы и статистика, 2004. – 156 с. : ил. – (UML CASE) . – ISBN 5-279-02440-6.

[5] Х. Гома, UML. Проектирование систем реального времени, распределенных и параллельных приложений, 2-е издание, 2017. – 44 с. – ISBN 978-5-94074-723-9.

[6] К. Ларман, Применение UML и шаблонов проектирования, 2002 – 624 c.

[7] А. Якобсон, Г. Буч, Дж. Рамбо, Унифицированный процесс разработки программного обеспечения, 2002 – 352 c.

[8] R. Pooley, P. Wilcox, Applying UML: Advanced Applications, 2003. – 224 с.

# ПРИЛОЖЕНИЕ А

# (обязательное)

## Листинг кода

**Класс ChangeRequest**

**ChangeRequest.h**

#ifndef CHANGEREQUEST\_H\_HEADER\_INCLUDED\_A327EEDE

#define CHANGEREQUEST\_H\_HEADER\_INCLUDED\_A327EEDE

//##ModelId=5CAE339001AF

class ChangeRequest

{

public:

//##ModelId=5CAE368D02D1

SendRequest();

};

#endif /\* CHANGEREQUEST\_H\_HEADER\_INCLUDED\_A327EEDE \*/

**ChangeRequest.cpp**

#include "ChangeRequest.h"

//##ModelId=5CAE368D02D1

ChangeRequest::SendRequest()

{

}

**Класс InputForm**

**InputForm.h**

#ifndef INPUTFORM\_H\_HEADER\_INCLUDED\_A32781DB

#define INPUTFORM\_H\_HEADER\_INCLUDED\_A32781DB

//##ModelId=5CAE33EA03AC

class InputForm

{

public:

//##ModelId=5CAE36C3011B

TransferRequest();

//##ModelId=5CAE36CC0060

InputData();

};

#endif /\* INPUTFORM\_H\_HEADER\_INCLUDED\_A32781DB \*/

**InputForm.cpp**

#include "InputForm.h"

//##ModelId=5CAE36C3011B

InputForm::TransferRequest()

{

}

//##ModelId=5CAE36CC0060

InputForm::InputData()

{

}

**Класс System**

**System.h**

#ifndef SYSTEM\_H\_HEADER\_INCLUDED\_A327B2C7

#define SYSTEM\_H\_HEADER\_INCLUDED\_A327B2C7

//##ModelId=5CAE340D014C

class System

{

public:

//##ModelId=5CAE36D00145

SendData();

};

#endif /\* SYSTEM\_H\_HEADER\_INCLUDED\_A327B2C7 \*/

**System.cpp**

#include "System.h"

//##ModelId=5CAE36D00145

System::SendData()

{

}

# ВЕДОМОСТЬ КУРСОВОГО ПРОЕКТА

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | *Дополнительные сведения* | | | | | |
|  | | | | | *Текстовые документы* |  | | | | | |
|  | | | | |  |  | | | | | |
| *БГУИР КП 1-53 01 02 06 018 ПЗ* | | | | | *Пояснительная записка* | *38 с.* | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | | *Графические документы* |  | | | | | |
|  | | | | |  |  | | | | | |
| *ГУИР 000000.001 ПД* | | | | | *Диаграмма вариантов* |  | | | | | |
|  | | | | | *использования* | *Формат А1* | | | | | |
|  | | | | |  |  | | | | | |
| *ГУИР 000000.002 ПД* | | | | | *Диаграмма последовательности* | *Формат А1* | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  |  |  |  |  | *БГУИР КП 1-53 01 02 06 018 П3* | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Изм.* | *Л.* | *№ докум.* | *Подп.* | *Дата* | *Система поддержки составления расписания занятий*  *Ведомость курсового проекта* | | *Лит* | | | *Лист* | *Листов* |
| *Разраб.* | | *Шумигай* |  |  |  | *Т* |  | *38* | *38* |
| *Провер.* | | *Ярмолик* |  |  | *Кафедра ИТАС* | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |