СОДЕРЖАНИЕ

[Введение 8](#_Toc127119621)

[1 Анализ и моделирование предметной области программного средства 9](#_Toc127119622)

[1.1 Описание предметной области 9](#_Toc127119623)

[1.2 Разработка функциональной модели предметной области 9](#_Toc127119624)

[1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 14](#_Toc127119625)

[1.4 Разработка информационной модели предметной области 16](#_Toc127119626)

[1.5 UML-модели представления программного средства и их описание 18](#_Toc127119627)

[1.5.1 Диаграмма последовательности 18](#_Toc127119628)

[1.5.2 Диаграмма деятельности 20](#_Toc127119629)

[1.5.3 Диаграмма развертывания 21](#_Toc127119630)

[2 Проектирование и конструирование программного средства 23](#_Toc127119631)

[2.1 Постановка задачи 23](#_Toc127119632)

[2.2 Обоснование выбора компонентов и технологий для реализации программного средства 24](#_Toc127119633)

[2.3 Архитектурные решения 25](#_Toc127119634)

[2.4 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства 28](#_Toc127119635)

[2.5 Проектирование пользовательского интерфейса 30](#_Toc127119636)

[2.6 Методы и средства, используемые для обеспечения безопасности данных 31](#_Toc127119637)

[3 Тестирование и проверка работоспособности программного средства 32](#_Toc127119638)

[4 Руководство по развертыванию и использованию программного средства 36](#_Toc127119639)

[4.1 Методы и средства, используемые для обеспечения безопасности данных 36](#_Toc127119640)

[4.2 Руководство пользователя 36](#_Toc127119641)

[Заключение 54](#_Toc127119642)

[Список использованных источников 55](#_Toc127119643)

[Приложение А (обязательное) Отчет о проверке на заимствование в системе «Антиплагиат» 56](#_Toc127119644)

[Приложение Б (обязательное) Листинг кода алгоритмов, реализующих бизнес-логику 57](#_Toc127119645)

[Приложение В (обязательное) Листинг скрипта генерации базы данных 68](#_Toc127119646)

Ведомость документов курсового проекта…………………………………….72

Перечень условных обозначений, символов и терминов

|  |  |
| --- | --- |
| БД (база данных) | – представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины |
| Информационная система | – система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы (человеческие, технические, финансовые), которые обеспечивают и распространяют информацию |
| Нормальная форма | – свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных |
| Среда выполнения | – вычислительное окружение, необходимое для выполнения компьютерной программы и доступное во время выполнения компьютерной программы |
| СУБД (система управления базами данных) | – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных |
| ТЗ (техническое задание) | – документ, содержащий требования заказчика к объекту разработки, определяющий порядок и условия её проведения |
| *API* (*application programming interface*) | – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой |
| *IDE* (*Integrated development environment*) | – комплекс программных средств, используемый программистами для разработки программного обеспечения |
| *IDEF* | – методология функционального моделирования (англ. *function modeling*) и графическая нотация, предназначенная для формализации и описания бизнес-процессов |
| *Java* | – строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией *Sun* *Microsystems* |
| *SQL* (*structured query language*) | – язык структурированных запросов, декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных |
| *MySQL Server* | – свободная реляционная система управления базами данных |
| *UML* (*Unified Modeling Language*) | – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур |
| *Декомпозиция* | – разделение сложного объекта, системы, задачи на составные части, элементы. Она показывает из каких более мелких работ состоит основной процесс. |
| *ORM (объектно-реляционное отображение)* | – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». |
|  |  |
|  |  |

# Введение

Качество обучения и особенно эффективность использования научно-педагогического потенциала зависят в определенной степени от уровня организации учебного процесса. Одна из основных составляющих этого процесса – расписание занятий – регламентирует трудовой ритм, влияет на творческую отдачу преподавателей, поэтому его можно рассматривать как фактор оптимизации использования ограниченных ресурсов – преподавательского состава и аудиторного фонда.

Проблему составления расписания следует воспринимать не только как трудоемкий процесс, объект автоматизации с использованием ЭВМ, но и как акцию оптимального управления. Поскольку все факторы, влияющие на расписание, практически невозможно учесть, а интересы участников учебного процесса многообразны, задача составления расписания является многокритериальной с нечетким множеством факторов. Решение таких задач, как правило, осуществляется в два этапа: получение оптимального (с точки зрения используемых критериев) варианта и его последующая доработка человеком (диспетчером) с целью максимального учета неформализованных факторов.

Поэтому целью данного курсового проекта будет создание программного средства, способного учитывать различные критерии и на их основе формировать начальное оптимальное расписание, и предоставлять возможность корректировать расписание.

Для достижения поставленной цели необходимо выполнить ряд задач:

* изучить алгоритм, требования, критерии оптимального составления расписания;
* определить список функций, необходимых для эффективной работы программного продукта;
* составить полную схему алгоритма работы программы;
* спроектировать базу данных хранимой информации;
* реализовать серверную часть приложения, которая будет реализовывать бизнес логику, и будет выполнять работу с базой данных;
* реализовать клиентскую часть приложения, с удобным интерфейсом для пользователя;
* протестировать полученное программное средство и убедиться, что оно корректно реализует свою бизнес логику.

Курсовой проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности составляет 97,84%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников». Скриншот приведен в приложении А.

# Анализ и моделирование предметной области программного средства

## Описание предметной области

Cуществующая теория расписаний применима при составлении расписаний работы машин в цехах, и в то же время имеет существенные ограничения по применению для составления расписания занятий в вузе. В настоящей главе предложено решение первого этапа проблемы – разработка алгоритма получения оптимального расписания. Для разработки алгоритма в первую очередь были выделены требования к расписанию занятий. В основу алгоритма была положена идея оценки свободы расположения занятий в расписании, после чего разработана структурная схема алгоритма, и описаны все этапы его работы, а также был предложен вариант повышения качества составленного расписания при помощи генетических алгоритмов.

**Требования к расписанию**

При составлении расписания возникает проблема оптимального управления ресурсами: преподавательским составом и аудиторным фондом. В процессе решения задачи необходимо учитывать обязательные ограничения, а также дополнительные требования, которые могут нарушаться в некоторых случаях. К обязательным ограничениям относятся:

- вместительность аудиторий должна быть достаточной для групп, которые в ней занимаются, при этом возможен вариант, когда в одной аудитории проводятся занятия одновременно для нескольких групп студентов;

- должны выполняться требования занятий к оборудованию аудиторий, в которых они проводятся;

- преподаватели из других вузов могут проводить занятия только в определенные дни и часы.

К дополнительным требованиям относятся:

- лекции должны проводиться в начале дня, практики – в конце;

- нагрузка каждой группы должна быть равномерной, во избежание переутомления студентов, то есть в те дни, когда проводится лекция по сложному предмету, остальные занятия должны проводиться по относительно простым;

- в занятиях студентов не должно появляться окон, в то же время возможно наличие окна в расписании преподавателя;

- по возможности преподавателям должны предоставляться дни, свободные от проведения занятий в вузе;

- в пятницу количество занятий должно быть меньше, чем в остальные дни недели;

- первым занятием в понедельник должен идти относительно простой предмет, иначе успеваемость студентов может существенно снизиться.

**Разработка алгоритма**

В результате анализа требований к расписанию занятий было принято решение о необходимости в разработке алгоритма, в котором были бы заложены возможности по расширению списка требований к расписанию занятий, а также возможности регулирования приоритетов выполнения отдельных требований при составлении расписания. В основе предлагаемого алгоритма составления расписания была положена идея оценки свободы расположения отдельного занятия в полученном расписании. Было установлено, что занятия, для проведения которых требуется выполнение обязательных требований по специальному оборудованию, могут быть проведены только в существенно ограниченном количестве аудиторий, а занятия, которые проводит преподаватель, приходящий только в определенные дни недели, могут быть проведены только в определенные дни недели, а, следовательно, такие занятия имеют меньшую свободу расположения в расписании. Для успешного размещения занятий с относительно малой свободой расположения в расписании решено начинать составление расписания с добавления в него занятий с наименьшими оценками свободы расположения. На рисунке 1.1 приведена структурная схема предлагаемого алгоритма.

Составление расписания занятий начинается с ввода данных об учебной нагрузке. Также для работы алгоритма требуются данные об аудиторном фонде, в случае если они не были введены ранее – производится их ввод. В зависимости от вариантов реализации пользовательского интерфейса списки преподавателей и групп могут формироваться автоматически, на основании данных об учебной нагрузке, и после этого дополняться недостающими данными, либо могут быть введены в ручном режиме.

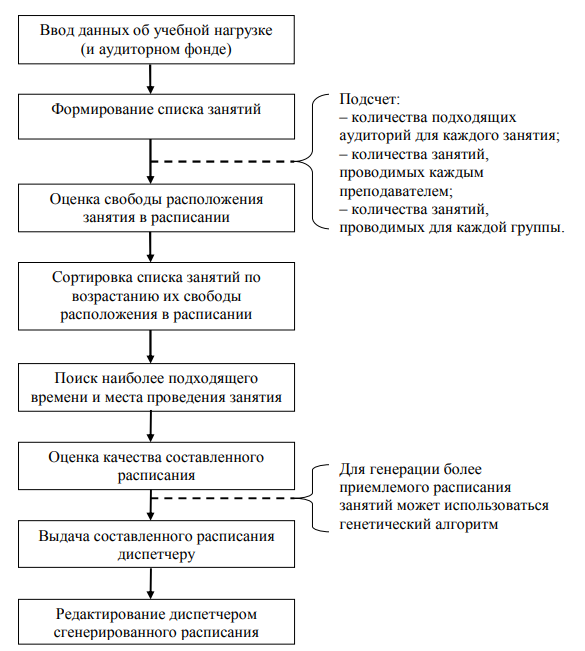


Рисунок 1.1 – структурная схема составления расписания

В результате обработки данных об учебной нагрузке потоков формируется список занятий (см. таблицу 1.1).

Таблица 1.1 – Пример составленного списка занятий

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа | Название дисциплины | Тип занятия | Преподаватель | Дополнительные требования к аудитории |
| 073601 | ВДиШП | Лекция | Логинова И.П. | Нет |

Продолжение таблицы 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа | Название дисциплины | Тип занятия | Преподаватель | Дополнительные требования к аудитории |
| Нет073601 | ВДиШП | Лекция | Логинова И.П. | Нет |
| 073601 | ВДиШП | Практика | Милентьев В.А | Компьютеры |
| 073601 | Физкультура | - | - | Спортивный инвентарь |
| 073603 | УКвЛ | Лекция | Шевченко В.И. | Нет |
| 073603 | УКвЛ | Практика | Шевченко В.И. | Компьютеры |

Как видно из примера, в списке занятий могут встречаться повторения. Это обусловлено тем, что каждое занятие является отдельным объектом, который необходимо расположить в расписании. Дублирование происходит вследствие того, что в течение одной недели проводятся два занятия одинакового типа.

При оценке свободы расположения *i*-го занятия в расписании производится подсчет:

- количества аудиторий *ai*, подходящих для проведения занятия, на основании требований занятия к оборудованию аудитории и количеству рабочих мест;

- количества занятий в неделю *pi*, которые проводит преподаватель этого занятия;

- количества занятий в неделю *gi* для заданной группы студентов.

На основании этих данных определяется оценка свободы расположения занятия в расписании:

где *Si* – оценка свободы расположения *i*-го занятия в расписании;

*n* – количество занятий.

После оценивания свободы расположения занятий в расписании производится сортировка списка занятий по возрастанию их оценок свободы расположения:

, (2)

После проведения сортировки в расписание в первую очередь добавляются занятия, находящиеся в голове списка, то есть с наименьшей свободой расположения в расписании. При добавлении занятий в расписание производится поиск наиболее выгодной аудитории и времени для ее проведения. Для этого необходим полный перебор вариантов проведения занятия в пространстве (аудитории) и времени (номер пары, день недели). В процессе перебора вариантов расположения занятия – в первую очередь происходит проверка возможности проведения занятия по трем условиям:

* + - 1. не происходит «перекрытия» занятий. В случае если оно произошло, то занятие не может быть проведено;
      2. аудитория оборудована всем необходимым для проведения занятия, например компьютерами, стендами для проведения экспериментов проектором и т.д.;
      3. количество рабочих мест в аудитории не меньше количества учащихся в группе.

В случае если обязательные условия выполняются, происходит оценка качества расположения занятия по следующим критериям:

1. появление окна в расписании группы студентов;
2. появление окна в расписании преподавателя;
3. избыточность количества мест в аудитории по отношению к количеству

учащихся;

1. проведение занятия в неудачное время, например четвертым или пятым по счету в этот день для этой группы студентов;
2. исчезновение окна в расписании группы студентов;
3. исчезновение окна в расписании преподавателя;
4. исчезновение окна в расписании использования аудитории.

Каждый из критериев оценки качества расположения занятия в расписании дол жен быть реализован в виде отдельной специализированной функции, возвращающей значения в диапазоне [0;1]. Гибкость и адаптационные способности алгоритма достигнуты за счет возможности дополнять список критериев качества без существенных изменений программного кода. Одним из возможных дополнений может быть проверка необходимости перемещения группы студентов или преподавателя между корпусами вуза для проведения занятия.

Оценка качества расположения занятия по всем критериям может быть использована для получения общей оценки, необходимой в дальнейшем для выбора максимально выгодного времени и места проведения занятия.

Для получения оценки качества расположения занятия в расписании применяется формула вида:

где *Ril* – качество расположения *i*-го занятия на l-й позиции в расписании;

*kjl* – значение, полученное по *j*-му критерию оценки качества расположения занятия на l-й позиции в расписании;

*wj* – весовой коэффициент j-го критерия оценки качества;

*m* – количество критериев оценки качества.

После оценки качества всех возможных вариантов расположения занятия в расписании выбирается вариант, при котором достигается максимальное значение оценки качества расположения:

(4)

где *l* – возможная позиция *i*-го занятия в расписании;

*Ri* – качество расположения *i*-го занятия в расписании;

*h* – количество возможных вариантов расположения занятия в расписании.

После расположения всех занятий в расписании производится оценка качества составленного расписания. Для оценки качества расписания решено использовать сумму оценок качества расположения всех занятий в расписании. Ввиду того, что оценка качества расположения каждого занятия по критериям появления и исчезновения окна в расписаниях студентов и преподавателей зависит от взаимного расположения занятий в расписании – необходимо произвести повторную оценку качества расположения каждого занятия в составленном расписании.

Для оценки качества составленного расписания используется формула вида:

где *R* – качество составленного расписания;

*n* – количество занятий.

Полученные результаты работы алгоритма предоставляются диспетчеру, который решает, стоит ли провести повторную генерацию расписания с новыми настроечными коэффициентами *wj*, либо модифицировать полученное расписание вручную с целью дальнейшего использования.

**Генетический алгоритм**

Для повышения качества полученного расписания занятий, основной алгоритм составления расписания может быть усовершенствован. Ввиду наличия набора настроечных коэффициентов алгоритма, а также критериев оценки качества полученного расписания, возможно использовать генетический алгоритм.

Предлагается использовать в качестве генов настроечные коэффициенты wj алгоритма составления расписания. В результате мутаций будут получены новые наборы настроечных коэффициентов и как следствие – результаты работы алгоритма будут различны. При этом оценку качества составленного расписания следует проводить с использованием значений wj заданных диспетчером. Таким образом, учитывается возможность генерации расписания, наиболее удовлетворяющего оценкам качества с точки зрения диспетчера, при использовании несколько измененных оценок качества в процессе его генерации.

Хромосома такого генетического алгоритма будет представлять собой набор действительных чисел:

(6)

где *w1…wm* – гены хромосомы, настроечные коэффициенты алгоритма составления расписания;

*m* – количество настроечных коэффициентов алгоритма.

Целью работы генетического алгоритма является достижение максимума функционала:

(7)

где *Ril* – качество расположения *i*-го занятия на l-й позиции в расписании;

*kjl* – значение, полученное по *j*-му критерию оценки качества расположения занятия на l-й позиции в расписании;

*wj* – весовой коэффициент *j*-го критерия оценки качества;

*m* – количество критериев оценки качества;

*l* – возможная позиция *i*-го занятия в расписании;

*Ri*– качество расположения *i*-го занятия в расписании;

*h* – количество возможных вариантов расположения занятия в расписании;

*R* – качество составленного расписания.

**Заключение**

Предложенный алгоритм может быть использован для составления расписания занятий в отдельном учреждении образования. Среди преимуществ можно отметить возможность генерации приемлемых вариантов расписания уже с первой итерации. В алгоритме предусмотрена возможность значительного улучшения расписания благодаря добавлению дополнительных критериев оценки свободы и качества расположения занятий в расписании. Например, можно модифицировать алгоритм для составления расписания занятий сразу в нескольких корпусах вуза, добавив при этом дополнительный коэффициент качества расположения занятия, который будет учитывать количество вынужденных перемещений студентов и преподавателей между корпусами – с целью их минимизации. Использование генетического алгоритма в системе составления расписания может значительно улучшить качество составляемого расписания. Реализация алгоритма на начальном этапе подразумевает его работу на локальной ЭВМ с использованием одной из существующих реализаций баз данных с диалоговым интерфейсом взаимодействия.

## Разработка функциональной модели предметной области

Основным бизнес-процессом данного курсового проекта является составление расписания занятий для учреждений образования (рисунок 1.1). Для этого был использован стандарт *IDEF*0 и программное средство *AllFusion* *Process* *Modeler*.

В результате анализа предметной области было получено, что в качестве входных параметров для системы выступает учебная нагрузка и аудиторный фонд. К управляющим воздействиям относятся: кодекс об образовании Республики Беларусь; Cанитарными нормами и правилами «Санитарно-эпидемиологические требования для учреждений высшего образования и учреждений дополнительного образования взрослых»; Cпецифическими санитарно-эпидемиологическими требованиями к содержанию и эксплуатации учреждений образования; Порядком разработки и утверждения учебных планов для реализации содержания образовательных программ. В качестве механизма осуществления главной функции выступает диспетчер, который участвует во всех под-процессах. На выходе процесса мы получаем расписание занятий.

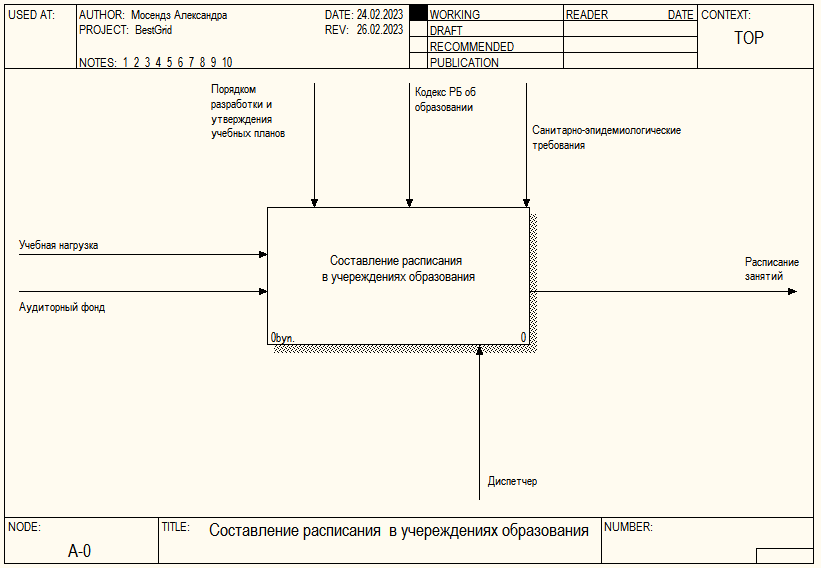


Рисунок 1.2 – Контекстный уровень диаграммы

Далее представлена декомпозиция контекстной диаграммы, состоящая из четырех блоков (рисунок 1.3):

1. Формирование и анализ требований к расписанию;
2. Составление оптимального расписания (опорный план);
3. Оценка качества составленного расписания;
4. Модификация расписания.

Требования к расписанию формируются исходя из требований, утвержденных законодательством, времени работы преподавателей, учебной нагрузки и аудиторного фонда (например: физкультура сразу у 2 групп). Далее учитывая эти требования составляется опорный план (начальное оптимальное расписание), и производится оценка данного расписания (возможно не все требования выполняются, либо необходимо учесть новые требования). Если оценка качества не пройдена, цикл повторяется до тех пор, пока все требования не будут выполнены.

Составление расписания имеет алгоритм, однако по итогу диспетчер может вручную (не по алгоритму) внести изменения в расписание.

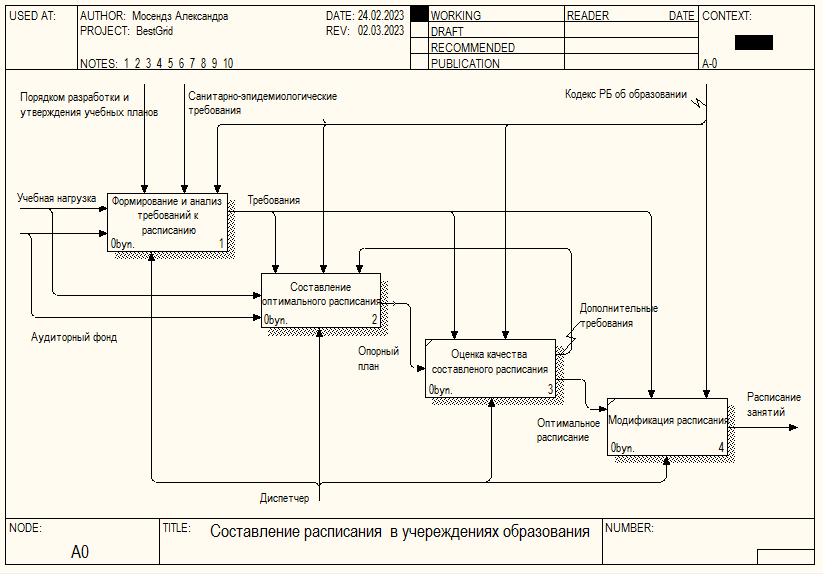


Рисунок 1.3 – Декомпозиция контекстной диаграммы

Этап «Формирование и анализ требований к расписанию» разбит на четыре функциональных блока (рисунок 1.4):

1. Сбор требований;
2. Классификация требований;
3. Разрешение противоречий;
4. Назначение противоречий;
5. Проверка требований.

Сбор требований - это процесс взаимодействия с лицами, формирующими требования. На этапе классификации требований бесформенный набор требований преобразуется в логически связанные группы требований. Разрешение противоречий. Требования многочисленных лиц, занятых в процессе формирования требований, будут противоречивыми. На этом этапе определяются и разрешаются противоречия различного рода. В любом наборе требований одни из них будут более важны, чем другие, поэтому на этапе назначения приоритетов совместно с лицами, формирующими требования, определяются наиболее важные требования. На этапе проверки требований определяется их полнота, последовательность и непротиворечивость.

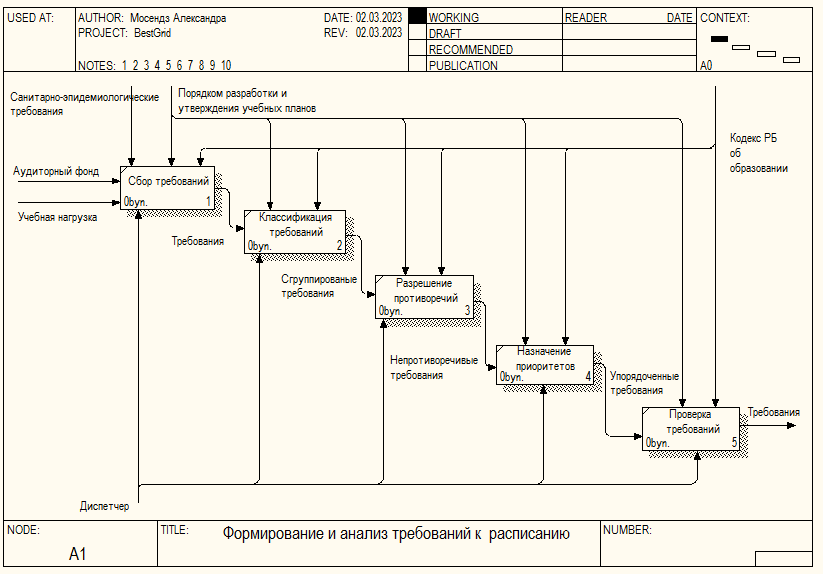


Рисунок 1.4 – Декомпозиция блока «Формирование и анализ требований к расписанию»

Этап «Составление оптимального расписания» разбит на четыре функциональных блока (рисунок 1.5):

1. Формирование списка занятий.
2. Оценка свободы расположения занятий.
3. Сортировка занятий по возрастанию свободы.
4. Поиск наиболее подходящего времени и места проведения занятия.

В списке занятий расписаны все предметы для всех групп (классов), курсов на определенный период (например: для БГУИР это 4 недели, для школ это 1 неделя). Далее определяется степень свободы каждого занятия, она зависит от условий проведения: необходимого оборудования (урок химии может проводиться только в кабинете химии), возможностей преподавателя и так далее. Когда коэффициенты свободы определены необходимо отсортировать список, чтобы вначале разнести по расписанию занятия проведение которых зависит от множества факторов.

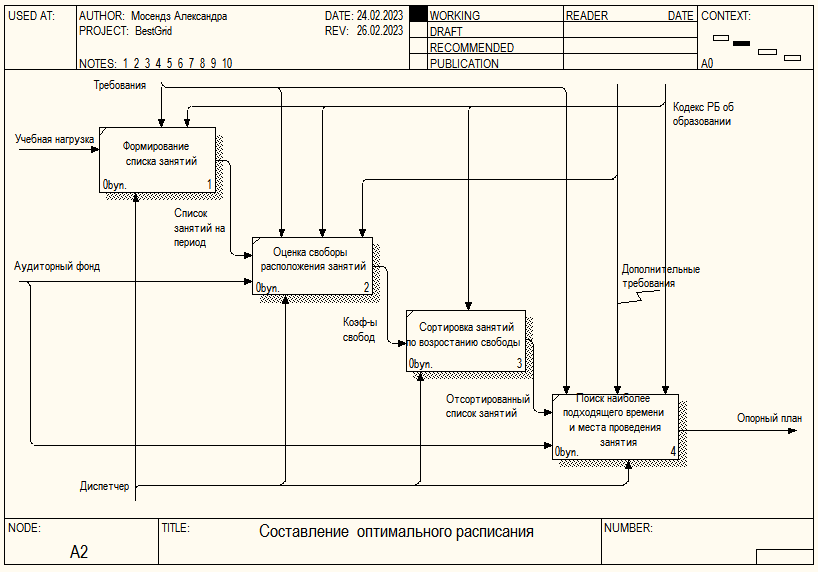


Рисунок 1.5 – Декомпозиция блока «Составление оптимального расписания»

При составлении расписания нужно учитывать множество факторов, если не учесть все факторы, может пострадать учебный процесс (студенты хуже усваиваю материал, преподаватели хуже его дают). Однако данный процесс уже имеет налаженный алгоритм, поэтому его можно автоматизировать, что снизит число ошибок допущенных при учитывании всех факторов.

## Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований (будет дополнена во время разработки)

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. Каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

Все актеры могут зайти под своей ролью и, в зависимости от авторизации, им даются разные возможности (рисунок 1.6).

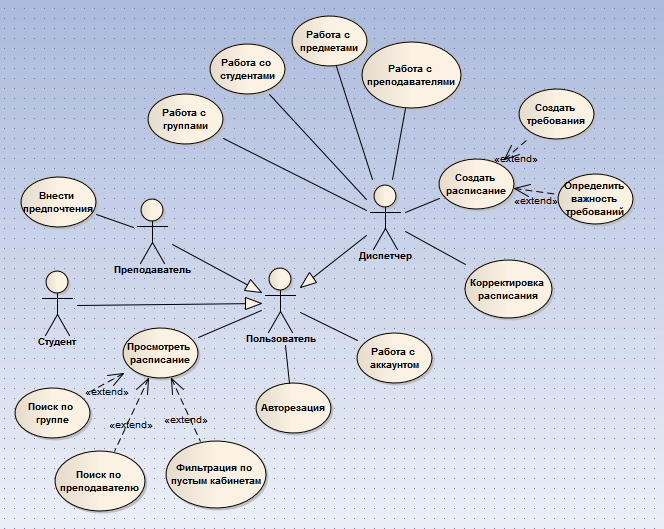


Рисунок 1.6 – Диаграмма вариантов использования

Пользователь не может зарегистрироваться в приложении, все аккаунты создает диспетчер (в том числе аккаунты других диспетчеров). Поэтому при входе в приложение пользователь должен авторизировать (данные для это диспетчер должен предоставить).

Далее все актеры могут просмотреть существующее расписание, а именно осуществлять поиск по группе, преподавателю, или просмотреть список пустых аудиторий.

Диспетчер может создать новое расписание, для этого ему необходимо внести требования к расписанию, и весовые коэффициенты, насколько то или иное требование важно. Так же диспетчер может работать со всеми сущностями из базы данных, создавать, менять, удалять и так далее. После создания расписания диспетчер может внести в него изменения вручную или повторить цикл (меняя веса, требования).

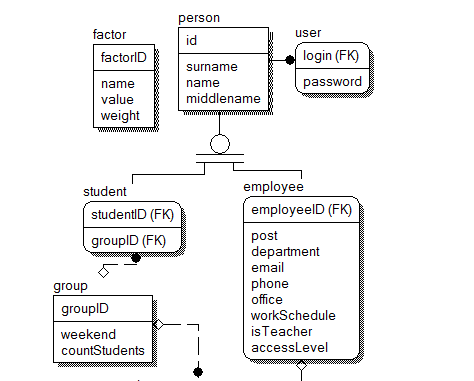
Преподаватель может внести свои предпочтения (время работы, методический день).

## Разработка информационной модели предметной области (будет менятся)

При проектировании системы было принято решение использовать следующие сущности:

* *User*;
* *Person*;
* *Teacher*;
* *Student*;
* *Group*;
* *Factor*;
* *Lesson*.

Графическое отображение информационной модели приведено на диаграмме рисунке 1.7.



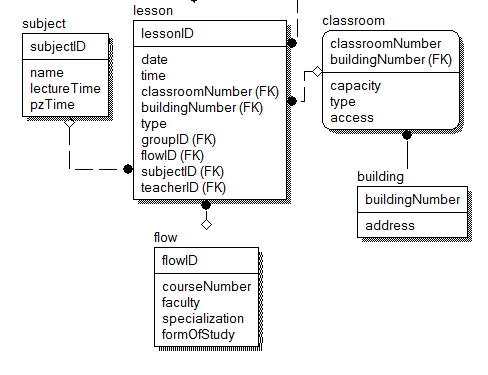


Рисунок 1.7 - Информационная модель системы

Информация о пользователях хранится в сущности «*user*». Данные в нее вносятся, при создании сущности «person», его primary key будет логином пользователя. Для хранения информации об уроках, студентах, сотрудниках, потоках, группах, корпусах и аудиториях есть сущности соответственно *lesson, student, employee, flow, group, building, classroom*. Для составления оптимального расписания, необходимы факторы, оценки, условия и так далее, все это будет объединять сущность *factor*.

Ниже представлено подробное описание всех сущностей, входящих в модель.

Сущность *user* содержит в себе следующие атрибуты:

* *login*– атрибут унаследованный от сущности person;
* *password* –хранит пароль.

Сущность *person* содержит в себе следующие атрибуты:

* *Id*– хранит уникальный идентификатор пользователя;
* *name* –имя пользователя;
* *surname* –фамилия пользователя;
* *middlename*–отчество пользователя;

Сущность *Factor* содержит в себе следующие атрибуты:

* *factorID*– хранит уникальный идентификатор фактора;
* *name* – название фактора;
* *value* – значение фактора;
* *weight*- вес (значимость) фактора.

Сущность *student* содержит в себе следующие атрибуты:

* *sudentID*– атрибут унаследованный от сущности person;
* *GroupId*– id группы.

Сущность *employee* содержит в себе следующие атрибуты:

* *employeeID*– атрибут унаследованный от сущности person;
* *post* –должность преподавателя;
* *department*– отдел или кафедра;
* *email* – адрес электронной почты;
* *phone* – номер телефона;
* *office* – номер кабинета;
* *workShedule*– график работы;
* *isTeacher* –является ли преподавателем;
* *accessLevel* – уровень доступа к системе.

Сущность *group* содержит в себе следующие атрибуты:

* *groupID*– хранит уникальный идентификатор группы;
* *weekend*– выходной для группы (например военная подготовка);
* *countStudents*–количество студентов в группе.

Сущность *building* содержит в себе следующие атрибуты:

* *buildingNumber*– номер корпуса;
* *address*– адрес корпуса.

Сущность *classroom* содержит в себе следующие атрибуты:

* *classroomNumber*– номер аудитории;
* *buildingNumber* – атрибут унаследованный от сущности *building;*
* *capacity*–вместимость аудитории;
* *type*–предназначение кабинета (для лекций, пз и т.д.);
* *access*– доступ к кабинету (например открыть может только определенный сотрудник).

Сущность *flow* содержит в себе следующие атрибуты:

* *flowID*– хранит уникальный идентификатор потока;
* *courseNumber*–номер курса;
* *faculty*–название факультета;
* *specialization*– название специальности;
* *formOfStudy*–форма обучения.

Сущность *lesson* содержит в себе следующие атрибуты:

* *lessonID*– хранит уникальный идентификатор урока;
* *date* –даты проведения;
* *time* –время проведения занятия;
* *classroomNumber*– атрибут унаследованный от сущности *classroom*;
* *buildingNumber*– атрибут унаследованный от сущности *building*;
* *type*–тип проводимого урока;
* *groupID* – атрибут унаследованный от сущности *group*;
* *flowID* – атрибут унаследованный от сущности *flow*;
* *subjectID* – атрибут унаследованный от сущности *subject*;
* *teacherID*– атрибут унаследованный от сущности *employee*.

Любые обычные отношения находятся в первой нормальной форме, значит, разработанная модель находится в первой нормальной форме. Когда отношение находится в первой нормальной форме и нет не ключевых атрибутов, которые зависят от части сложного ключа, тогда отношение находится во второй нормальной форме. Разработанная модель находится в первой нормальной форме, а также нет ключевых атрибутов, являющихся зависимыми от части сложного ключа, следовательно, модель находится во второй нормальной форме. Если все не ключевые атрибуты взаимно независимы и модель находится во второй нормальной форме, то модель находится в третьей нормальной форме. В разработанной модели все не ключевые атрибуты взаимно независимы, а также модель находится во второй нормальной форме, следовательно, модель находится и в третьей нормальной форме.

*SQL*-скрипт для генерации базы данных приведен в приложении B.

## UML-модели представления программного средства и их описание

Язык UML предназначен для описания моделей, причем для работы с этим языком используется специальные редакторы диаграмм. На UML можно содержательно описывать классы, объекты и компоненты в различных предметных областях, часто сильно отличающихся друг от друга.

Диаграммы UML – графическое представление набора элементов, изображенное чаще всего в виде связанного графа с вершинам (сущностями) и ребрами (отношениями).

## Диаграмма последовательности (будет выполнена после разработки программнго средства)

Диаграмма последовательности является одной из разновидности диаграмм взаимодействия и предназначена для моделирования взаимодействия объектов системы во времени, а также обмена сообщениями между ними.

На диаграмме последовательности объекты в основном представляют экземпляры класса или сущности, обладающие поведением. В качестве объектов могут выступать пользователи, инициирующие взаимодействие, классы, обладающие поведением в системе или программные компоненты, а иногда и системы в целом. Данный вид диаграммы отображен на рисунке 1.8

* + 1. Диаграмма деятельности

На диаграмме деятельности представлены переходы потоков управления от одной деятельности к другой внутри системы. Используется при моделировании функционирующей системы т.к. отражает передачу потока управления между объектами. Ее основное назначение – отражение бизнес-процессов объекта. Позволяет показать последовательность процесса, ветвление, синхронизацию процессов. Она позволяет проектировать алгоритм поведения объектов любой сложности, в том числе она может быть использована для составления блок-схем.

Данный процесс начинается, когда пользователь входит в приложение. Вначале Клиентская часть должна загрузить форму для пользователя, в которой он вводил логин и пароль и выбирает роль для входа в систему. После заполнения формы, Клиент считывает введенные данные и отправляет их на Сервер для проверки. Сервер подключается к БД и проверяет существует ли пользователь с таким логином, паролем и ролью, далее отправляет сообщение Клиенту об успешности. Если данные не найдены, пользователю выводится сообщение об ошибке. Если такой пользователь есть в системе Клиентская часть считает роль и откроет соответствующее главное меню.

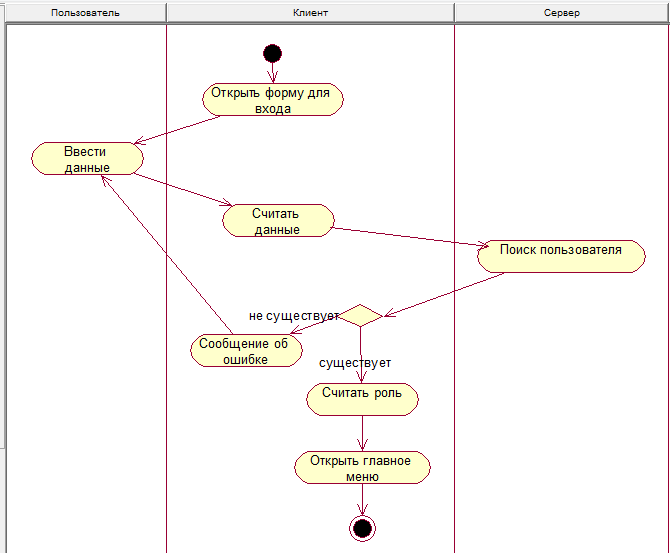


Рисунок 1.9 – Диаграмма деятельности

* + 1. Диаграмма развертывания (будет меняться)

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их.

Диаграмма развёртывания показывает топологию системы и распределение компонентов системы по ее узлам, а также соединения - маршруты передачи информации между аппаратными узлами. Это единственная диаграмма, на которой применяются “трехмерные” обозначения: узлы системы обозначаются кубиками.

Диаграмма развертывания представлена на рисунке 1.10.

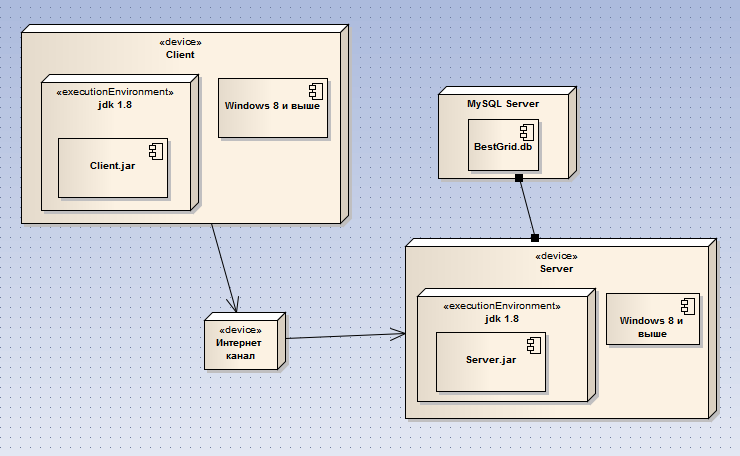


Рисунок 1.10 – Диаграмма развертывания приложения

В качестве узлов выступает ПК пользователя c приложением, сервер и БД.

# Проектирование и конструирование Программного средства

## Постановка задачи

Для разработки программного средства необходимо определить, для кого оно будет создаваться и что необходимо этому кругу лиц. Далее на основе этих данных определить, что нужно реализовать в программном продукте.

Программный продукт может предназначаться как для специалистов (диспетчеры) в крупных учреждениях образования (например БГУИР), так и обычных преподавателей (завучи и директора школ) которые могут плохо разбираться в компьютерных технологиях. Следовательно, данное приложение должно иметь удобный и понятный интерфейс, все вычисления производить самостоятельно, от пользователя запрашивать только информацию (список занятий, график работы и так далее).

К данному приложению так же могут обращаться учащиеся и преподаватели для просмотра готового расписания. Поэтому нужно реализовать возможность вводить и выводить расписание из использования.

При составлении расписания возникает проблема оптимального управления ресурсами: преподавательским составом и аудиторным фондом. В процессе решения задачи необходимо учитывать обязательные ограничения, а также дополнительные требования, которые могут нарушаться в некоторых случаях. Учет всех ограничений влечет за собой множество ошибок (часто ли вы сталкивались с тем, что двум группам назначили занятие в одной аудитории), поэтому процесс составления расписания необходимо автоматизировать.

Так как программное средство направлено на составление расписания для учреждений образования, необходимо спроектировать базу данных для хранения всех необходимых данных, удобный интерфейс для ввода всей информации. Необходимо автоматически (по желанию пользователя) составлять новое расписание основываясь на выбранных, ранее введенных значениях. Пользователь должен иметь возможность обратиться к расписанию в любой момент и отредактировать данные (только для составителей расписания).

Для надежной работы приложения необходимо обеспечить информационную безопасность данных (шифрование, резервное копирование).

## Архитектурные решения

Диаграммы классов используются при моделировании программных средств наиболее часто. На диаграммах классов показываются классы, интерфейсы и отношения между ними. В данном курсовом проекте были реализованы классы

Диаграммы классов показаны на рисунках 2.1 – 2.2.

Рисунок 2.1 - Диаграмма классов

Диаграмма компонентов описывает особенности физического представления системы. Она позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный и исполняемый код. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними. Диаграмму можно увидеть на рисунке 2.3.

Рисунок 2.3 – Диаграмма компонентов

Разделение приложения на составные части, каждая из которых реализует определенную часть функциональности, позволяет повысить структурированность системы, позволяет повторно использовать компоненты; обеспечивает большую гибкость продукта и облегчает его изменение.

## Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства

## Проектирование пользовательского интерфейса

Пользовательский интерфейс – это система средств для взаимодействия пользователя с компьютером, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т.п.). При этом, в отличие от интерфейса командной строки, пользователь имеет произвольный доступ (с помощью клавиатуры или указательного устройства ввода) ко всем видимым экранным объектам, а на экране реализуется модель мира в соответствии с некоторой метафорой и осуществляется прямое манипулирование.

Основное достоинство хорошего интерфейса пользователя заключается в том, что пользователь всегда чувствует, что он управляет программным обеспечением, а не наоборот.

Для создания у пользователя такого ощущения «внутренней свободы» интерфейс должен обладать рядом свойств:

* Естественность интерфейса (не вынуждает пользователя существенно изменять привычные способы решения задач);
* Согласованность интерфейса (позволяет переносить знания на новые задания, для реализации этого свойства, все формы для создания выкроек должны иметь схожее оформления, а именно поля для ввода разделены на три части: дополнительная информация (название, пряжа, плотность), мерки (прилагается картинка), плотность (значения петельной пробы) (рисунок 4.12). Далее во всех формах идет сама выкройка, при нажатии на кнопку рассчитать, на ней отобразятся значения, а снизу появится текстовое описание (рисунок 4.14);
* Дружественность интерфейса (принцип «Прощения пользователя»), для реализации этого свойства, необходимо выводит сообщения при неправильном вводе, не разрешать выполнение действий, пока данные являются не корректными, так же необходимо подтверждать важные действия (например «Вы точно хотите удалить аккаунт»);
* Принцип обратной связи, на все действия пользователя должен приходить ответ, визуальное текстовое сообщение (например об успешности сохранения данных). Если не удалось выполнить действие, нужно выводить причину почему не удалось.
* Простота интерфейса;
* Гибкость интерфейса – способность учитывать уровень подготовки и производительность труда пользователя. Так как приложение создается для использования всеми слоями населения, в нем не должно быть сложной терминологии и все выкройки (чертежи) должны быть интуитивно понятны, а также иметь проработанное описание;
* Эстетическая привлекательность.

## Обоснование выбора компонентов и технологий для реализации программного средства

Программное приложение должно быть написано на объектно-ориентированном языке *Java* и работать в архитектура web-приложения. Web-приложение в данном курсовом проекте является клиент-серверным, клиентом выступает браузер. Архитектура клиент-сервер – взаимодействие двух самостоятельных процессов – клиента и сервера, которые могу выполняться как на одном, так и на разных компьютерах, обмениваясь данными по сети (см. рисунок 2.9). Серверы – процессы, которые реализуют определенную службу (например, службу файловой системы или БД), и принимают запросы. Клиенты – процессы, которые запрашивают службы у серверов с помощью отправки запросов и последующего ответа от него. Таким образом, архитектура «клиент-сервер» позволяет эффективно распределить работу между клиентскими и серверными частями системы: приложения, которые работают на стороне клиента не читают записи базы данных «напрямую», а посылают запросы на сервер, где они обрабатываются, результаты обработки отсылаются назад клиенту, что сокращает потоки информации.

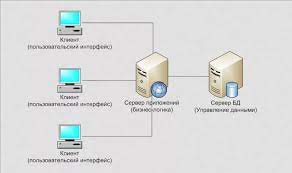


Рисунок 2.9 – Архитектура «клиент-сервер»

Клиент реализует пользовательский интерфейс. Для этого используется веб технология Servlet, jsp, Bootstrap. Расписать их преимущества Клиент отправляет запросы на сервер c помощью http запросов. Сервер обрабатывает запросы клиента и взаимодействует с базой данных используя фреймворк Hibernate, который реализует ORM технологию и позволяет связать (преобразовывать) объектную и реляционную модели. Бизнес логика реализована на основе фреймворка Spring.

Основной средой разработки был выбран *IDE VSCode,* самым главным преимуществом которого является легкость данной *IDE* в сравнении например с *IntelliJ IDEA*. *MySQL Server* используется в качестве СУБД. *Enterprise Architect* используется для разработки и построения *UML*-диаграмм (*Use Case, Statechart, Sequence diagram*, диаграмма классов, *Component diagram, Deployment diagram*). *CASE-*средство *CA AllFusion Process Modeler r*7 (*BPwin*) используется для проектирование *IDEF*0 модели. *CA AllFusion ERwin Data Modeler r*7 (*ERwin*) используется для проектирования информационной модели. *Microsoft Visio* для проектирования блок схем и чертежей*.*

# Заключение

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Береговых Ю.В. Алгоритм составления расписания занятий/ Ю.В. Береговых, Б.А. Васильев, Н.А. Володин, 2009. – 7 с.
2. Галузин К.С. разработка модуля для автоматизации составления оптимального учебного расписания в рамках единой информационной системы образовательного учреждения/ К.С. Галузин, В.Ю. Столбов. - Известия Белорусской инженерной академии, 2003.
3. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. – Горячая Линия - Телеком, 2007. – 452 с.
4. Гладков Л.А. Генетические алгоритмы: учебное пособие / Л.А. Гладков, В.В. Курейчик, В.М. Курейчик. – М. : Физматлит, 2004. – 407 с.
5. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест // МЦНМО. - Москва, 2000. – 960 с.

**Приложение А**(обязательное)  
Отчет о проверке на заимствование в системе «Антиплагиат»

Рисунок А.1 – Отчет о проверке на заимствования в системе «Антиплагиат»

**Приложение Б**(обязательное)  
Листинг кода алгоритмов, реализующих бизнес-логику

// Методы класса Patern. Подсчет параметров для выкроек

**Приложение В**  
(обязательное)  
Листинг скрипта генерации базы данных

-- MySQL Workbench Forward Engineering