Министерство образования и науки Российской Федерации
Филиал Федерального государственного автономного образовательного учреждения
высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
в г. Нижневартовске

Кафедра «Информатика»

РАБОТА ПРОВЕРЕНА

РЕЦЕНЗЕНТ

_____

_____/А.П Суздальцев

«_____»_____

ДОПУСТИТЬ К ЗАЩИТЕ

И.о.зав.кафедрой «Информатика»
к.ф.-м.н., доцент

_____/А.В.Ялаев

«_____»_____

Информационная система для автоматизации ведения и автоматического заполнения расписания колледжа

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ-09.03.01.2018.360.ПЗ ВКР

Консультанты
Экономическая часть
к.э.н., доцент

_____/А.В.Прокопьев

«_____»_____

Безопасность жизнедеятельности
к.ф.-м.н., доцент

_____/ А.В.Ялаев

«_____»_____

Руководитель работы
к.т.н., доцент

_____/Д.В.Топольский

«_____»_____

Автор работы
обучающийся группы НвФл-526

_____/ Д.О.Шишков

«_____»_____

Нормоконтролер
старший преподаватель

_____/ Л.Н.Буйлушкина

«_____»_____

Нижневартовск 2018

АННОТАЦИЯ

Шишков Д.О. Информационная система для автоматизации ведения и автоматического заполнения расписания колледжа - Нижневартовск: филиал ЮУрГУ, Информатика: 2018, 79 с., 12 ил, 3 табл., библиогр. список – 20 наим., 3 прил.

Целью выпускной квалификационной работы является разработка информационной системы для автоматизации ведения и автоматического заполнения расписания колледжа на основе справочных данных и получения удалённого доступа к расписанию для студента.

Объектом выпускной квалификационной работы является процесс обработки информации при составлении расписания занятий в БУ «Нижневартовский социально-гуманитарный колледж».

Предметом выпускной квалификационной работы является ручной формат составления расписания занятий, а также бумажный формат расписания, доступный для просмотра студентами.

Изучена предметная область составления расписания. Выполнен расчет экономической эффективности от внедрения. Предоставлены рекомендации и требования по обеспечению безопасности жизнедеятельности. Проведен литературный обзор.

					09.03.01.2018.360.ПЗ		
Изм	Лист	№ докум.	Подпись	Дата			
Разработал		Шишков Д.О			Информационная система для автоматизации ведения и автоматического заполнения расписания колледжа	Лит.	Лист 5 / Листов 79
Проверил		Топольский Д.В.				20	
Рецензент		Суздальцев А.П.				Филиал ФГАОУ ВО «ЮУрГУ (НИУ)» в г. Нижневартовске кафедра «Информатика»	
Н.контр.		Буйлушкина Л.Н.					
Утвердил		Ялаев А.В.					

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире каждое предприятие старается автоматизировать процесс работы с клиентами. Каждый день предприятия сталкиваются с компьютеризацией различных систем. Они во многом улучшают нашу жизнь, освобождают время, упрощают нашу работу, и облегчают наш труд. Важным ресурсом в современном мире является информация. Работа с потоками не структурированной информации ведёт к увеличению затрат времени и ресурсов на обработку этой информации. Поэтому рациональное хранение, преумножение и использование информации является особенно актуальной темой.

Целью выпускной квалификационной работы является разработка информационной системы для автоматизации ведения и автоматического заполнения расписания колледжа на основе справочных данных и получения удалённого доступа к расписанию для студента.

Задачи:

1. анализ предметной области;
2. определение функциональных требований;
3. анализ существующих разработок;
4. выбор средства реализации;
5. разработка базы данных;
6. разработка приложения
7. оценка экономической эффективности;
8. организация условий труда.

Объектом выпускной квалификационной работы является процесс обработки информации при составлении расписания занятий в БУ «Нижневартовский социально-гуманитарный колледж».

Предметом выпускной квалификационной работы является ручной формат составления расписания занятий, а также бумажный формат расписания, доступный для просмотра студентами.

Разработанное web-приложение позволяет оперативно обрабатывать большой объем информации, безошибочно формировать расписание, уменьшить время на поиск свободного кабинета.

1 АНАЛИТИЧЕСКИЙ РАЗДЕЛ

1.1 Анализ предметной области

Диспетчер составляет расписание, основываясь на учебных планах, полученных от учебной части.

Учебные планы имеют следующие параметры:

- дисциплина;
- количество часов лекционных занятий;
- количество часов практических занятий;
- количество часов лабораторных занятий;
- семестр проведения занятий;
- группа.

Так же диспетчеру от учебной части поступают сведения, о том, какой преподаватель ведет ту или иную дисциплину у конкретной группы.

Некоторые дисциплины имеют свою специфику, в зависимости от которой её можно проводить в том или ином кабинете. Например, для некоторых дисциплин необходимы специализированные кабинеты. Но в специализированных кабинетах можно проводить лекции по разным дисциплинам, которым не требуется специфика кабинета, если тот в свою очередь не занят. Другими словами, диспетчеру, при составлении расписания необходимо не только учитывать свободные кабинеты, но и составлять приоритет для назначения занятий в определённое время.

При составлении расписания диспетчер должен учитывать:

- нагрузка на группы в течение недели не должна превышать 36 часов;
- в расписании у преподавателя в одном промежутке времени должно быть только одно занятие;
- в одном промежутке времени в одном кабинете может проводиться только одно занятие;
- в семестр должна выполниться норма вычитки часов.

Ручное решение задачи составления расписания занятий требует больших затрат времени, квалифицированных специалистов, в то же время результат такого решения часто получается далеко не оптимальным. После ввода исходной информации требуется её согласование, в то время как невозможность получения требуемого расписания может быть определена ещё на этапе анализа. Во время составления расписания возможно возникновение тупиковых ситуаций. Всё это требует изменения исходных данных и ослабления ограничений, и здесь без человека не обойтись. Без внесения данных изменений расписание не будет иметь практической ценности. Также следует учесть тот момент, что расписание может меняться и во время его использования, т.е. после составления, и здесь весьма важен человеческий фактор. В этом плане важна поддержка данного процесса автоматизированными методами и процедурами. Основное преимущество состоит в том, что автоматизированное составление устраняет массу рутинной работы, такой как: поиск возможных вариантов внесения очередных элементов в расписание, проверку выполнения требований, поиск случайных ошибок в готовом расписании, оформление расписания на бумаге в виде различных таблиц.

Для решения существующих проблем требуется построение гибкой и легко адаптируемой системы на основе новых принципов, с использованием современных web-технологий.

Функциональные требования к разрабатываемой системе:

При работе с системой диспетчер должен иметь возможность решать следующие задачи:

- автоматически формировать расписание на основе имеющихся данных;
- производить ручную корректировку расписания, с учетом возможных вариантов и ограничений;
- выводить расписание на печать.

Студенты колледжа должны иметь возможность получать информацию о своём расписании, так же должны иметь возможность найти интересующего их преподавателя.

При работе с системой диспетчер должен иметь возможность:

- доступ через Интернет к справочникам системы;
- информация о преподавателях;
- поиск расписания для определенной группы.

1.2 Анализ существующих аналогов разработок

В процессе проектирования приложения были проанализированы разрабатываемые и уже внедренные системы, связанные с автоматизацией составления расписания.

Для анализа были выбраны наиболее популярные решения.

Например, «TimeTableTechnologes» предлагает программу под названием «Колледж» и «Колледж+» с возможностью редактирования сетки расписания, поиску определённых подгрупп до целевой группы, организацией замен в расписании, создание неограниченного количества подгрупп в рамках любого предмета, с указанием списочного состава подгруппы, экран для формирования списочного состава подгрупп в процессе составления расписания. Автоматическая минимизация окон учащихся, путем переноса из одной подгруппы в другую. Гибкое задание рабочих и нерабочих дней недели, включая воскресенье.

Плюсы программ «Колледж» и «Колледж+» это реализованный стандартный функционал для составления расписания. Так же к плюсам можно отнести, цену продукта от 9 000 до 12000 рублей.

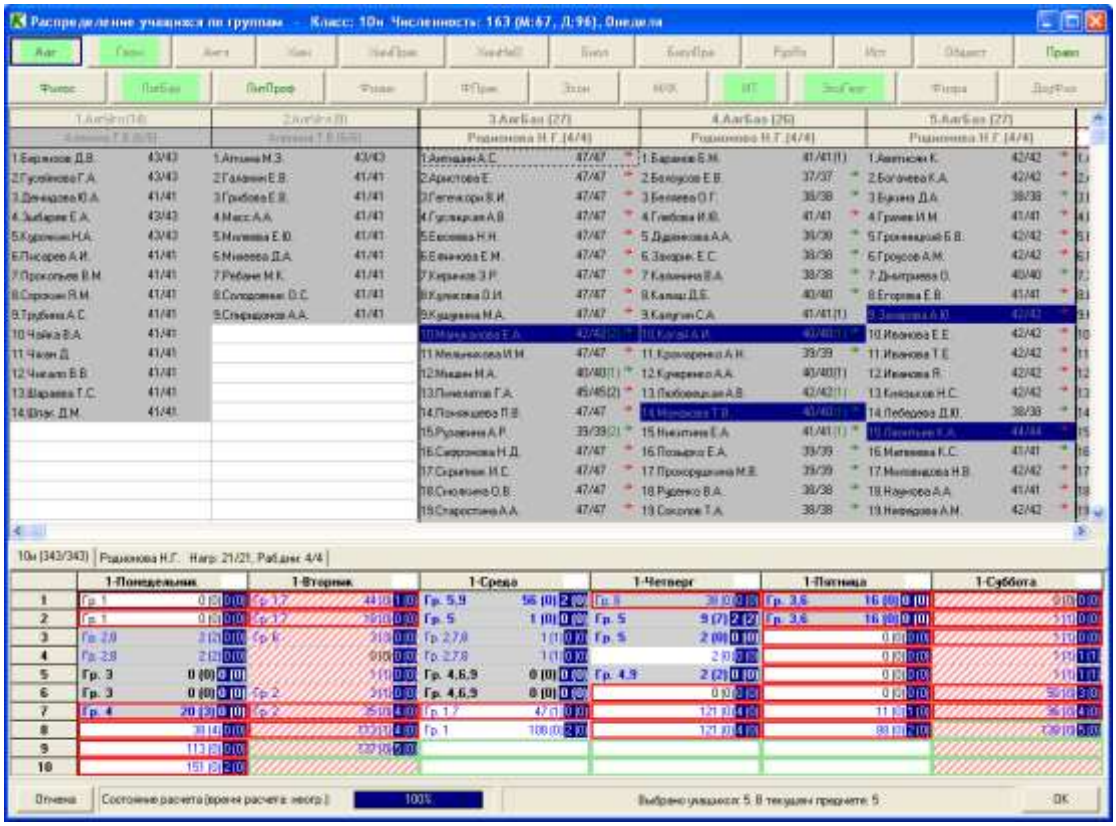

Рисунок 1.1 – Интерфейс программы «Колледж+»

«Лаборатория ММИС» предлагает своё программное решение «Авторасписание M College». Стоимость одного рабочего места диспетчера составляет 40000 рублей. Реализован стандартный функционал. Позволяет составлять семестровые расписания (по неделям и датам), в том числе со сложным графиком учебного процесса; может учитывать выездные практики, разовые мероприятия, праздничные дни; строить расписания занятий и сессий всех форм обучения; находить различные виды замен преподавателей и вести учет вычитанной нагрузки; учитывать переходы между учебными корпусами; учитывать занятость преподавателей и аудиторий в головном ВУЗе, если колледж входит как подразделение в состав ВУЗа, использующего программное обеспечение AVTOR.

Все перечисленные аналоги реализуют стандартный функционал для автоматического формирования расписания, стандартные формы можно настроить. В рассмотренных решениях отсутствует поддержка других операционных систем, кроме Windows, что обязует диспетчера пользоваться определённой операционной системой. Так же отсутствует возможность программных доработок.

Самостоятельная разработка программы как web-приложения с адаптивной вёрсткой решает проблему программы привязанности к операционной системе. В системе можно работать как на компьютере, так и на смартфоне. Стоимость затрат не высока. Полноценная поддержка и сопровождение без привлечения сторонних специалистов, возможность включения в проект студентов, возможность на этапе разработки проведения пробного внедрения и корректировки результатов в ходе тестирования, а также гибкость среды разработки, позволяющая дорабатывать систему в процессе работы.

1.3 Требования к разрабатываемому приложению

Целью данной выпускной квалификационной работы является разработка, приложения позволяющего автоматически формировать расписание и получить удалённый доступ к расписанию занятий в колледже.

Проанализировав предметную область и оценив ресурсы предприятия, было принято решение разработать web-приложение. На сегодняшний день технологии позволяют создавать легко масштабируемые web-приложения высокой сложности с различным функционалом.

Основные задачи приложения:

- сокращение временных затрат на поиск преподавателя студентом;
- просмотр расписания занятий;
- заполнения планов занятий и справочной информации;
- автоматическое формирование расписания для диспетчера;
- обеспечить доступ через сеть Интернет к информационной системе;

Проанализировав предметную область и оценив ресурсы предприятия, было принято решение разработать web-приложение. На сегодняшний день технологии позволяют создавать легко масштабируемые web-приложения высокой сложности с различным функционалом.

Достоинства web-приложений:

– программный код web-приложений выполняется на удаленном сервере, не используя ресурсов машины пользователя;

– возможность использования практически любого компьютера, на котором установлен современный web-браузер;

– отсутствие привязанности к аппаратному и программному обеспечению, полная кроссплатформенность;

– отсутствие проблем с обновлением и поддержкой старых версий программ;

– стабильность в хранении результатов тестирования и остальных данных.

Безусловно, у web-приложений есть существенные недостатки:

– необходимость подключения пользователя к сети (локальной или глобальной) для доступа к серверу;

– скорость работы приложения зависит от скорости передачи данных между клиентом и сервером и загруженности сервера, которая тем выше, чем больше пользователей одновременно обращаются к серверу.

Сегодня web-приложения широко используются на предприятиях с развитой корпоративной сетью, так как сопровождение и обновление таких приложений обходится гораздо дешевле и не требует больших временных затрат.

Разработанное web-приложение будет осуществлять функции связующего звена между пользователем и базой данных (далее – БД).

Хранение данных в БД на сервере обусловлено следующими причинами:

– централизованное хранение на сервере более надёжно по сравнению с хранением на локальных машинах: к серверу ограничен как физический, так и программный доступ, постоянно выполняется резервное копирование данных;

– реляционная структура БД обеспечивает более быстрый доступ к связанным данным;

– исключается нежелательное дублирование данных;

– возможность выбирать только те данные, которые необходимы в данный момент.

Для web-приложения предполагается установка сервера. На первом этапе можно ограничиться типовым ПК с предполагаемой заменой его в будущем на полноценный сервер (при необходимости). Требуемая мощность определится при эксплуатации расписания и далее не рассматривается. Для начального этапа выбираем следующую конфигурацию сервера, приведенную в таблице 1.1.

Таблица 1.1 – Конфигурация сервера на начальном этапе

№	Наименование компонента	Кол-во
1	Процессор семейства Celeron Gxxxx или AMD Athlon	1
2	RAID массив уровня 1 из двух жестких дисков по 100 GB	1
3	Модуль памяти 1024 MB DDR3	1
4	Материнская плата начального уровня с интегрированным контролерам LAN 1GB и встроенным графическим контролером	1
5	Клавиатура, мышь, любой монитор	1
6	Корпус с блоком питания FSP 300W ATX2.0	1

На выбранном сервере необходимо установить сервер web-приложений, интерпретатор и драйвер БД. Если предполагается доступ к web-приложению вне локальной вычислительной сети (далее – ЛВС) предприятия (например, из Интернет), то необходимо также установить межсетевой экран.

В таком случае администратору сети необходимо настроить соответствующим образом DNS-сервер для правильной переадресации внешних запросов к серверу информационной системы.

Выводы по разделу один:

В данном разделе проанализирована предметная область, выделены основные проблемы и задачи, которые предлагается решить с помощью разработанного web-приложения. Определены требования, как к самому web-приложению, так и к программно-техническим средствам для корректной работы сервиса.

2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА

На этапе анализа предметной области были выделены основные задачи, которые должно решать приложение, и определены технические требования. Однако для начала практической реализации работы этой информации недостаточно, так как мы не имеем детального представления о компонентах, из которых будет состоять приложение и механизмы их взаимодействия. Моделирование предметной области облегчает выбор инструментов и технологий, которые будут применяться для реализации программы.

2.1 Общая архитектура приложения

В основе разрабатываемой системы лежит архитектура MVC.

Модель (Model)

Модель – это данные и правила, которые используются для работы с данными, которые представляют концепцию управления приложением. В любом приложении вся структура моделируется как данные, которые обрабатываются определённым образом. Только данные, которые должны быть обработаны в соответствии с правилами (дата не может указывать в будущее, e-mail должен быть в определённом формате, имя не может быть длиннее X символов, и так далее).

Модель даёт контроллеру представление данных, которые запросил пользователь (сообщение, страницу книги, фотоальбом, и тому подобное). Модель данных будет одинаковой, вне зависимости от того, как мы хотим представлять их пользователю. Поэтому мы выбираем любой доступный вид для отображения данных.

Модель содержит наиболее важную часть логики приложения, логики, которая решает задачу, с которой мы имеем дело (форум, магазин, банк, и тому подобное). Контроллер содержит в основном организационную логику для самого приложения (очень похоже на ведение домашнего хозяйства).

Представление (View)

Представление обеспечивает различные способы представления данных, которые получены из модели. Он может быть шаблоном, который заполняется данными. Может быть несколько различных видов, и контроллер выбирает, какой подходит наилучшим образом для текущей ситуации.

Web-приложение, обычно состоит из набора контроллеров, моделей и представлений. Контроллер может быть устроен как основной, который получает все запросы и вызывает другие контроллеры для выполнения действий в зависимости от ситуации. [2]

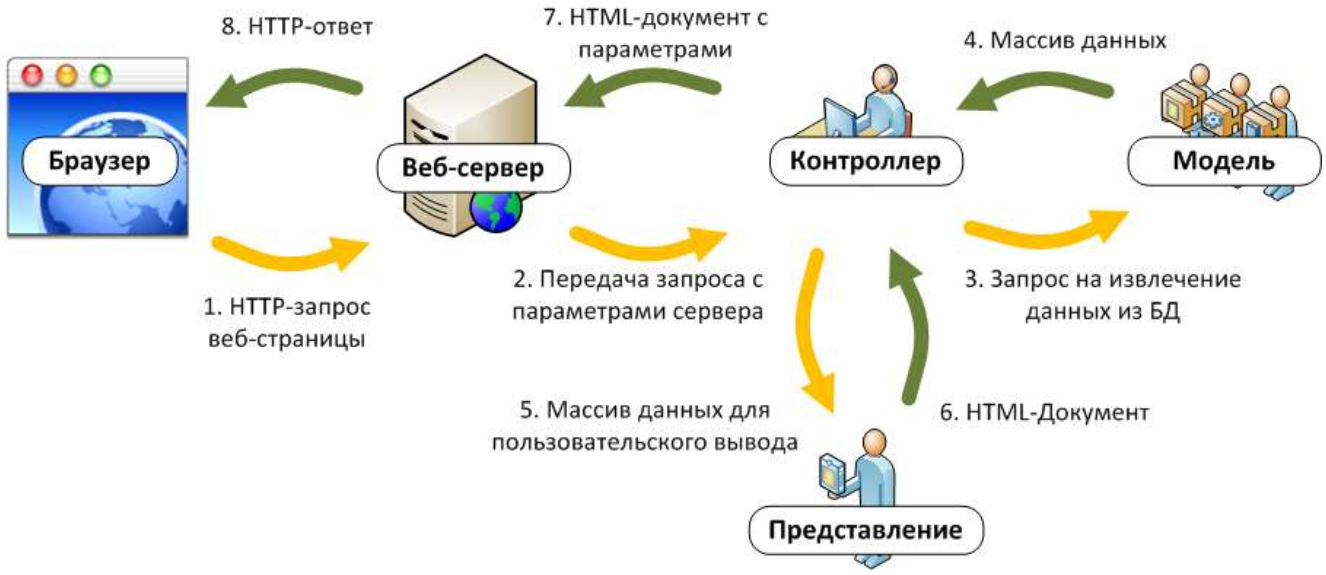

Рисунок 2.1 – MVC архитектура

Рассмотрим MVC с технической стороны:

– запрос пользователя на просмотр страницы, посредством ввода url-адреса;

– приложение соотносит URL с маршрутом (route);

– вызывается controller action, связанный с маршрутом;

– controller action использует модели (models) для получения всех необходимых данных из БД, помещает данные в массив, и передает представлению (view);

– представление получает доступ к данным и использует их, для отрисовки запрошенной страницы, которая затем показывается пользователю в браузере.

2.2 Проектирование БД

Изучив предметную область и спроектировав структуру приложения, можно приступить к следующему шагу – проектированию структуры БД. В качестве метода проектирования БД выбран метод ER-диаграмм. Согласно данному методу в первую очередь определим основные сущности. Названия и атрибуты сущностей, приведённые в списке ниже, могут отличаться от названий таблиц и полей в БД, так как при создании запросов на языке SQL удобнее использовать наименования, написанные латинскими буквами.

Основные сущности с указанием ключевого атрибута:

1. Кабинеты (id_кабинета).
2. Кафедры (id_кафедры).
3. Специфики (id_специфики).
4. Группы (id_группы).
5. Направления групп (id_направления).
6. Семестры (id_семестра).
7. Дисциплины (id_дисциплины).
8. Учебные планы (id_плана).
9. Расписание времени проведения занятий (id_время).
10. Преподаватели (id_преподавателя).
11. Квалификации (id_квалификации).
12. Виды дисциплин (id_вида_дисциплин).
13. Расписание (id_расписания).

Также имеются следующая сущность, образованная в результате связывания нескольких сущностей из списка выше:

Добавим не ключевые атрибуты в сущности:

1. Кабинеты (id_кабинета, наименование_кабинета, id_специфики).
2. Кафедры (id_кафедры, наименование_кафедры).
3. Специфики (id_специфики, наименование_специфики).
4. Группы (id_группы, наименование_группы, номер_курса, id_направления).
5. Направления групп (id_направления, наименование_направления).
6. Семестры (id_семестра, номер_семестра, дата_начала, дата_окончания).
7. Дисциплины (id_дисциплины, наименование_дисциплины, id_кафедры).
8. Учебные планы (id_плана, часы, id_группы, id_дисциплины, id_преподавателя, id_вида_дисциплин).
9. Расписание времени проведения занятий (id_время, номер, время_начала, время_окончания).
10. Преподаватели (id_преподавателя, ФИО, общий_опыт, местный_опыт, id_кафедры, недельная_нагрузка, id_дисциплины, id_квалификации).
11. Квалификации (id_квалификации, наименование_квалификации).
12. Виды дисциплин (id_вида_дисциплин, наименование_вида_дисциплин).
13. Расписание (id_расписания, день, это_четная_неделя, id_кабинета, id_группы, id_дисциплины, id_время, id_преподавателя, id_вида_дисциплин, id_семестра).

На Рисунке 2.2 представлена логическая модель спроектированной БД.

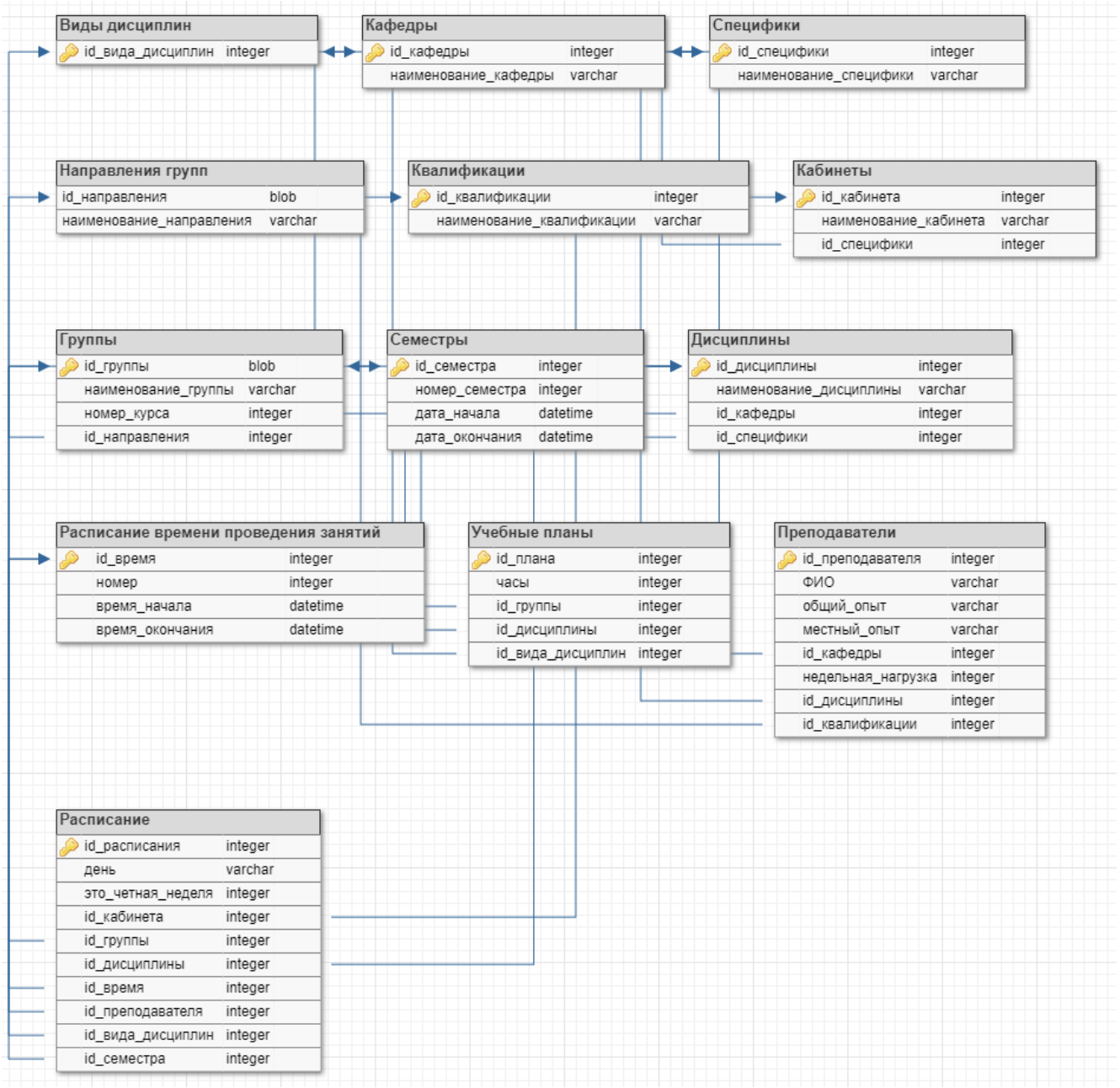

Рисунок 2.2 – Представление таблиц (сущностей) и связей между ними

2.3 Технологическое обеспечение работы

В этом разделе в соответствии со спроектированным приложением будут выдвигаться требования к технологической платформе приложения (программной составляющей), и в соответствующих подразделах будет произведен обзор существующих решений и выбор наиболее оптимального варианта.

2.3.1 Языки программирования

Web-технологии никогда не были такими изысканными и гибкими, как сегодня. При необходимости создать качественный web-сайт или web-приложение, можно без труда найдёте массу инструментов для реализации любых своих задумок. И один из них – Python.

Вопреки распространенному мнению, Python полезен не только для обработки данных и скриптов. Посмотрев на рейтинг портала HotFrameworks.com можно заметить, что пятый по популярности web-фреймворк создан для работы с Python.

При разработке web-приложения Python в основном используется для обработки бэкэнд и маршрутизации, где в роли конкурентов у него PHP и Ruby. Web-страницы все равно необходимо отображать с использованием HTML и CSS, а функциональная часть фронтэнда по-прежнему выполняется на JavaScript.

В зависимости от того, какой фреймворк Python используется, взаимодействия существенно упрощаются. Например, Django имеет систему шаблонов для написания специальных HTML-файлов, которые могут вставлять код Python и взаимодействовать с данными из бэк-энда.

Такой тип взаимодействия называется full-stack фреймворком. С его помощью можно работать с системами, обрабатывающими HTTP-запросы, хранилищами БД, шаблонами web-страниц, запросами маршрутизации и т. д. С другой стороны, есть и не full-stack фреймворки, которые также называют микрофрейм-

ворками, которые обрабатывают только базовую логику. А для сторонних работ они должны быть объединены со сторонними БД, шаблонизаторами и т. д.

В общем, full-stack фреймворки заставляют принимать множество решений относительно структуры, но предоставляют все, что может портребоваться. Микрофреймворки могут быть изучены в кратчайшие сроки и являются более гибкими, но на них, зачастую необходимо придумывать новое решение в том случае, когда уже имеется надёжное, проверенное существующее решение.

Для использования языка Python в web-разработке, следует рассмотреть возможность использования одного из следующих фреймворков. Каждый из них по-своему хорош, проверен временем и тысячами программистов.

Django – наиболее известный и популярный фреймворк для web-разработки с использованием Python. Он поставляется с десятками встроенных модулей, прекрасно собранных и безупречно взаимодействующих друг с другом. Сначала потребуется немного времени, чтобы понять алгоритм создания web-приложений, освоить внутренние структуры на Django. Но как только придёт понимание, быстрая разработка не станет проблемой. К плюсам разработки, можно отнести то, что для web-разработчик на Python, знание Django – часто единственный путь.

Главный плюс Django – с его помощью приложение очень хорошо масштабируется. По мере того, как в процессе разработки оно будет становится все больше и больше, с Django куда проще поддерживать организованность, чем с любым другим Python-фреймворком. Django – фреймворк с открытым исходным кодом.

Flask – микрофреймворк, в некотором роде являющийся полном противоположностью Django. Он будет прост и понятен новичку, но обеспечит лишь базовый уровень возможностей, в то время как основную функциональность на себя должны будут взять сторонние интегрированные компоненты.

Если нет ни опыта работы, ни понимания того, как должно функционировать будущее приложение, то Flask – далеко не лучший выбор.

Pyramid – это некий компромисс между Django и Flask. Данный фреймворк не так функционален, как Django, и более функционален, чем Flask, но прост, удобен и вполне достаточен для организации большинства web-приложений. Здесь

есть большая библиотека официальных и неофициальных плагинов, с помощью которых возможно реализовать все задумки для приложения.

Для написания приложения был выбран фреймворк Django. Рассмотрим подробнее выбранный фреймворк. Опишем причины его выбора.

Так как в БУ «Нижневартовский Социально Гуманитарный Колледж» имеются обучающиеся по направлению «Программирование в КС», то вероятно у студентов может возникнуть идея провести SQL-инъекцию или как-то иначе повлиять на работу приложения, например, заменить себе расписание. Это подталкивает нас к проблеме безопасности выбранного фреймворка.

Django помогает разработчикам избегать многих распространенных ошибок безопасности, предоставляя инфраструктуру, которая была разработана для «правильного решения», чтобы автоматически защитить сайт. Например, Django обеспечивает безопасный способ управления учетными записями пользователей и паролями, избегая распространенных ошибок, таких как включение информации о сеансе в файлы cookie, где она уязвима (вместо этого cookie-файлы содержат только ключ, а фактические данные хранятся в БД), или хранение паролей в открытом виде, вместо их хэшей.

Хэш пароля – это значение фиксированной длины, созданное путем обработки пароля через криптографическую хэш-функцию. Django может проверить правильность введенного пароля, пропустив его через хэш-функцию и сравнив вывод с сохраненным значением хэша. Благодаря «одностороннему» характеру функции, даже если сохраненное хэш-значение скомпрометировано, злоумышленнику будет сложно извлечь исходный пароль.

Django обеспечивает защиту от многих уязвимостей по умолчанию, включая SQL-инъекцию, межсайтовый скриптинг, межсайтовую подделку запросов и кликджекинг.

Так как требования к приложению могут изменятся и самому приложению может потребоваться новый функционал, то фреймворк должен быть масштабируемый.

Django использует компонентную архитектуру (каждая часть архитектуры не зависит от других, и следовательно, может быть заменена или изменена при необходимости). Четкое разделение между различными частями означает, что оно может масштабироваться для увеличения трафика путем добавления оборудования на любом уровне: кеширующие серверы, серверы БД или серверы приложений. Некоторые из самых посещаемых сайтов успешно масштабируются Django для удовлетворения своих требований.

Стоит учесть ещё тот факт, что после внедрения web-приложения его нужно сопровождать.

Код Django написан с использованием принципов и шаблонов дизайна, которые поощряют создание поддерживаемого и многоразового кода. В частности, он использует принцип Do not Repeat Yourself (DRY), поэтому нет ненужного дублирования, что уменьшает количество кода. Django также способствует группировке связанных функций в многоразовые «приложения» и на более низком уровне группирует связанный код модулей в соответствии с Model View Controller (MVC) паттерном.

2.3.2 Среда разработки

Для написания программирования и отладки web-приложения использовался кроссплатформенный проприетарный текстовый редактор Sublime Text 3. Поддерживает плагины на языке программирования Python.

Поддержка языков

Sublime Text поддерживает большое количество языков программирования и имеет возможность подсветки синтаксиса для C, C++, C#, CSS, D, Dylan, Erlang, HTML, Groovy, Haskell, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, MATLAB, OCaml, Perl, PHP, Python, R, Ruby, SQL, TCL и XML.

В дополнение к тем языкам программирования, которые включены по умолчанию, пользователи имеют возможность загружать плагины для поддержки других языков.

Менеджер пакетов

Sublime Text может быть оснащён менеджером пакетов, который позволяет пользователю находить, устанавливать, обновлять и удалять пакеты без перезагрузки программы. Менеджер поддерживает установленные пакеты в актуальном состоянии, загружая новые версии из репозиториев. Кроме того, он предоставляет команды для активации и деактивации установленных пакетов.

Интерфейс

Редактор содержит различные визуальные темы, с возможностью загрузки дополнительных.

Пользователи видят весь свой код в правой части экрана в виде мини-карты, при клике на которую можно осуществлять навигацию.

Есть несколько режимов экрана. Один из них включает от 1 до 4 панелей, с помощью которых можно показывать до четырёх файлов одновременно. Полноценный (free modes) режим показывает только один файл без каких-либо дополнительных вокруг него меню.

Выделение столбцов и множественная правка

Выделение столбцов целиком или расстановка несколько указателей по тексту, что делает возможным мгновенную правку. Указатели ведут себя, будто каждый из них – единственен в тексте. Команды типа: перемещение на знак, перемещение на строку, выборка текста, перемещение на слово или его части (CamelCase, разделённый дефисом или подчёркиванием), перемещение в начало/конец строки и т. д., влияет на все указатели независимо и сразу, позволяя править сложно структурированный текст быстро, без использования макрокоманд или регулярных выражений.

Автодополнение

Когда пользователь набирает код, Sublime Text, в зависимости от используемого языка, будет предлагать различные варианты для завершения записи. Редактор также автоматически завершает созданные пользователем переменные.

Подсветка синтаксиса и высокая контрастность

Тёмный фон Sublime Text предназначен для увеличения контрастности текста. Основные элементы синтаксиса выделены разными цветами, которые лучше сочетаются с тёмным фоном, нежели со светлым.

Поддержка систем сборки

Sublime Text позволяет пользователю собирать программы и запускать их без необходимости переключаться на командную строку. Пользователь также может настроить свою систему сборки и включить автоматическую сборку программы каждый раз при сохранении кода.

Заготовки (сниппеты)

Сохранение фрагментов часто используемого кода, ключевые слова для их запуска.

Переход по файлам

Навигационный инструмент, который позволяет пользователям перемещаться между файлами, а также внутри них, с помощью нечёткого поиска.

Другие особенности

− Дополнительно реализована функция автосохранения, помогающая пользователям не потерять проделанную работу.

− Возможность поиска по мере набора используется для поиска в документе. Функция проверки синтаксиса работает подобным же образом, проверяя корректность прямо во время ввода.

− Есть возможность автоматизации с помощью макросов и повтора последних действий.

− Команды редактирования, включая редактирование отступов, переформатирование параграфов и объединение строк.

2.3.2.1 Настройка среды разработки

Для более удобной работы в Sublime Text были установлены плагины, которые помогли ускорить процесс разработки.

Плагины – это то, что поможет сделать из Sublime Text IDE. Для начала установим Sublime Package Control, он поможет легко управлять плагинами.

После установки в меню появился пункт Package Control. Большинство плагинов можно уже найти в списке предложенных для установки. Далее будут описаны те плагины, которые были выбраны для написания приложения.

- Djaneiro – Плагин добавляет подсветку для шаблонов Django и большое количество снипетов для ускорения процесса написания кода.
- Python Improved – Улучшенная подсветка синтасиса для Python 2 и 3. Чтобы подсветка работала с SublimeLinter, необходимо в настройке syntax_map последнего добавить: "pythonimproved": "python".
- SublimeLinter 3 – Для установки валидаторов. Валидаторы обычно требуют установки необходимых библиотек в систему, при установке. Sublime Text 3 использует Python 3.

Список валидаторов, которые были использованы для написания приложения:

- SublimeLinter-pep8;
- SublimeLinter-pylint;
- SublimeLinter-jshint;
- SublimeLinter-pyflakes.

pylinter обычно показывает большое количество предупреждений. В документации SublimeLinter можно найти описание настроек проверки. Были использованы следующие настройки:

```
{
    "spell_check": true,
    "dictionary": "Packages/Language - English/en_US.dic",
    "highlight_line": true,
    "translate_tabs_to_spaces": true,
    "trim_trailing_white_space_on_save": true,
    "auto_complete_triggers":   [{"selector":   "source.python",
"characters": "."}]
}
```

2.3.3 Web-сервер

Web-сервер – сервер, принимающий HTTP-запросы от клиентов, обычно web-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными.

Web-сервером называют как программное обеспечение, выполняющее функции web-сервера, так и непосредственно компьютер, на котором это программное обеспечение работает.

Клиент, которым обычно является web-браузер, передаёт web-серверу запросы на получение ресурсов, обозначенных URL-адресами. Ресурсы – это HTML-страницы, изображения, файлы, медиа-потоки или другие данные, которые необходимы клиенту. В ответ web-сервер передаёт клиенту запрошенные данные. Этот обмен происходит по протоколу HTTP.

Для разработки web-приложения будет использоваться отладочный сервер, предоставляемый Django.

2.4 Техническое обеспечение работы

Рассмотри аппаратное оснащение, необходимое для работы приложения. В каждом подразделе выдвигаются требования к вычислительным ресурсам сервера и пользовательского компьютера на основании произведенных расчетов и измерений.

Серверной тестовой платформой для проведения измерений потребления ресурсов приложениями является виртуальная машина, созданная в программе Sun VirtualBox, со следующей конфигурацией:

- процессор – Intel Pentium G3420 3.20GHz;
- размер ОЗУ – 2 Гб;
- размер ПЗУ – 8 Гб;
- операционная система – Ubuntu Server 16.04 LTS;
- web-сервер.

2.4.1 Серверное аппаратное оснащение

Django сервер и СУБД работают на одном сервере по следующим причинам:

- высокая доступность такого решений;
- удобство резервного копирования данных;
- отсутствие высоких нагрузок при работе приложения.

Основа работы web-сервера – это выполнение запросов от клиента, все процессы выполняться на клиенте. Такой вариант является наиболее производительным.

Стоит учесть, что для поддержания работоспособной ОС Ubuntu Server необходимо в среднем 150 Мб, для работы СУБД – 30 Мб.

Расчет необходимого места в ПЗУ:

- Установленный дистрибутив Ubuntu Server 16.04 – 1 Гб,
- SQLlite – 60 Мб,

Итого, для работы приложения необходимо приблизительно 6 Гб ПЗУ.

После приведенных расчетов можно сделать вывод, что для полноценной работы приложения при обслуживании порядка не более 100 запросов в день достаточно будет следующей конфигурации:

- процессор – Intel с поддержкой EM64T, AMD с поддержкой AMD64 (с архитектурой x86-64)
- ОЗУ – от 2048 Мб DDR3,
- ПЗУ – от 40 Гб;
- LAN (100 Мбит.с)
- USB-порт
- SVGA-видеокарта

2.5 Разработка приложения на Django

Рассмотрим структуру приложения. Опишем основные этапы создания разрабатываемого приложения.

2.5.1 Создание структуры проекта. Настройка

Для создания пустого проекта в терминале выполним команду:

```
django-admin.py startproject CollegeTimetable
```

После выполнения скрипта создастся шаблон для написания приложения. Структура шаблона выглядит слудующим образом:

CollegeTimetable /
- manage.py
- CollegeTimetable /
  - __init__.py
  - settings.py
  - urls.py
  - wsgi.py

Рассмотрим эти файлы:

Внешний каталог CollegeTimetable / – просто контейнер для проекта. Его название никак не используется Django и можно назвать его как угодно.

manage.py: скрипт, который позволяет взаимодействовать с проектом Django.

Внутренний каталог CollegeTimetable / это пакет Python проекта. Его название – это название пакета Python, которое будет использоваться для импорта чего-либо из проекта (например, import CollegeTimetable.settings).

CollegeTimetable /__init__.py: пустой файл, который указывает Python, что текущий каталог является пакетом Python.

CollegeTimetable/settings.py: Настройки/конфигурация проекта.

CollegeTimetable /urls.py: Конфигурация URL для проекта Django.

CollegeTimetable /wsgi.py: Точки входа для WSGI-совместимый web-серверов.

Далее с помощью скрипта manage.py создадим приложение main, для этого в терминале выполним команду:

```
Python manage.py startapp main
```

Эта команда создаст каталог main:
main /

__init__.py

models.py

tests.py

views.py

Отредактируем файл CollegeTimetable/settings.py под свой проект.

Укажем пути к файлам, где будут храниться статические файлы(скрипты js и листы стилей):

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, "static")]
```

Установим русский язык для стандартных элементов:

```
LANGUAGE_CODE = 'ru-RU'
```

Установим временную зону:

```
TIME_ZONE = 'UTC'
```

Для того, чтобы иметь возможность удалить свыше 1000 записей из БД, необходимо задать следующую настройку:

```
DATA_UPLOAD_MAX_NUMBER_FIELDS = None
```

Зарегистрируем наше приложение:

```
 INSTALLED_APPS = [
…
    'main',
]
```

Добавим разрешимые хосты, на которых можно запускать сервер:

```
ALLOWED_HOSTS = ['192.168.1.XXX, '0.0.0.0', '127.0.0.1']
```

2.5.2 Создание структуры БД

В Django большинство взаимодействий с БД осуществляется посредством механизма объектно-ориентированного отображения (Object-Relational Mapper или ORM) – функциональности, имеющейся, помимо Django, и в других современных инфраструктурах Web-разработки, таких как Rails. Системы ORM обретают все большую популярность среди разработчиков, так как они автоматизируют множество типичных взаимодействий с БД и используют знакомый объектно-ориентированный подход вместо инструкций SQL.

Откроем файл models.py и запишем структуру, определённую в разделе 2.1

Полная структура ORM описана в Приложении А. Исходный код web-приложения «Информационная система для автоматизации ведения и автоматического заполнения расписания колледжа».

После заполнения файла models.py необходимо выполнить миграцию, для этого выполним две команды в терминале:

```
Python manage.py migrate
Python manage.py makemigrations
```

2.5.3 Алгоритмы

В данном подразделе опишем основные алгоритмы работы приложения. Алгоритмы описаны в файле view.py.

2.5.3.1 Алгоритм создание или обновления шаблона расписания

Для проверки шаблона таблицы расписания нам необходимо проверить чтобы для каждого заданного дня, для каждого вида недели (четная\нечётная), для каждой имеющейся группы, для каждого имеющегося номера пары, для каждого имеющегося семестра была создана строка. Для этого циклами проходимся по таблице шаблона расписания и проверяем, есть ли заданные строки, если строки имеются, то пропускаем и двигаемся на шаг вперед, если строка отсутствует, то со-

здаём её с заполненными значениями: Группа, Семестр, Чётная\нечетная неделя, День недели, Номер пары. И с пустыми значениями: Предмет, Кабинет, Преподаватель.

2.5.3.2 Алгоритм распределения нагрузки

Для распределения нагрузки необходимо узнать, сколько чётных и нечётных недель в каждом семестре, определить какие дисциплины можно равномерно распределить по неделям, сколько часов равномерно не распределены и поставить им метку, что данные часы необходимо поставить в расписание вручную. Из журнала нагрузки сформировать временный журнал с метками, указывающими в каком семестре, в какой неделе будут данные часы, и какие часы необходимо распределить вручную

2.5.3.3 Алгоритм автоматического составления расписания

Для автоматического формирования расписания необходимо записи, полученные в алгоритме 2.5.3.2 распределить по таблице, полученной в алгоритме 2.5.3.1, с учётом всех ограничений описанных в техническом задании.

2.5.4 Структура разработанного приложения

По окончанию разработки структура приложения представлена на рисунке 2.3. Весь код программы можно посмотреть в Приложении А. Исходный код web-приложения «Информационная система для автоматизации ведения и автоматического заполнения расписания колледжа».

Рисунок 2.3 – Структура разработанного приложения

2.6 Разработка пользовательского интерфейса. Руководство пользователя

В разработке интерфейса web-приложения использовалась библиотека «Bootstrap».

Bootstrap – свободный набор инструментов для создания сайтов и web-приложений. Включает в себя HTML и CSS-шаблоны оформления для типографики, web-форм, кнопок, меток, блоков навигации и прочих компонентов web-интерфейса, включая JavaScript-расширения [9].

Интерфейс пользователя приложения разрабатывался с учётом требований простоты, удобства.

Начальная страница web-приложения для неавторизованного пользователя представляет собой страницу «Расписание» с фильтрацией по группам и кнопкой авторизации (Рисунок 2.4). Студентам для использования web-приложения авторизация не требуется, так как они пользуются им только для просмотра расписания и поиска преподавателя, иначе говоря никаких данных на сервере не изменяют.

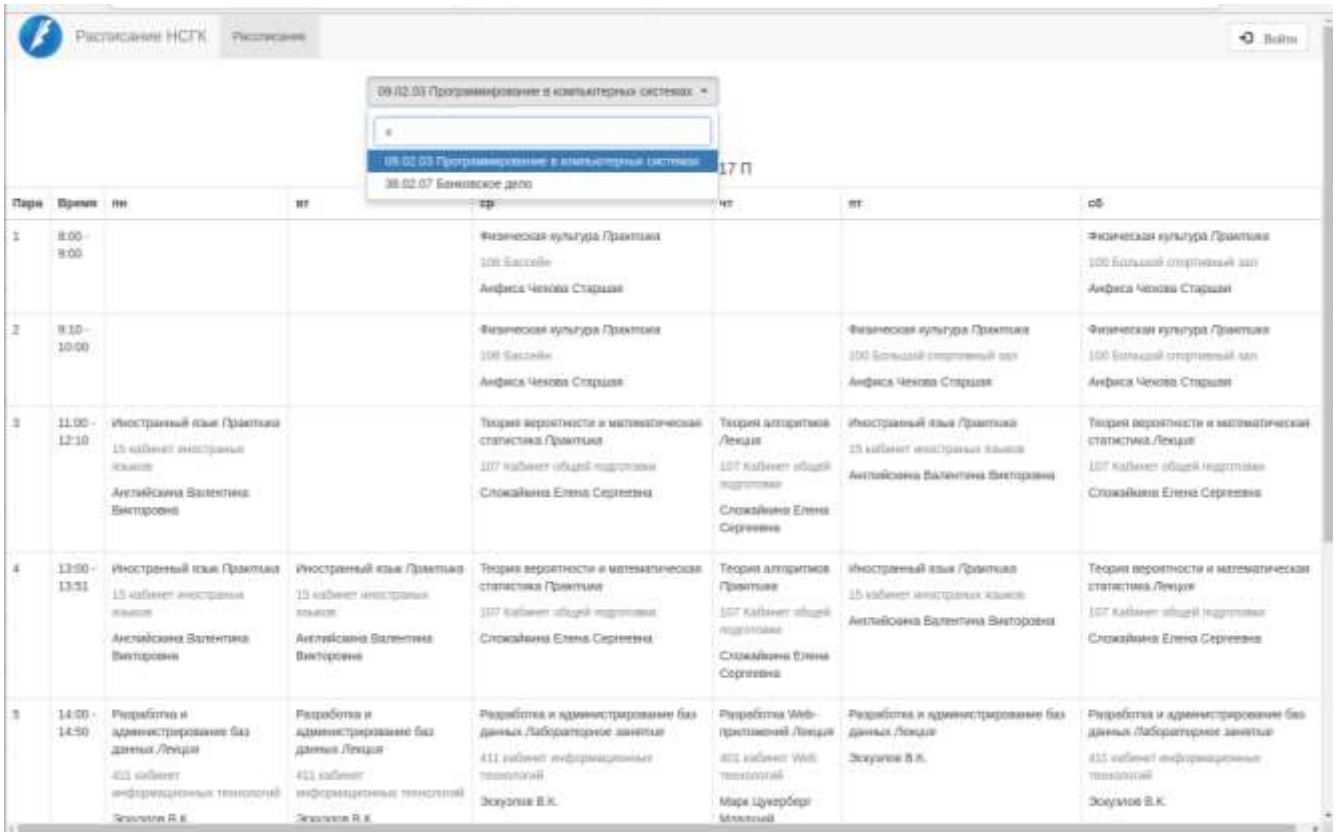

Рисунок 2.4 – Начальная страница

Для работы диспетчера, необходимо авторизоваться в web-приложении щёлкая на копку «Войти». Появляется окно авторизации, в которое диспетчер вводит свой логин и пароль (Рисунок 2.5).

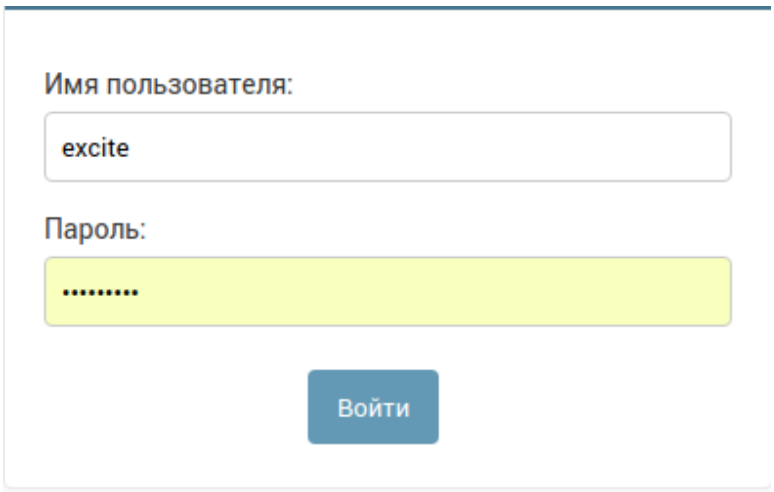

Рисунок 2.5 – Авторизация в приложении

После успешного прохождения авторизации в приложении появляется модуль диспетчера для работы с составлением расписания и появляется возможность вывести печатную версию расписания. В печатной версии расписания выводятся данные по всем группам и за все дни недели. (Рисунок 2.6).

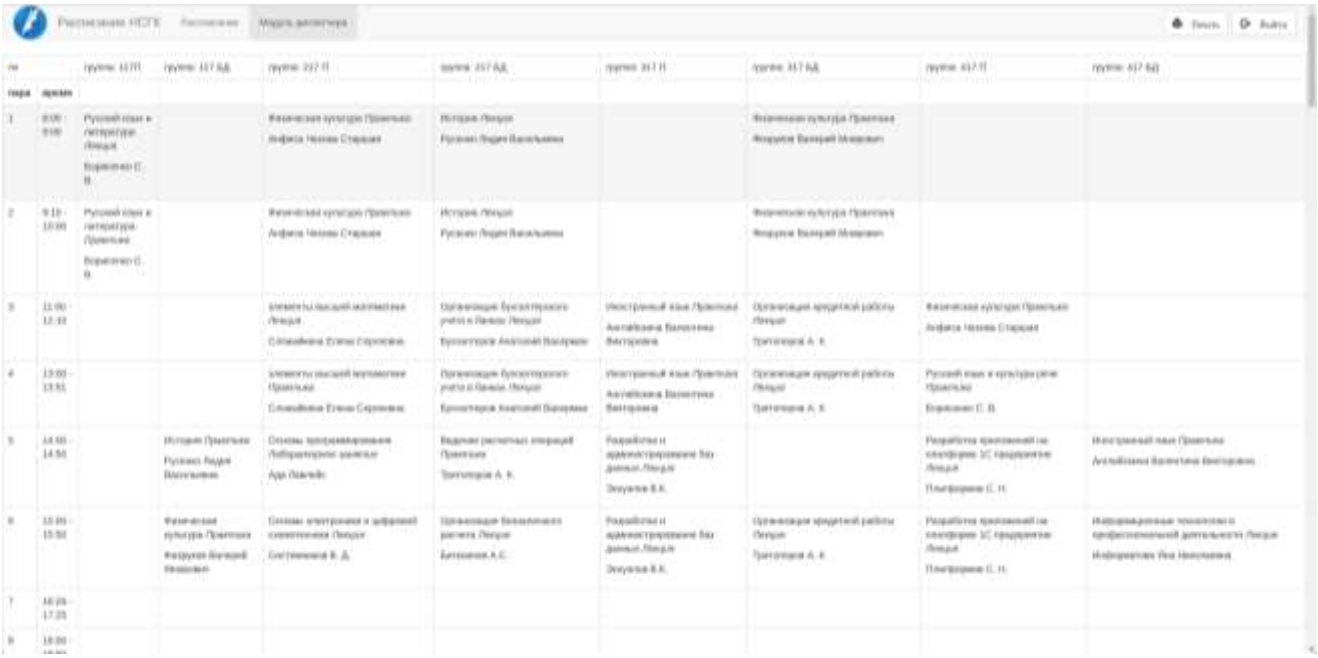

Рисунок 2.6 – Печатная форма «расписания»

Для авторизованного в сервисе пользователя доступен пункт меню «Модуль диспетчера», в котором компактно собран весь функционал работы диспетчера.

Для начала работы в сервисе пользователю необходимо заполнить начальные данные. Начальные данные заполняются в разделе справочная информация на странице «Модуль диспетчера». (Рисунок 2.7). Диспетчеру необходимо завести в систему преподавателей, кабинеты, расписание звонков, имеющиеся в колледже дисциплины.

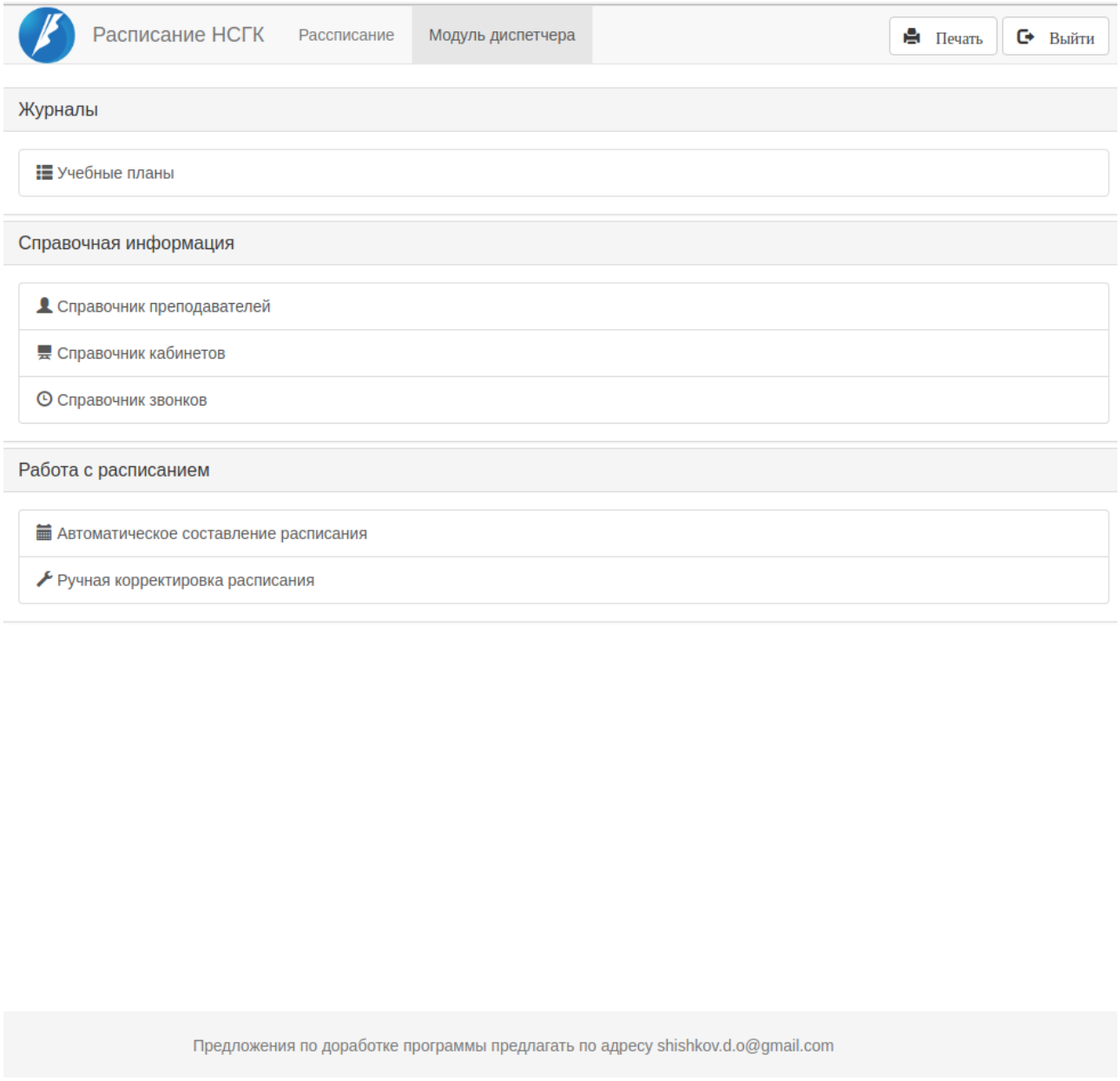

Рисунок 2.7 – Страница «Модуль диспетчера»

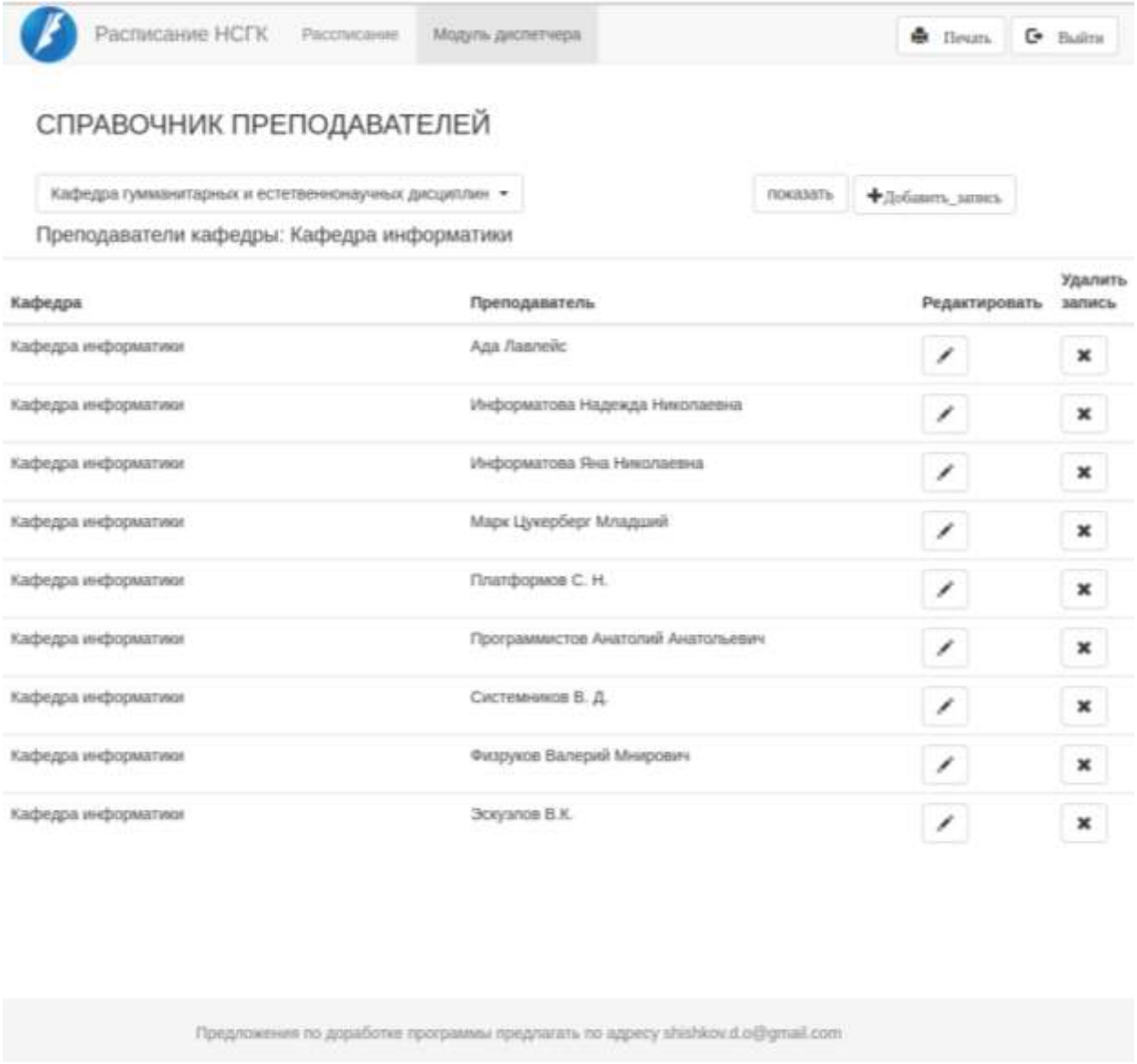

Рисунок 2.8 – Страница «Справочник преподавателей»

Как все необходимые справочники будут заполнены. Диспетчер должен заполнить учебный план. Для этого на модуле диспетчера, он переходит на страницу «Учебные планы» (Рисунок 2.9) и нажимает на кнопку добавить запись.

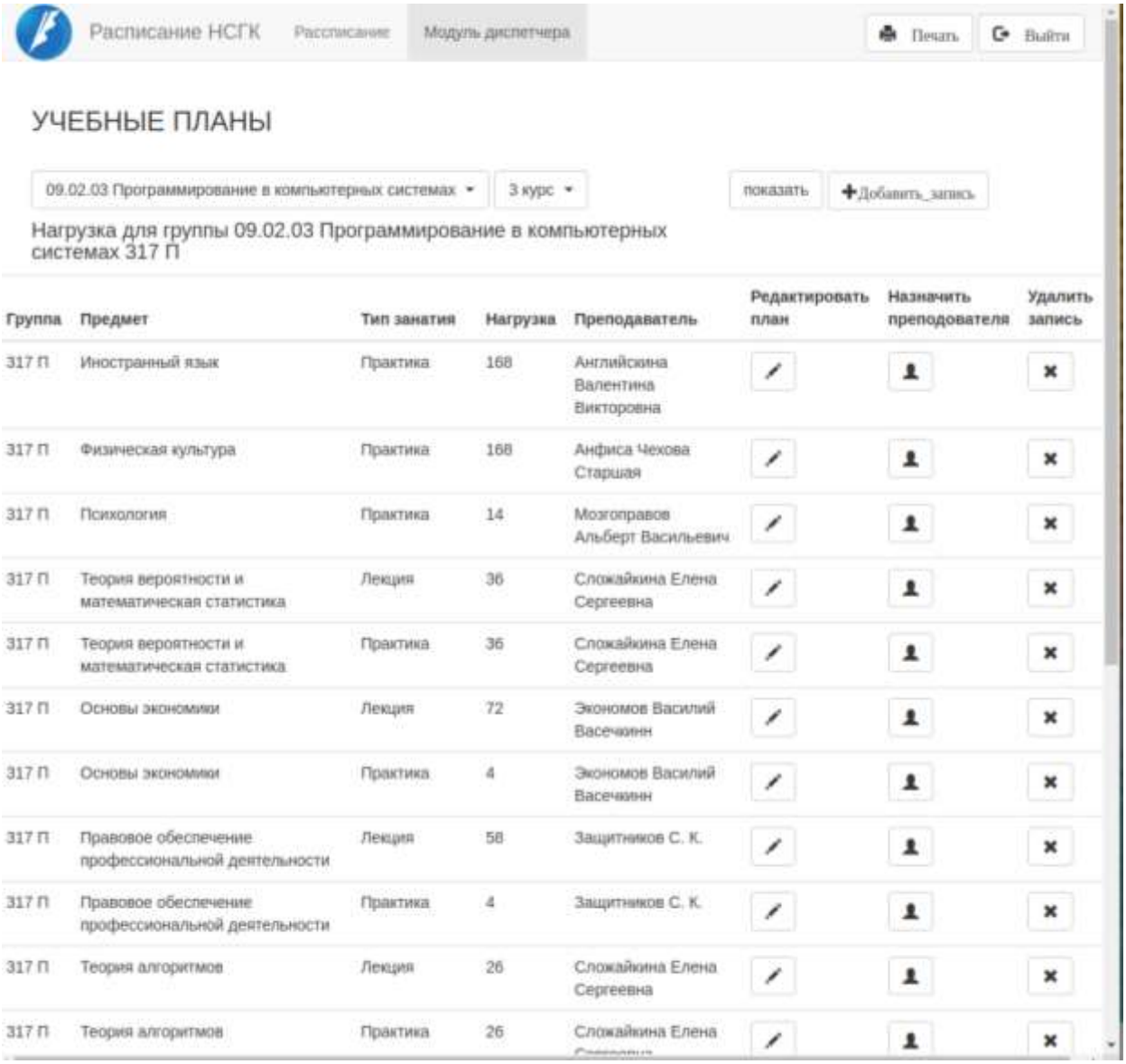

Рисунок 2.9 – Журнал «Учебный план»

Изначально учебные планы заполняются без учета ведущих дисциплину преподавателей и кабинетов. Диспетчер выбирает группу из списка доступных, выбирает дисциплину, тип занятий, в каких семестрах будет проводится данная дисциплина. Далее если диспетчуру известно какой преподаватель будет вести дисциплину у выбранной группы, то он може сразу указать эти сведения. Иначе говоря поля «возможные кабинеты для проведения занятия» и «преподаватель» не являются обязательными для заполнения на данном этапе. (Рисунок 2.10)

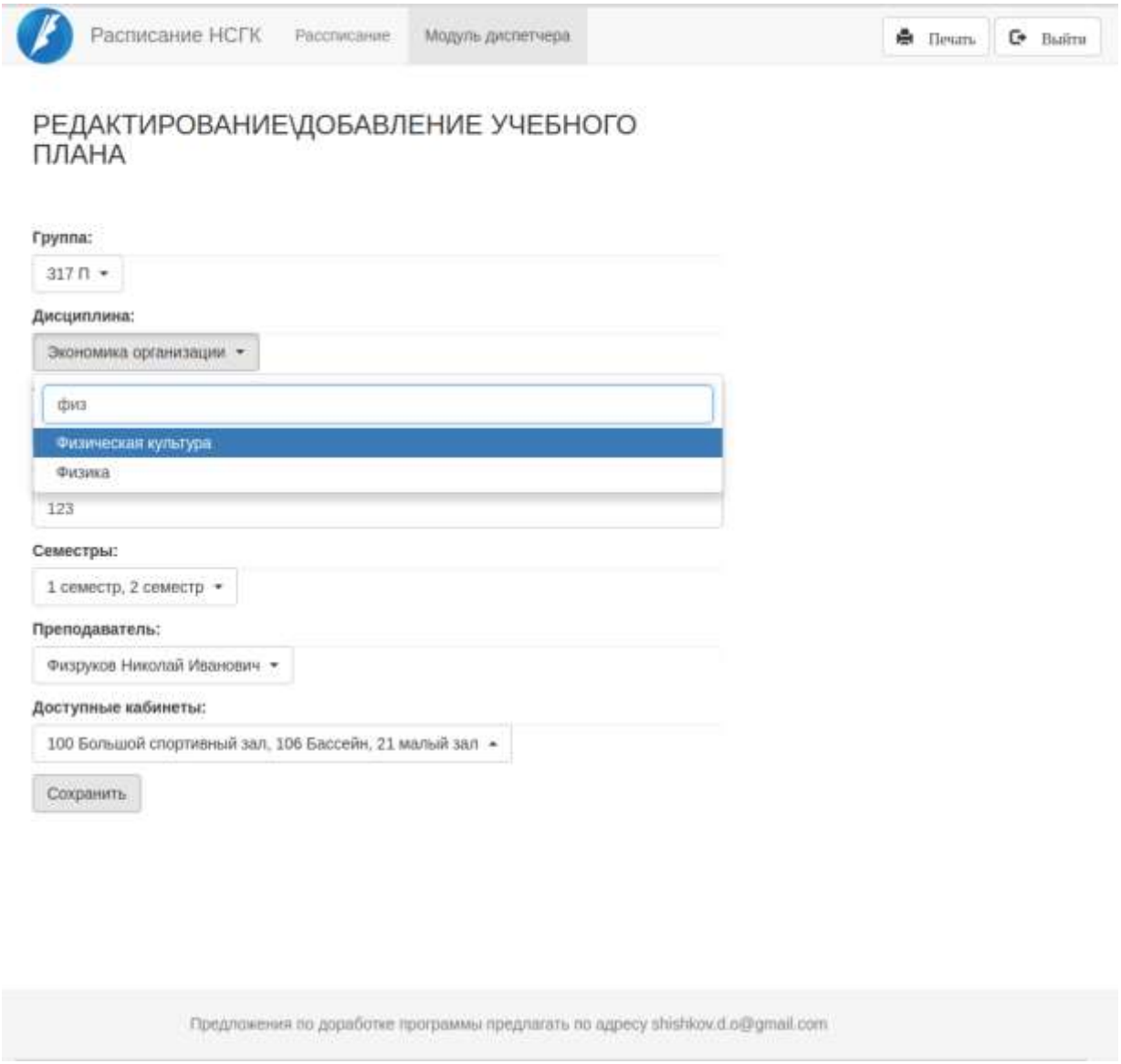

Рисунок 2.10 – Добавление учебного плана

Как только все данные будут заполнены, диспетчер переходит в раздел автоматического формирования расписания.

Автоматическое формирование расписания происходит в три этапа.

Первым этапом является проверка целостности шаблона расписания, при добавлении новых групп или удалении, алгоритм переформирует шаблон для заполнения под новые данные. Если шаблон расписания отсутствует, то создастся новый шаблон.

Вторым этапом алгоритм вычисляет или пересчитывает нагрузку дисциплины в неделю, выявляет её возможность стоять в чётной или нечётной неделе. Распределяет нагрузку по сессиям.

Третий этап расставляет дисциплины из полученной недельной нагрузки для групп по дням, в шаблон недельного расписания с проверкой на пересечение кабинетов, преподавателей и нагрузкой преподавателя и группы.

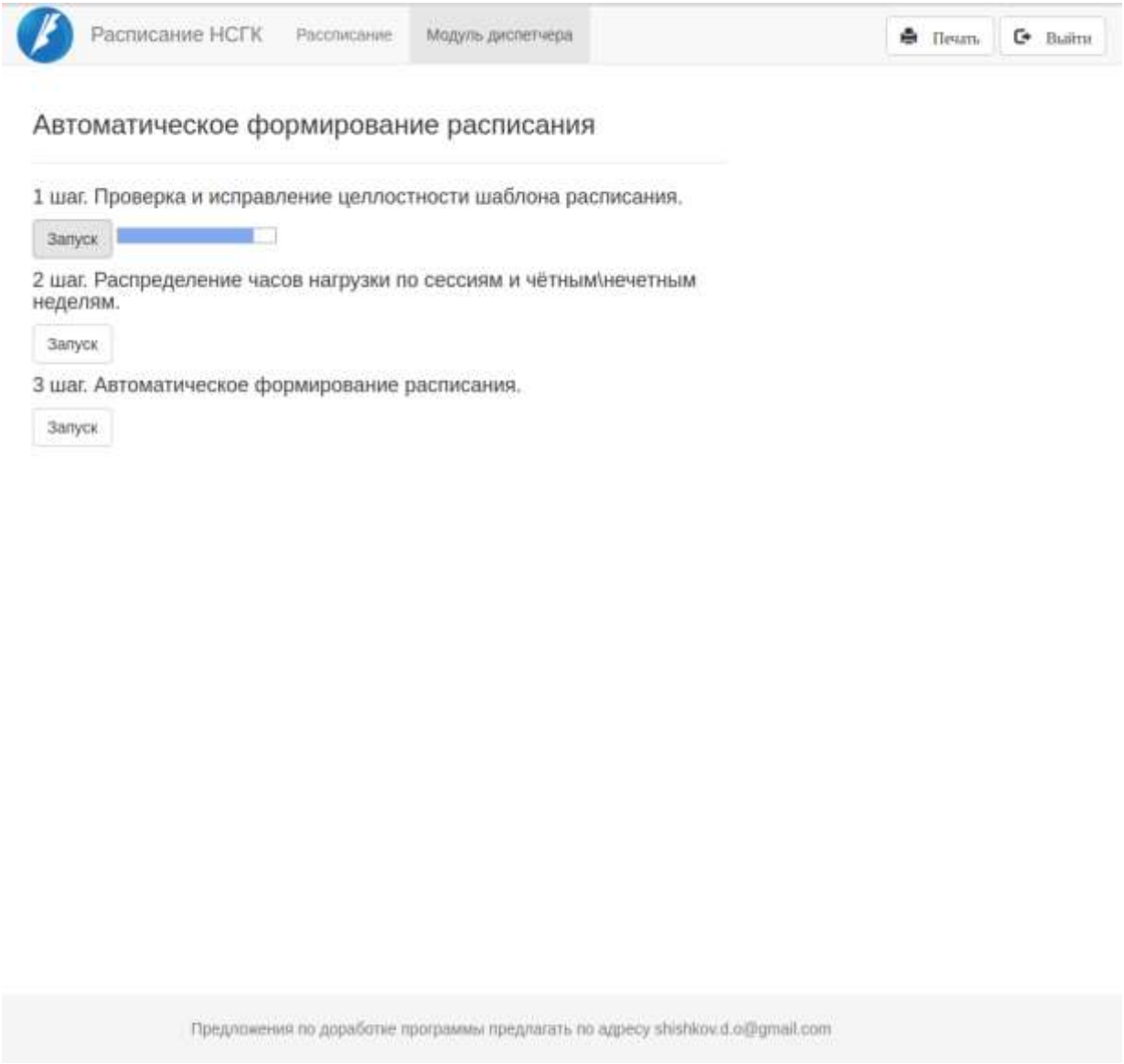

Рисунок 2.11 – «Автоматическое формирование расписания»

Далее диспетчер заходит в раздел web-приложения «ручная корректировка расписания» и распределяет оставшиеся дисциплины, которые не имеют частых повторений для внесения в недельную таблицу, по конкретным дням. Система проверяет на все ограничения и только после этого даёт диспетчеру возможность сохранить дисциплину на день.

Выводы по разделу два:

В данном разделе были рассмотрены следующие вопросы:

1. Проектирование общей структуры программы. Построен и описан принцип взаимодействия всех элементов, поддерживающих приложение в работоспособном состоянии.

2. Выбор средств реализации приложения. Подобранно программное обеспечение для внедрения БД в эксплуатацию, с подробным описанием возможных вариантов использования.

3. Технологическое обеспечение. Подобранна оптимальная конфигурация персонального компьютера для поддержки работы сервера в оптимальном состоянии.

4. Разработка web-приложения. Описание спроектированной структуры БД посредством ORM, разработка алгоритмов.

5. Разработка руководства пользователя и описание работы приложения.

3 ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

Web-приложение для автоматического формирования расписания предназначено для повышения эффективности работы диспетчера.

Целью выпускной квалификационной работы является разработка web-приложения, которое позволяет автоматически формировать расписания занятий на основе справочных данных и получить удалённый доступ к расписанию для студента.

Снижение трудовых затрат позволит уменьшить и финансовые затраты, что приведет к общему увеличению производительности и экономии.

Основной задачей этого раздела является определение величины затрат на проведение исследований, себестоимость для определения экономического эффекта от использования в общественном производстве основных и сопутствующих результатов, получаемых при решении поставленной технической задачи в данной выпускной квалификационной работе. Оценка эффективности принятого научно-технического решения должна учитывать все необходимые расходы и затраты, для этого требуется провести ряд необходимых расчетов по определенной схеме.

3.1 Расходы на приобретение, содержание и эксплуатацию программного и аппаратного обеспечения

Стоимость затрат взята из списков цен в сетевых магазинах, и представлена в таблице 3.1.

Таблица 3.1 – Стоимость программного обеспечения и аппаратных средств

Наименование	Кол-во	Цена, руб.
Ноутбук LENOVO IdeaPad 110-15ACL	1	19 999
Текстовый редактор Sublime Text 3	1	4 566
Итого		24565

Таблица 3.2 – Материалы

Наименование	Кол-во	Цена, руб.
Kingston DataTraveler SE9 16GB16Gb (DTSE9H)	1	380
Диск DVD+R TDK 4.7Gb 16x Slim	1	46
Итого		426

Затраты на электроэнергию находятся исходя из продолжительности периода разработки ПО, количества кВт/ч, затраченных на проектирование ПО и тарифа за 1 кВт/ч. Тариф по городу Нижневартовску для юридических лиц составляет 4.90 руб. за кВт/ч. Затраты отражены в таблице 3.3.

Таблица 3.3 – Затраты на электроэнергию

Элемент системы	Установленная мощность, кВт	Стоимость 1кВт в час (руб.)	Кол-во часов работы	Общая стоимость рублей
Ноутбук LENOVO IdeaPad 110-15ACL	0,057	4,90	398	111,161
Итого				111,161

Затраты на амортизацию оборудования проводятся за период их использования, т.е. за период внедрения и создания дополнений к программному обеспечению.

Денежное выражение амортизации является амортизационным отчислением, которое входит в текущие затраты.

Величина амортизационных отчислений определяется на основе норм амортизации.

Норма амортизации – это установленный размер амортизационных отчислений на полное восстановление, выраженное в %. Норма амортизации устанавливается на основе экономически целесообразного срока службы и должна обеспечить возмещение износа основных средств к моменту возможного их морального и физического износа и создать экономическую основу для замены.

Амортизационные отчисления, приходящиеся на 1 час работы системы, рассчитываются по формуле (1).

$$\text{Ач} = \Phi_{\text{перв}} \cdot \frac{\text{а}}{F_{\text{д}}}, \quad (1)$$

где $Ф_{перв}$ – первоначальная стоимость системы или отдельных элементов;

а – норма амортизации (0.2);

$F_д$– фонд времени работы за год (2500 часов).

Таблица 3.4 – Расчет амортизационных отчислений

п/п	Элемент КТС	Ф перв	Fд	Ач	Кол-во часов работы	Общая стоимость (руб.)
1	Ноутбук LENOVO IdeaPad 110-15ACL	19 999	2500	1.5999	398	638.8
2	Текстовый редактор Sublime Text 3	4 566	2500	0.365	398	145,38
Итого						784.181

Просуммировав расчет, мы получили расходы на приобретение, содержание и эксплуатацию программного и аппаратного обеспечения равного 1321,342 рублей.

3.2 Затраты на заработную плату

Для расчета расходов на заработанную платы необходимо умножить среднюю часовую ставку программиста на трудоемкость работы, чел/час по каждому из этапов разработки системы.

Средняя часовая ставка взята по формуле (2):

$$З_ч = \frac{З_м}{168}, \quad (2)$$

где $З_ч$– средняя часовая ставка программиста.

$З_м$ – средняя месячная ставка начинающего программиста (30000 рублей).

$З_ч$ = 30000 /168 = 179 рублей.

$З_ч$= 250 рублей – среднечасовая ставка руководителя ВКР и консультанта по БЖД.

$З_ч$= 350 рублей – среднечасовая ставка консультанта по экономической части.

Исходя из полученных данных, можно вычислить заработную плату по всем этапам разработки, результат в таблице 3.5.

Таблица 3.5 – Расчет основной и дополнительной заработной платы

п/п	Содержание работы	Трудоемкость работы, чел /час	Основная заработная плата (руб.)
1.	Анализ предметной области	16	2880
2.	Постановка задачи	8	1040
3.	Разработка технического задания	12	2260
4.	Проектирование БД	32	4160
5.	Разработка интерфейса программы	24	4120
6.	Разработка модулей	115	15950
7.	Тестирование системы	120	15600
8.	Документирование	11	1550
9.	Руководство ВКР (руководитель ВКР и консультанты)	20	5100
10.	ИТОГО́	398	52660

3.3 Расчет затрат на дополнительную заработную плату

Дополнительную заработную плату разработчиков определяют в процентах от итоговой суммы основной заработной платы (15 %).

ЗП доп.= 52660 · 0,15 = 7899 (руб.).

3.4 Отчисления на социальные нужды

Отчисления на социальные нужды рассчитывают в процентах от суммы основной и дополнительной заработных плат, в пенсионный фонд, в ФСС и мед. страхование. На 2018 год данный процент составляет 30%, рассчитывается по формуле (3).

$$ОСН = 30\% \ (ЗП_{осн} + ЗП_{доп}), \quad (3)$$

где *ОСН* – отчисления на социальные нужды;

$ЗП_{осн}$ – основная заработная плата;

$ЗП_{доп}$ – дополнительная заработная плата.

$ОСН = 0{,}30 \cdot (52660+7899) = 18167{,}7$ (руб.)

3.5 Общая смета затрат на внедрение приложения

Таблица 3.6 – Общая смета затрат

п/п	Элементы затрат	Сумма, руб.
1.	Затраты на основную заработную плату	52660
2.	Затраты на дополнительную заработную плату	7899
3.	Отчисления на социальные нужды	18167,7
4.	Расходы на приобретение, содержание и эксплуатацию программного и аппаратного обеспечения	1321,342
ИТОГО		80 048

3.6 Оценка экономической эффективности

Без использования приложения разработанного в рамках выпускной квалификационной работы происходит нерациональная трата рабочего времени сотрудника, отвечающего за составление расписания, деятельность которого связана с учетом большого количества операций, расчётов и сопоставлений. Любое изменение групп, преподавательского состава требует ручного пересчёта часов вычитки. Используя разработанное web-приложение, диспетчер рациональнее использует трудовые ресурсы.

Разрабатываемое web-приложение предназначено для облегчения работы диспетчера, поэтому дополнительных рабочих мест, а соответственно и увеличения заработной платы не планируется. Соответственно, дополнительных затрат на оплаты труда не будет.

Необходимо учесть значение средней стоимости часа работника выполняющего операции.

Для нахождения средней стоимости часа $З_ч$, необходимо разделить среднюю заработанную плату диспетчера за день (1666 рублей) на количество рабочих часов (8 часов).

$З_ч$ = 1666 /8 = 208.33 рублей.

При использовании ручного способа работы с учебными планами, сопоставления таблицы расписания тратиться $t_{п1}$ = 3ч, а с помощью программы работа с документами должна сократиться примерно до $t_{п2}$ = 0,5ч.

Экономия времени составит:

$t_{п1} – t_{п2}$ = 3ч – 0,5ч = 2.5 час в день для одного рабочего места

За месяц экономия времени одного рабочего места составит: $Т_{рм1}$ = 26 · 2,5 час = 52 ч. С учетом, что средняя заработная плата специалиста, который будет пользоваться разработанным ПО, составляет 208.33 руб/час, поэтому экономия на заработной плате за 1год составит: 12 · 26 · 2.5ч · 208.33 руб/час = 162497 руб.

В результате экономия составляет 162497 рублей.

Срок окупаемости рассчитывается по формуле (4):

$$Т_{ок} = \frac{К}{(Зр-За)+F}, \quad (4)$$

где Ток – срок окупаемости;

Зр – Затраты на ручную обработку информации, руб.;

За – Затраты на автоматизированную обработку информации, руб.;

F – Затраты на покупку формуляров в год.

К – полная стоимость владения системой, руб.

В данном случае Ток равен:

Ток=80 048/162 497 = 6 месяцев.

Срок окупаемости = 6 месяцев.

Выводы по разделу три:

В данном разделе проведен анализ расходов на приобретение, содержание и эксплуатацию программного обеспечения. Так же произведён расчет необходимых затрат на разработку системы и проведена оценка экономической эффективности.

4 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

4.1 Требования к помещениям при работе за компьютером

В помещениях, предназначенных для эксплуатации ПЭВМ, обязательно должны иметь естественное и искусственное освещение. Расположение рабочих мест с ПЭВМ в подвальных помещениях запрещено.

Площадь рабочего места с компьютером и вспомогательным оборудованием должна составлять не менее 6м^2. Без вспомогательных устройств допускается площадь 4м^2 на одно рабочее место [13].

Помещения с ПЭВМ должны обязательно быть оборудованы заземлением. Также запрещается ставить ПЭВМ вблизи силовых кабелей.

4.2 Требования к освещению помещений и рабочих мест

В помещениях с ПЭВМ должно обязательно как естественное, так и искусственное освещение. Световой поток из оконного должен падать на рабочее место с левой стороны.

Искусственное освещение в помещениях с эксплуатацией ПЭВМ обязательно должен осуществляться системой общего равномерного освещения.

Освещенность поверхности стола должна быть от 300лк до 500лк. Разрешается установка светильников местного освещения для подсветки поверхности стола при работе с документами, но запрещено, чтобы данное освещение создавала блики на поверхности экрана и увеличивала его освещенность более 300лк. Яркость светящихся поверхностей (окна, светильники), находящихся в поле зрения, должна быть не более 200 кд/м^2 [14].

Яркость бликов на экране монитора не должна превышать 40 кд/м^2[14].

Люминесцентные лампы, при применении общего освещения должны быть распложены в боковой стороне от рабочего места, параллельно уровня глаз пользователя в случае, когда экраны расположены в ряд.

В случае, когда компьютеры равномерно распределены по площади помещения, линии источников света должны быть расположены над рабочим местом ближе к передней части и направленны на оператора.

В целях обеспечения соответствующих параметров освещенности требуется проводить очистку осветительных приборов, а также окон и оконных поверхностей не менее чем два раза в год и вовремя осуществлять замену перегоревших ламп.

4.3 Требования к микроклимату

В помещениях, где происходит работа с ПЭВМ должны обеспечиваться оптимальные параметры микроклимата.

Температура воздуха должны быть в холодный период не более 22 – 24°С.

Температура воздуха в теплый период года 20° – 25°С.

Относительная влажность воздуха должна составлять Влажность воздуха составляет 40 – 60%, а подвижность воздуха – от 0,1 до 0,2 м/с [15].

Для повышения влажности воздуха в помещениях следует применять увлажнители воздуха, ежедневно заправлять их дистиллированной или кипяченой водой.

4.4 Требования к шуму

Длительное пребывание персоналах в рабочем помещением с высоким уровнем шума отрицательно сказывается на состоянии здоровья. Со временем человек начинает испытывать утомленность, головную боль, повышенное давление и общее недомогание. Все это приводит к значительному ухудшению производительности и качества работы. При длительном пребывании человека в помещении с интенсивностью шума свыше 80 дБА может привести к частичной или полной потере слуха.

В таблице 4.1 приводятся допустимые и эквивалентные параметры уровня звука по категориям тяжести и напряженности труда для различных видов трудовой деятельности и рабочих мест [16].

Таблица 4.1 – Предельно допустимые и эквивалентные уровни звука на рабочем месте для разных категорий труда в дБА [16]

Трудовой процесс по категории напряженности	Категория тяжести процесса труда				
	Легкая нагрузка	Средняя нагрузка	Тяжелый труд		
			I степень	II степень	III степень
Легкая степень напряженности	80	80	75	75	75
Средняя степень напряженности	70	70	65	65	65
Напряженность труда I степени	60	60	-	-	-
Напряженность труда II степени	50	50	-	-	-

Уровень шума в помещениях, предназначенных для работы с персональными компьютерами, не должен составлять более 50 дБА. В помещениях для размещения шумных вычислительных машин уровень шума не должен превышать 75 дБА [16].

Снижается же звуковое давление в данных помещениях с помощью звукопоглощающих материалов.

4.5 Требования к рабочему месту

Рабочие места с ПЭВМ должны располагаться так, чтобы естественный свет падал сбоку, желательно с левой стороны.

Обязательно нужно учитывать расстояние между рабочими столами с мониторами. Расстояние между боковыми поверхностями мониторов должен составлять не менее 1,2 м. Расстояние же между экраном и задней частью другого монитора не менее 2,0 м [17].

При выполнении творческой работы, которая требует высоких энергозатрат и умственного напряжения, рабочие места с ПЭВМ следует отделить друг от друга

перегородками высотой 1,5 – 2,0 м [17].

Рабочий стол может быть любой конструкции, отвечающий современным требованиям эргономики. При этом конструкция должна быть такой, чтобы оборудование, применяемое на рабочей поверхности, было размещено оптимальным образом. Поверхность рабочего стола должна иметь коэффициент отражения от 0,5 до 0,7 [17].

Конструкция рабочего кресла должна обеспечивать комфортное положение при работе с ПЭВМ, а также позволять изменить положения тела.

Рабочее кресло должно обязательно регулироваться по высоте и углам наклона сидения, спинки и расстоянию спинки от переднего края сиденья, быть подъемно-поворотным.

Верхняя отделка сидения, спинки и других частей кресла должна быть полумягкой, покрытие должно быть нескользящим, мало электризующимся и

4.6 Требования к организации и оборудованию рабочих мест

Обеспечения наилучших условий труда необходимо так, как повышает работоспособность и снижает утомляемость работников. Для этого следует организовать рабочее место согласно следующим рекомендациям:

1. Высота рабочей поверхности для пользователей ПЭВМ должна регулироваться в пределах 680–800мм. При отсутствии регулировки высота рабочей поверхности стола должна составлять 725мм[18];

2. Модульными размерами рабочей поверхности стола для ПЭВМ, на основании которых должны рассчитываться конструктивные размеры, следует считать ширину 800, 1000, 1200, 1400 мм, глубину 800 и 1000 мм при нерегулируемой его высоте, равной 725 мм [18];

3. Также необходимо чтобы рабочий стол имел пространство для ног. Высота, которого должна составлять не менее 600 мм, а ширина не менее 500 мм. Глубина же на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650 мм [18].

Также не маловажным является конструкция рабочего кресла.

Конструкцией рабочего кресла должны быть обеспечены:

− ширина и глубина поверхности сиденья не меньше 400 мм;

− поверхность сиденья, которая имеет закругленный передний край;

− регулировка по высоте сиденья в пределах 400 - 550 мм и углам наклона вперед до 15 град. и назад до 5 град.[19];

− высота опорной поверхности спинки 300 ± 20 мм, а ширина не меньше 380 мм и радиус кривизны горизонтальной плоскости 400 мм [19];

− угол наклона спинки в вертикальной плоскости в пределах 0±30 град.;

− регулировка расстояния спинки от переднего края сиденья от 260 до 400 мм;

− стационарные или съемные подлокотники длиной не менее 250 мм и шириной от 50 до 70 мм;

− регулировка подлокотников по высоте над сиденьем в пределах 230±30 мм и внутреннего расстояния между подлокотниками от 350 до 500 мм [19].

Располагать клавиатуру на рабочей плоскости стола необходимо на расстоянии 100 – 300 мм от переднего края, который обращен к пользователю или на специальной рабочей поверхности, регулируемой по высоте и изолированной от основной столешницы.

4.7 Режим труда и отдыха при работе за компьютером

Режим труда и отдыха предполагает длительное пребывание при работе на ПЭВМ и перерывов, которые регламентированы продолжительной рабочей сменой.

Виды трудовой деятельности при работе с ПЭВМ делятся на три группы: группа А – работа по считыванию информации с экрана с предварительным запросом; группа Б – работа по вводу информации; группа В – творческая работа в режиме диалога с ПК.

В случае, когда в течение рабочей смены пользователь при взаимодействии с ПЭВМ выполняет разные виды работ, основной работой считается та, на которую

отводится не менее 50% рабочего времени.

По видам трудовой деятельности разделяют три категории тяжести и напряженности при работе с ПЭВМ, уровень нагрузки для которых определяется как: суммарное число считываемых знаков – группа А; суммарное число считываемых или вводимых знаков –группа Б; общее время проведенной работы на ПЭВМ – группа В.

Таблица 4.2 – Категории тяжести и напряженности в зависимости от вида работы с ПЭВМ [19]

Категории тяжести работы с ПЭВМ	Нагрузка при видах работ за рабочую смену с ПЭВМ			Регламентированные перерывы, мин	
	группа А, кол-во знаков	группа Б, кол-во знаков	группа В, ч	Смена 8 часов	Смена 12 часов
I категория	До 20 тыс	До 15 тыс	2,0	50	80
II категория	До 40 тыс	До 30 тыс	4,0	70	110
III категория	До 60 тыс	До 40 тыс	6,0	90	140

Суммарное время регламентированных перерывов устанавливается в зависимости от категории трудовой деятельности и уровня нагрузки за рабочую смену.

Рекомендуется организовывать рабочий день таким образом, чтобы работа с использованием персонального компьютера чередовалась с какой-либо другой, в которой не используется ПЭВМ. Это необходимо для предотвращения преждевременной утомляемости работника.

При работе, характер которой предполагает постоянное взаимодействие с ВДТ (ввод текстов или набор данных и т.п.) с высокой степенью сосредоточенности и внимания, в случае, когда переключение на другие виды трудовой деятельности, не связанные с ПЭВМ невозможно, необходима организация перерывов на 10 – 15 мин через каждые 45 – 60 минут работы [19].

При непрерывной работе с видеотерминалами без регламентированного перерыва продолжительности не должна превышать 1 ч [19].

Независимо от категории и вида трудовой деятельности при работе с ПЭВМ в ночную смену следует увеличить продолжительность регламентированных перерывов на 30%.

В регламентированные перерывы, которые предназначены для снижения нервного и эмоционального напряжения, утомления и устранения влияния гиподинамии и гипокинезии, следует выполнять комплекс упражнений для глаз, с плечевого пояса и рук, туловища и ног. Целесообразно менять комплексы упражнений через 2–3 недели.

Для пользователей с высоким уровнем напряженности при работе с персональным компьютером, рекомендуется отдых в специально оборудованных помещениях.

4.8 Требования к электробезопасности

Очень важно уделять большое внимание безопасности при работе с ПЭВМ на рабочем месте. Не рекомендуется очищать ПЭВМ от пыли во включенном состоянии и работать непосредственно с компьютером во влажной одежде и влажными руками.

Перед началом работы с персональным компьютером необходимо убедиться в отсутствии различных свисающих проводов со стола, в целостности состояния проводом электропитания, в отсутствии повреждений рабочей аппаратуры, а также убедиться в наличии заземления экранного фильтра [19].

В процессе работы токи статического электричества, наведенные на корпус монитора, системного блока и клавиатуры, могут приводить к разрядам при прикосновении к этим элементам. Опасности для пользователя это не представляет, но может привести к перебоям при работе оборудованием. Снизить уровень стоков статического электричества можно используя нейтрализаторы, местное и общее увлажнители воздуха, покрытие полов с антистатической пропиткой.

Выводы по разделу четыре:

В данном разделе был проведен анализ опасных и вредных производственных факторов, изучены требования к безопасности.

Компьютеризация нашего общества уже давно вышла за рамки простой технической оснащенности. Сегодня компьютер неотъемлемая часть во всех сферах человеческой жизни. По этой же причине появились новые проблемы, а именно влияние компьютера на здоровье человека, которое характеризуется:

- постоянным сидячим положением;
- большим зрительным напряжением;
- однообразными повторяющимися нагрузками на плечевой пояс и руки.

Соблюдение условий, определяющих оптимальную организацию рабочего места пользователя, позволит сократить степень влияния компьютера на организм человека. Более того, меры направленные на защиту здоровья трудящихся, обеспечение безопасности условий труда позволит сохранить работоспособность пользователя в течение всего рабочего дня.

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе исследовались различные аспекты разработки для автоматизации работы диспетчера по расписанию в колледже.

Были проведены функциональные требования и проведён анализ существующих разработок, а также выявлены их достоинства и недостатки. В соответствии с функциональными требованиями был выбран язык программирования Python и фреймоворк «Django», разработана БД, web-приложение, и описан интерфейс пользователя.

В экономическом разделе, определены затраты на разработку системы и показатели экономического эффекта.

Немалую роль играет обеспечение безопасности и комфорта пользователя. Поэтому при разработке приложения большое внимание было уделено проектированию дружественного и эргономичного интерфейса, приведены инструкции по использованию программы и рекомендации по организации рабочего места и режима работы пользователя.

В результате проделанной работы было разработано программное обеспечение для автоматизации работы диспетчера в колледже.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Жаринов, К.В. Основы web-мастеринга: учебник / К.В. Жаринов. – СПб.: БХВ-Петербург, 2003. – 352 с.
2. Эрик Фримен – Паттерны проектирования / Эрик Фримен, Элизабет Фримен, Кэтти Сьерра, Берт Бейтс. СПб.: Питер, 2017.– 656 с.
3. Хомоненко, А. Д. Базы данных: учебник / А. Д. Хомоненко, В.М. Цыганков, М.Г. Мальцев. – СПб.: КОРОНА-Век, 2009. – 736 с.
4. Ржеуцкая, С.Ю. Базы данных. Язык SQL: учеб.пособие / С.Ю. Ржеуцкая – Вологда: ВоГТУ, 2010. – 159с.
5. Холл, М. Программирование для Web. Библиотека профессионала: учебник / М. Холл, Л. Браун. – М.: Вильямс, 2002. – 1264 с.
6. Дмитриева, М.В. Самоучитель JavaScript: учебник / М.В. Дмитриева. – СПб.: БХВ-Петербург, 2003. – 512 с.
7. Никсон, Р. Создаем динамические web-сайты с помощью PHP, MySQL и JavaScript: учебник / Робин Никсон; пер. с англ. Н. Вильчинский. –СПб.: Питер, 2011. – 496 с.
8. Холмогоров, В.В. Основы Web-мастерства: учебник / В.В. Холмогоров. – СПб.: Питер, 2001. – 352 с.
9. Мэтью, Д. HTML5. Разработка web-приложений: учебник / Д. Мэтью. – М.: Рид Групп, 2012. – 320 с.
10. Пьюривал, С. Основы разработки web-приложений: учебник / С. Пьюривал.– СПб.: Питер, 2015. – 272 с.
11. Михайлова, Э. А. Экономическая оценка инвестиций: Учебное пособие. / Э.А. Михайлова., Л.Н. Орлова. – Рыбинск: РГАТА, 2008. – 176 с.
12. Налоговый кодекс Российской Федерации от 05.08.2000 № 117- ФЗ // «Собрание законодательства РФ», 07.08.2000, N 32, ст. 3340.
13. ГОСТ 12.2.032-78 Рабочее место при выполнении работ сидя. Общие эргономические требования. //Справочно-правая система «КонсультантПлюс». –

http://www.consultant.ru/cons/cgi/online.cgi?req=doc;base=STR;n=6342#0 [дата обращения – 27.11.2017].

14. СП 52.13330.2011 Свод правил. Естественное и искусственное освещение. – М.: Минрегион России, 2011.

15. ГОСТ 12.1.005-88 Система стандартов безопасности труда. Общие санитарно-гигиенические требования к воздуху рабочей зоны. – М.: ИПК Издательство стандартов, 2002.

16. СанПиН 2.2.4/2.1.8.562-96 «Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки» //Справочно-правая система «Гарант» http://base.garant.ru/4174553/ [дата обращения – 27.11.2017]

17. ГОСТ 22269-76. Система «Человек-машина». Рабочее место оператора. Взаимное расположение элементов рабочего места. Общие эргономические требования. – М.: Издательство стандартов, 1990.

18. Беляков, Г.И. Безопасность жизнедеятельности. Охрана труда: Учебник для бакалавров / Г.И. Беляков. – М.: Юрайт, 2013. – 572 с.

19. ГОСТ 12.1.038-82 Система стандартов безопасности труда. Электробезопасность. Предельно допустимые значения напряжений прикосновения и токов.– М.: Издательство стандартов, 2001.

20. Методические рекомендации по подготовке и оформлению выпускной квалификационной работы (проекта) для технических направлений подготовки 09.03.01 Информатика и вычислительная техника, 09.03.04 Программная инженерия, 12.03.01 Приборостроение, 23.03.01 Технология транспортных процессов / сост. Л.Н.Буйлушкина. - Нижневартовск, 2017. - 35с.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД WEB-ПРИЛОЖЕНИЯ «ИНФОРМАЦИОННАЯ СИСТЕМА ДЛЯ АВТОМАТИЗАЦИИ ВЕДЕНИЯ И АВТОМАТИЧЕСКОГО ЗАПОЛНЕНИЯ РАСПИСАНИЯ КОЛЛЕДЖА»

Файл models.py: описание структуры БД через ORM.

```
from django.db import models
from django.db import utils

class Kafedra(models.Model):
    caption = models.CharField("Кафедра",max_length=100)
    def str(self):
        return self.caption
    class Meta:
        verbose_name = 'Кафедра'
        verbose_name_plural = 'Кафедра'

class TeacherQualification(models.Model):
    caption = models.CharField("Квалификация",max_length=100)
    def str(self):
        return str(self.caption)
    class Meta:
        verbose_name = 'справочник квалификаций'
        verbose_name_plural = 'Квалификация'

class Profile(models.Model):
    caption = models.CharField("Профиль",max_length=100)
    def str(self):
        return str(self.caption)
    class Meta:
        verbose_name = 'профиль'
        verbose_name_plural = 'профили'

class TypeSubject(models.Model):
    caption = models.CharField("тип занятий",max_length=100)
    def str(self):
        return str(self.caption)
    class Meta:
        verbose_name = 'тип занятий'
        verbose_name_plural = 'тип занятий'

class Subject(models.Model):
    caption = models.CharField("Дисциплина",max_length=100)
    kafedra = models.ForeignKey(Kafedra)
    profile = models.ManyToManyField(Profile)
    def str(self):
        return str(self.caption)
```

```
    class Meta:
        verbose_name = 'Дисциплина'
        verbose_name_plural = 'Дисциплины'

class Teacher (models.Model):
    fio = models.CharField(max_length=100)
    kafedra = models.ForeignKey(Kafedra)
    qualification = mod-els.ManyToManyField(TeacherQualification)
    subjects = models.ManyToManyField(Subject)
    total_exp = models.DateField()
    local_exp = models.DateField()
    week_load = models.PositiveSmallIntegerField()
    def str(self):
        return self.fio
    class Meta:
        verbose_name = 'Преподователь'
        verbose_name_plural = 'Преподователи'

class RoutGroup(models.Model):
    caption = models.CharField(max_length=100)
    def str(self):
        return str(self.caption)
    class Meta:
        verbose_name = 'справочник специальностей'
        verbose_name_plural = 'справочник специальностей'

class Group(models.Model):
    K1 = '1'
    K2 = '2'
    K3 = '3'
    K4 = '4'
    NUM_KURS = (
        (K1,'первый курс'),
        (K2,'второй курс'),
        (K3,'третий курс'),
        (K4,'четвёртый курс'),
        )
    caption = models.CharField("Группа", max_length=100)
    rout = models.ForeignKey(RoutGroup)
    num_kurs = models.CharField(max_length = 1,
                                choices = NUM_KURS,
                                default = K1)
    def str(self):
        return str(self.caption)
    class Meta:
        verbose_name = 'Группа'
        verbose_name_plural = 'Группы'

class ClassRoom(models.Model):
    caption = models.CharField(max_length=100)
    profile = models.ManyToManyField(Profile)
    def str(self):
        return str(self.caption)
```

```
    class Meta:
        verbose_name = 'Кабинет'
        verbose_name_plural = 'Кабинеты'

class SubjectNumber (models.Model):
    num = models.PositiveSmallIntegerField()
    time_start = models.TimeField(unique = True)
    time_end = models.TimeField(unique = True)
    def str(self):
        return str(self.num) + " пара"
    class Meta:
        verbose_name = 'Расписание звонков'
        verbose_name_plural = 'Расписание звонков'

class Semestr (models.Model):
    num = models.PositiveSmallIntegerField()
    date_start = models.DateField(unique = True)
    date_end = models.DateField(unique = True)
    def str(self):
        return str(self.num) + " семестр"
    class Meta:
        verbose_name = 'семестр'
        verbose_name_plural = 'семестры'

class WeekTimeTable (models.Model):
    MONDAY = 'пн'
    TUESDAY = 'вт'
    WEDNESDAY = 'ср'
    THURSDAY = 'чт'
    FRIDAY = 'пт'
    SATURDAY = 'сб'
    SUNDAY = 'вс'
    WEEK_DAY = (
        (MONDAY,'понедельник'),
        (TUESDAY,'вторник'),
        (WEDNESDAY,'среда'),
        (THURSDAY,'четверг'),
        (FRIDAY,'пятница'),
        (SATURDAY,'суббота'),
        (SUNDAY,'воскресенье'),
        )
    day = models.CharField(max_length = 2,
                           choices = WEEK_DAY,
                           default = MONDAY)
    subject_num = models.ForeignKey(SubjectNumber)
    group = models.ForeignKey(Group)
    subject_type = models.ForeignKey(TypeSubject,
                                     verbose_name = "Тип занятия",
                                     blank = True,
                                     null = True)
    subject = models.ForeignKey(Subject,
                                blank = True,
                                null = True)
```

```
    class_room = models.ForeignKey(ClassRoom,
                                   blank = True,
                                   null = True)
    teacher = models.ForeignKey(Teacher,
                                blank = True,
                                null = True)
    is_even = models.BooleanField(default = True)
    semestr = models.ForeignKey(Semestr)


    def str(self):
        return str(self.day) + " " + str(self.subject_num) + " " +
               str(self.group) + " " + str(self.subject)
    class Meta:
        verbose_name = 'Расписание'
        verbose_name_plural = 'Расписание'
        unique_together = (
             ('group','day','subject_num','is_even', 'semestr'),
             ('day','subject_num','class_room','is_even','semestr'),
             ('day','subject_num','teacher','is_even','semestr'))

class WeekTimeTableChange (models.Model):
    MONDAY = 'пн'
    TUESDAY = 'вт'
    WEDNESDAY = 'ср'
    THURSDAY = 'чт'
    FRIDAY = 'пт'
    SATURDAY = 'сб'
    SUNDAY = 'вс'
    WEEK_DAY = (
        (MONDAY,'понедельник'),
        (TUESDAY,'вторник'),
        (WEDNESDAY,'среда'),
        (THURSDAY,'четверг'),
        (FRIDAY,'пятница'),
        (SATURDAY,'суббота'),
        (SUNDAY,'воскресенье'),
        )
    day = models.CharField(max_length = 2,
                           choices = WEEK_DAY,
                           default = MONDAY)
    subject_num = models.ForeignKey(SubjectNumber)
    group = models.ForeignKey(Group)
    subject = models.ForeignKey(Subject)
    class_room = models.ForeignKey(ClassRoom)
    teacher = models.ForeignKey(Teacher)
    def unicode(self):
        return str(self.Group.caption)

class SubjectLoadForGroup(models.Model):
    group = models.ForeignKey(Group, verbose_name = "Группа")
    subject = models.ForeignKey(Subject,verbose_name = "Дисциплина")
```

```
    subject_type = models.ForeignKey(TypeSubject,
                                     verbose_name = "Тип занаятия")
    hours = models.PositiveSmallIntegerField("Часы")
    sem_exam = models.ManyToManyField(Semestr,
                                      verbose_name = "Семестры")
    teacher = models.ForeignKey(Teacher,
                                blank = True,
                                null = True,
                                verbose_name = "Преподаватель")
    may_cabs = models.ManyToManyField(
                               ClassRoom,
                               blank = True,
                               verbose_name = "Доступные кабинеты")
    def str(self):
        return str(self.id)+" "+str(self.group) + " " +
               str(self.subject)+ " " + str(self.subject_type)
    class Meta:
        verbose_name = 'Нагрузка предметов на группы'
        verbose_name_plural = 'Нагрузка предметов на группы'

class SubjectLoadForGroupTemp(models.Model):
    group = models.ForeignKey(Group,
                              verbose_name = "Группа")
    subject = models.ForeignKey(Subject,
                                verbose_name = "Дисциплина")
    subject_type = models.ForeignKey(TypeSubject,
                                verbose_name = "Тип занятия")
    is_even = models.BooleanField(default = True)
    not_every_week = models.BooleanField(default = False)
    over_not_every_week = models.BooleanField(default = False)
    sem = models.ForeignKey(Semestr)
    teacher = models.ForeignKey(Teacher, blank = True,null = True)
    may_cabs = models.ManyToManyField(ClassRoom, blank = True)
    stand = models.BooleanField(default = False)
    def str(self):
        return str(self.group) + " " + str(self.subject)+ " " +
               str(self.subject_type)
    class Meta:
        verbose_name = 'Промежуточная таблица для заполнения
                        расписания'
        verbose_name_plural = 'Промежуточная таблица для заполнения
                               расписания'
```

Файл view.py: представления.

```
from django.shortcuts import render, render_to_response,redirect,
get_object_or_404
from django.http import HttpResponse
from datetime import date, timedelta
```

```
from django.views.generic.edit import CreateView, UpdateView,
DeleteView
from django.contrib.auth.decorators import login_required
from django.urls import reverse_lazy
from .models import Teacher,Kafedra, WeekTimeTable, SubjectNumber,
Group, Subject,ClassRoom,RoutGroup,SubjectLoadForGroup, Semestr, Sub-
jectLoadForGroupTemp
from .forms import SubjectLoadForGroupForm
from django import forms

class PlanCreate(CreateView):
    model = SubjectLoadForGroup
    fields = 'all'
    success_url = reverse_lazy('reference_loadgroup_list')


class PlanUpdate(UpdateView):
    model = SubjectLoadForGroup
    teacher = Teacher.objects.filter(id = 1)
    fields = ['group','subject','subject_type','hours', 'sem_exam']
    success_url = reverse_lazy('reference_loadgroup_list')

class PlanDelete(DeleteView):
    model = SubjectLoadForGroup
    success_url = reverse_lazy('reference_loadgroup_list')

def add_teacher_to_plan(request, pk):
    plan = get_object_or_404(SubjectLoadForGroup, pk = pk)
    teach = Teacher.objects.filter(id = 1)
    if request.method == "POST":
        form = SubjectLoadForGroupForm(request.POST, instance=plan)
        if form.is_valid():
            plan = form.save(commit = False)
            plan.save()
            return redirect('reference_loadgroup_list')
    else:
        form = SubjectLoadForGroupForm(instance = plan)
    return render(request,'reference/loadgroup_edit.html',
{'form':form})

def reference(request):
    return render(request, 'reference/reference.html')

def reference_loadgroup_list(request):
    group_one = ''
    cho_rout = ''
    cho_kurs = ''
    filter_table = SubjectLoadForGroup.objects.all().order_by
                                                    ('subject')
    if 'cho_rout' in request.GET and request.GET['cho_rout']:
        cho_rout = request.GET['cho_rout']
        if 'cho_kurs' in request.GET and request.GET['cho_kurs']:
            cho_kurs = request.GET['cho_kurs']
```

```
            rout_one = RoutGroup.objects.filter(caption = cho_rout)
            group_one = Group.objects.filter(rout = rout_one,
                                             num_kurs = cho_kurs)
            filter_table = list(
                        SubjectLoadForGroup.objects.order_by(
                        'subject').filter(group = group_one))
    return render(request, 'reference/loadgroup_list.html', {
          'group_one': group_one,
          'num_list' : SubjectNumber.objects.all(),
          'rout' : RoutGroup.objects.all(),
          'cho_rout' : cho_rout,
          'cho_kurs' : cho_kurs,
          'filter_table':filter_table,
      })


def reference_teacher_list(request):
    cho_kaf = ''
    kaf_one = ''
    filter_table = Teacher.objects.all().order_by('fio')
    if 'cho_kaf' in request.GET and request.GET['cho_kaf']:
        cho_kaf = request.GET['cho_kaf']
        kaf_one = Kafedra.objects.filter(caption = cho_kaf)

        filter_table = Teacher.objects.order_by('fio').
                                filter(kafedra__in = kaf_one)
    else:
        print ('')
    return render(request, 'reference/teacher_list.html', {
        'kaf' : Kafedra.objects.all(),
        'kaf_one' : kaf_one,
        'cho_kaf' : cho_kaf,
        'filter_table':filter_table,
    })

def reference_loadgroup_edit(request):
    if request.method == "POST":
        form = SubjectLoadForGroupForm(request.POST)
        obj = SubjectLoadForGroup(request.POST)
        if form.is_valid():
            tmg = form.save(commit = False)
            tmg.save()
            return redirect('/',pk=tmg.pk)
    else:
        form = SubjectLoadForGroupForm()
    return render(request, 'reference/loadgroup_edit.html', {
        'form':form,
        'obj': obj,
        })

def teacher_group_subject(request):
    if request.method == "POST":
        form = TeacherMayGroupForm()
```

```
        if form.is_valid():
            tmg = form.save(commit = False)
            tmg.save()
            return redirect('sprav.html',pk=tmg.pk)
    else:
        form = TeacherMayGroupForm()
    return render(request, 'sprav.html', {'form':form})

def teacher_list(request):
    return render_to_response('teacher_list.html', {
        'teacher': Teacher.objects.all().order_by('fio'),
    })

def index(request):
    weekday = ['пн','вт','ср','чт','пт','сб']
    return render(request, 'index.html', {
        'weekday' : weekday,
        'ttw_list':  list(WeekTimeTable.objects.all()
                                        .order_by('group')),
        'groups': Group.objects.all().order_by('num_kurs'),
        'num_list': SubjectNumber.objects.all(),
        'rout': RoutGroup.objects.all(),

    })

def index_view_table(request):
    weekday = ['пн','вт','ср','чт','пт','сб']
    nk = ['1','2','3','4']
    if 'cho_rout' in request.GET and request.GET['cho_rout']:
        cho_rout = request.GET['cho_rout']
        if 'cho_kurs' in request.GET and request.GET['cho_kurs']:
            cho_kurs = request.GET['cho_kurs']
            rout_one = RoutGroup.objects.filter(caption = cho_rout)
            group_one = Group.objects.filter(rout = rout_one,
num_kurs = cho_kurs)
            filter_table = list(WeekTimeTable.objects.filter(group =
group_one))
     return render(request, 'index.html', {
        'weekday' : weekday,
        'ttw_list':  filter_table,
        'group_one': group_one,
        'num_list' : SubjectNumber.objects.all(),
        'rout' : RoutGroup.objects.all(),
        'cho_rout' : cho_rout,
        'cho_kurs' : cho_kurs,
    })

def print_tt(request):
    weekday = ['пн','вт','ср','чт','пт','сб']
    semestr = Semestr.objects.all()
    return render(request, 'print_tt.html', {
        'weekday' : weekday,
```

```
        'ttw_list':  list(WeekTimeTable.objects.all().
                                                order_by('group')),
        'groups': Group.objects.all().order_by('num_kurs'),
        'num_list' : SubjectNumber.objects.all(),
        'semestr' : semestr,
    })

def create_hole(request):
    return render(request, 'create_hole.html')
@login_required

def create_or_update_blank_table(request):
    def create_for(even):
        WEEK_DAY = ['пн','вт','ср','чт','пт','сб']
        ttw = WeekTimeTable.objects.all()
        group_list = Group.objects.all()
        subject_list = SubjectNumber.objects.all()
        semestr_list = Semestr.objects.all()
        for sem in semestr_list:
            semestr = sem
            for gr in group_list:
                group = gr
                for wd in WEEK_DAY:
                    day = wd
                    for sn in subject_list:
                        subject_num = sn
                        try:
                            ttw.create(day = day,
                                       subject_num = subject_num,
                                       group = group,
                                       is_even = even,
                                       semestr = semestr)
                        except:
                            print("Строка уже существует")

    if request.method == 'POST':
        is_even = True
        is_not_even = False
        create_for(is_even)
        create_for(is_not_even)

    return HttpResponse()

def fill_ttw(request):

    if request.method == 'POST':
        ttws = WeekTimeTable.objects.all()
        plans = SubjectLoadForGroup.objects.all()
        ttw_temp = SubjectLoadForGroupTemp.objects.all()

semestr = Semestr.objects.get(pk = 1)
        count_week = (semestr.date_end-semestr.date_start).days//7
```

```
#кол-во целых недель в 1 семестре
        col_ost_day = (semestr.date_end-semestr.date_start).days%7
#кол-во дней без недель в первом семестре
        semestr2 = Semestr.objects.get(pk = 2)
        count_week2 = (
semestr2.date_end-semestr2.date_start).days//7 )
#кол-во целых недель в 2 семестре
        col_ost_day =  (semestr.date_end-semestr.date_start).days%7
#кол-во дней без недель в втором семестре
        for plan in plans:
            all_sem = plan.sem_exam.all()
            if (semestr in all_sem and all_sem.count()==1):
                print("только первый семестр")
#узнаём количество часов и для четной и для не четной недели
                hour_in_every_week = plan.hours//count_week
#смотрим, сколько часов мы не можем распределить поравну:
                free_hour_in_year = plan.hours%count_week
                even_count_week = count_week//2
                if free_hour_in_year >= even_count_week:
                    hour_for_even_week = 1

free_hour_in_year=free_hour_in_year%even_count_week
                else:
                    hour_for_even_week = 0
                print(plan.subject_type, plan.subject, "часы каждую
неделю:", hour_in_every_week,"часы в парную
неделю",hour_for_even_week, "не делящиеся часы:", free_hour_in_year)
                i = 0
                all_cab = plan.may_cabs.all()
                i = 0
                while i < hour_in_every_week:
                    try:
                        ttw_temp.create(
                                        subject = plan.subject,
                                        subject_type =
                                                  plan.subject_type,
                                        group = plan.group,
                                        sem = semestr,
                                        teacher = plan.teacher,
                                        )

                        i+=1
                    except:
                        print("skip")
                i = 0
                while i < hour_in_every_week:
                    try:
                        ttw_temp.create(
                                        subject = plan.subject,
                                        subject_type
= plan.subject_type,
                                        group = plan.group,
                                        sem = semestr,
```

```
                                    teacher = plan.teacher,
                                    is_even = False
                                    )
                        i+=1
                    except:
                        print("skip")
            i = 0
            while i < hour_for_even_week:
                try:
                    ttw_temp.create(
                                    subject = plan.subject,
                                    subject_type
                                        = plan.subject_type,
                                    group = plan.group,
                                    sem = semestr,
                                    teacher = plan.teacher,
                                    not_every_week = True
                                    )
                    i+=1
                except:
                    print("skip")
            i = 0
            while i < free_hour_in_year:
                try:
                    ttw_temp.create(
                                    subject = plan.subject,
                                    subject_type
                                        = plan.subject_type,
                                    group = plan.group,
                                    sem = semestr,
                                    teacher = plan.teacher,
                                    over_not_every_week = True
                                    #,may_cabs = all_cab
                                    )
                    i+=1
                except:
                    print("skip")
        elif (semestr2 in all_sem and all_sem.count()==1):
            print("только второй семестр")
            hour_in_every_week2 = plan.hours//count_week2
            free_hour_in_year2 = plan.hours%count_week2
            even_count_week2 = count_week2//2
            if free_hour_in_year2 >= even_count_week2:
                hour_for_even_week2 = 1

        free_hour_in_year2=free_hour_in_year2%even_count_week2
            else:
                hour_for_even_week2 = 0

            print(plan.subject_type, plan.subject,
                            "часы каждую неделю:",
                            hour_in_every_week2,
                            "часы в парную неделю",
```

```
                    hour_for_even_week2,
                    "не делящиеся часы:",
                    free_hour_in_year2)
i = 0
all_cab = plan.may_cabs.all()
#для пар в чётной и нечетной недели
i = 0
while i < hour_in_every_week2:
    try:
        ttw_temp.create(
                        subject = plan.subject,
                        subject_type
                            = plan.subject_type,
                        group = plan.group,
                        sem = semestr2,
                        teacher = plan.teacher
                        )
        i+=1
    except:
        print("skip")
i = 0
while i < hour_in_every_week2:
    try:
        ttw_temp.create(
                        subject = plan.subject,
                        subject_type
                            = plan.subject_type,
                        group = plan.group,
                        sem = semestr2,
                        teacher = plan.teacher,
                        is_even = False
                        #,may_cabs = all_cab
                        )
        i+=1
    except:
        print("skip")
i = 0
while i < hour_for_even_week2:
    try:
        ttw_temp.create(
                        subject = plan.subject,
                        subject_type
                            = plan.subject_type,
                        group = plan.group,
                        sem = semestr2,

                        teacher = plan.teacher,
                        not_every_week = True)
        i+=1
    except:
        print("skip")
i = 0
while i < free_hour_in_year2:
```

```
                try:
                    ttw_temp.create(
                                    subject = plan.subject,
                                    subject_type
                                        = plan.subject_type,

                                    group = plan.group,
                                    sem = semestr2,
                                    teacher = plan.teacher,
                                    over_not_every_week = True)
                    i+=1
                except:
                    print("skip")
            elif (all_sem.count()==2):
                half_ph = plan.hours//2
                hour_in_every_week = half_ph//count_week
                hour_in_every_week2 = half_ph//count_week2
                free_hour_in_year = plan.hours%2
                free_hour_in_year += half_ph%count_week
                free_hour_in_year += half_ph%count_week2
                even_count_week = count_week//2
                even_count_week2 = count_week2//2
                if free_hour_in_year >= even_count_week:
                    hour_for_even_week = 1
                 free_hour_in_year=free_hour_in_year%even_count_week
                else:
                    hour_for_even_week = 0
                print(plan.subject_type,
                    plan.subject,
                   "часы каждую неделю:",
                    hour_in_every_week,
                    "часы в парную неделю",
                    hour_for_even_week,
                    "не делящиеся часы:",
                    free_hour_in_year)
                i = 0
                all_cab = plan.may_cabs.all()
                #для пар в чётной и нечетной недели
                i = 0
                while i < hour_in_every_week:
                    try:
                        ttw_temp.create(
                                        subject = plan.subject,


                                        subject_type
                                            = plan.subject_type,
                                        group = plan.group,
                                        sem = semestr,
                                        teacher = plan.teacher
                                        )
                        i+=1
                    except:
```

```
                    print("skip")
                i = 0
                while i < hour_in_every_week:
                    try:
                        ttw_temp.create(
                                        subject = plan.subject,
                                        subject_type
                                            = plan.subject_type,
                                        group = plan.group,
                                        sem = semestr,
                                        teacher = plan.teacher,
                                        is_even = False)
                        i+=1
                    except:
                        print("skip")
                i = 0
                while i < hour_for_even_week:
                    try:
                        ttw_temp.create(
                                        subject = plan.subject,
                                        subject_type
                                            = plan.subject_type,
                                        group = plan.group,
                                        sem = semestr,
                                        teacher = plan.teacher,
                                        not_every_week = True
                                        )
                        i+=1
                    except:
                        print("skip")
                i = 0
                while i < free_hour_in_year:
                    try:
                        ttw_temp.create(

                                        subject = plan.subject,
                                        subject_type
                                            = plan.subject_type,
                                        group = plan.group,
                                        sem = semestr,
                                        teacher = plan.teacher,
                                        over_not_every_week = True)


                        i+=1
                    except:
                        print("skip")
        return HttpResponse("suc")

def auto_create__ttw(request):
    if request.method == 'POST':
        sn12 = SubjectNumber.objects.all()[0:2]
        sn34 = SubjectNumber.objects.all()[2:4]
```

```
    sn56 = SubjectNumber.objects.all()[4:6]
    sn78 = SubjectNumber.objects.all()[6:8]
    sn910 = SubjectNumber.objects.all()[8:10]
    ttws_list12 = WeekTimeTable.objects.filter(
                        subject_num__in = sn12).order_by('day')
    ttws_list34 = WeekTimeTable.objects.filter(
                        subject_num__in = sn34).order_by('day')
    ttws_list56 = WeekTimeTable.objects.filter(
                        subject_num__in = sn56).order_by('day')
    ttw_temp_list = SubjectLoadForGroupTemp.objects.all()
for ttw_temp in ttw_temp_list:
        max_count_rep = 2
        count_rep = 0
        for ttw in ttws_list12:

            if (ttw.group == ttw_temp.group and
                ttw.subject == None and
                ttw.is_even == ttw_temp.is_even and
                ttw.semestr == ttw_temp.sem and
                ttw_temp.stand == False and
                ttw_temp.not_every_week == False and
                ttw_temp.over_not_every_week == False
                ):
                try:
                    ttw.subject = ttw_temp.subject
                    ttw.subject_type = ttw_temp.subject_type
                    ttw.teacher = ttw_temp.teacher
                    ttw.save()
                    ttw_temp.stand = True
                    ttw_temp.save()
                except:
                    print("skip")
    for ttw_temp in ttw_temp_list:
        max_count_rep = 2
        count_rep = 0
        for ttw in ttws_list34:

            if (ttw.group == ttw_temp.group and
                ttw.subject == None and
                ttw.is_even == ttw_temp.is_even and
                ttw.semestr == ttw_temp.sem and
                ttw_temp.stand == False and
                ttw_temp.not_every_week == False and
                ttw_temp.over_not_every_week == False
                ):
                try:
                    ttw.subject = ttw_temp.subject
                    ttw.subject_type = ttw_temp.subject_type
                    ttw.teacher = ttw_temp.teacher
                    ttw.save()
                    ttw_temp.stand = True
                    ttw_temp.save()
                except:
```

```
                    print("skip")
        for ttw_temp in ttw_temp_list:
            max_count_rep = 2
            count_rep = 0
            #while count_rep < max_count_rep:
            for ttw in ttws_list56:
                if (ttw.group == ttw_temp.group and
                    ttw.subject == None and
                    ttw.is_even == ttw_temp.is_even and
                    ttw.semestr == ttw_temp.sem and
                    ttw_temp.stand == False and
                    ttw_temp.not_every_week == False and
                    ttw_temp.over_not_every_week == False
                    ):
                    try:
                        ttw.subject = ttw_temp.subject
                        ttw.subject_type = ttw_temp.subject_type
                        ttw.teacher = ttw_temp.teacher
                        ttw.save()
                        ttw_temp.stand = True
                        ttw_temp.save()
                        #count_rep+=1
                    except:
                        print("skip")

        return HttpResponse("suc")
```

Файл index.html: Разметка главной страницы. Вывод расписания.

```
{% extends 'base1.html' %}
{% block content %}
  <div class="row">
  <div class="col-md-offset-2 col-md-8 col-lg-offset-3 col-lg-6">
  <div class="col-sm-12">
      <div class="col-xs-12 col-sm-8">
  <form action="{% url 'index_view_table' %}" method="GET" class="">
  <select  class="selectpicker" data-live-search="true" name =
  "cho_rout">
  <option disabled>Выберете направление</option>
  {% for r in rout %}
  <option value="{{r}}"
  {% if cho_rout == r %} selected {% endif %}>{{r}}
  </option>
  {% endfor %}
  </select>
  <select  class="selectpicker" name="cho_kurs">
  <option disabled>Выберете курс</option>
  <option value="1"{% if cho_kurs == "1" %} selected
  {% endif %}>1 курс
  </option>
      <option value="2"{% if cho_kurs == "2" %} selected {% endif
  %}>2 курс</option>
```

```
    <option value="3"{% if cho_kurs == "3" %} selected {% endif
%}>3 курс</option>
    <option value="4"{% if cho_kurs == "4" %} selected {% endif
%}>4 курс</option>
</select>
<button type="submit" value="Search" class="save btn btn-
default">показать</button>
<h4> Расписание для группы  {{cho_rout}} {{group_one.0}} </h4>
</form>
</div>
</div>
</div>
</div>
{% if cho_kurs  %}
<table class="table table-bordered table-hover table-sm">
<tbody>
   <thead class="thead-default">
<tr>
    <th width="10">Пара</th>
    <th width="20">Время</th>
    {% for day in weekday %}
    <th>{{day}}</th>
    {% endfor %}
</tr>
</thead>
{%for number in num_list%}
    <tr class="row3">
      <td class="column"> {{number.num}}</td>
      <td >{{number.time_start}} - {{number.time_end}} </td>
          {% for ttw in ttw_list %}
            {% if ttw.subject_num.num == number.num
and ttw.semestr.num == 1 and ttw.is_even == True %}
              <td class="column">
              <p>
              {{ttw.subject|default_if_none:" "}}
              <i>{{ttw.subject_type|default_if_none:" "}}</i>
              <p
style="color:grey;">{{ttw.class_room|default_if_none:" "}}
              <p>{{ttw.teacher|default_if_none:" "}}</td>
{% endif %}
          {% endfor %}
 {% endfor %}
</tr>
</tbody>
</table>
 {% endif %}
 {% endblock content %}
```

Весь исходный код можно посмотреть в ПРИЛОЖЕНИЕ Б.КОМПАКТ – ДИСК.

ПРИЛОЖЕНИЕ Б.КОМПАКТ-ДИСК

Содержание:

1. Пояснительная записка.

2. Презентация.

3. Web-приложение «Информационная система для автоматизации ведения и автоматического заполнения расписания колледжа».

4. Исходный код программы.